
Documentazione di LVGL v9.4

20251109 (Ita)

Release 9.4

LVGL Community

(Traduzione: [Baldassarre Cesarano](#))

09 nov 2025

Indice

1	Introduzione	1
2	Per iniziare	7
3	Esempi	15
4	Integrazione di LVGL	108
5	Funzionalità Comuni dei Widget	254
6	Tutti i Widget	325
7	Moduli Principali	459
8	LVGL Pro e XML	581
9	Moduli Ausiliari	627
10	Librerie di Terze Parti	643
11	Debugging	679
12	Contribuire	694
13	Log delle Modifiche	703
14	API	719

CAPITOLO 1

Introduzione

LVGL (Light and Versatile Graphics Library) è una libreria grafica C gratuita e open source che fornisce tutto il necessario per creare un'interfaccia utente grafica embedded con elementi grafici facili da usare, splendidi effetti visivi e un ridotto utilizzo di memoria.

LVGL fornisce il "motore GUI" che gestisce tutte le interazioni dell'utente finale con l'applicazione, tra cui non solo la visualizzazione del contenuto del display, l'accettazione dell'input dell'utente e la sua conversione in eventi "catturabili" dall'applicazione, ma anche oltre 30 widget integrati, oltre 100 proprietà di stile, scrolling [scorrimento], animazioni, software di rendering avanzato, supporto GPU integrato da MCU a OpenGL e molto altro. La combinazione di questi elementi consente di sviluppare interfacce utente in modo efficiente senza dover reinventare la ruota.

L'utilizzo di LVGL in un progetto firmware riduce efficacemente il lavoro dello sviluppatore della GUI alla sua definizione e alla fornitura di hook di eventi appropriati per svolgere il prezioso lavoro della GUI.

LVGL può utilizzare GUI definite al 100% nel codice C scritto dall'utente (per cui qualsiasi aggiornamento alla GUI richiede la modifica del codice C e il re-flashing del firmware), oppure definizioni di GUI in file XML (per cui gli aggiornamenti alla GUI richiedono solo la modifica dei file XML che definiscono la GUI per quel firmware), o qualsiasi combinazione di queste due opzioni adatta al progetto.

1.1 Caratteristiche Principali

- Elementi costitutivi potenti come *pulsanti, grafici, elenchi, slider, immagini*, ecc.
- Grafica avanzata con animazioni, anti-aliasing, opacità, scorrimento fluido
- Vari dispositivi di input come touchpad, mouse, tastiera, encoder, ecc.
- Supporto multilingua con codifica UTF-8
- Supporto multi-display, anche con formati di colore misti
- Elementi grafici completamente personalizzabili con stili simili a CSS
- Indipendente dall'hardware: utilizzabile con qualsiasi microcontrollore o display
- Scalabile: in grado di funzionare con poca memoria (64 kB Flash, 16 kB RAM)
- *OS*, memoria esterna e *GPU* sono supportati ma non obbligatori
- Funzionamento con "framebuffer" singolo anche con effetti grafici avanzati
- Scritto in C per la massima compatibilità (anche col C++)
- *Simulatore* per avviare la progettazione di GUI embedded su un PC senza hardware integrato

- Il codice utente sviluppato tramite simulatore può essere condiviso con il firmware per rendere più efficiente lo sviluppo dell'interfaccia utente.
- Binding di [MicroPython](#)
- Tutorial, esempi, temi per la progettazione rapida di interfacce grafiche (GUI)
- Documentazione disponibile online
- Gratuito e open source con licenza MIT
- Gratuito per progetti commerciali

1.1.1 Requisiti

In pratica, qualsiasi controller moderno in grado di pilotare un display è adatto all'esecuzione di LVGL. I requisiti minimi sono:

- Microcontrollore o processore a 16, 32 o 64 bit
- Velocità di clock consigliata >16 MHz
- Flash/ROM: >64 kB per i componenti essenziali (>180 kB consigliati)
- RAM:
 - Utilizzo di RAM statica: ~2 kB a seconda delle funzionalità utilizzate e dei tipi di Widget
 - stack: > 2kB (> 8 kB consigliati)
 - Dati dinamici (heap): >2 KB (>48 kB consigliati se si utilizzano molti widget GUI). Impostato da `LV_MEM_SIZE` in `lv_conf.h`.
 - Buffer di visualizzazione: > "Risoluzione orizzontale" pixel (>10 X "Risoluzione orizzontale")
 - Un frame buffer nell'MCU o in un controller video esterno
- Compilatore C99 o successivo
- Conoscenza base del C (o C++):
 - puntatori.
 - struct.
 - callback.

Nota

L'utilizzo della memoria può variare a seconda dell'architettura, del compilatore e delle opzioni di build.

1.1.2 Licenza

Il progetto LVGL (inclusi tutti i repository) è concesso con la [licenza MIT](#). Questo significa che lo si può utilizzare anche in progetti commerciali.

Non è obbligatorio, ma apprezzeremmo molto se si scrivesse qualche parola sul proprio progetto nella categoria [My projects](#) del forum o si inviasse un messaggio privato a [lvgl.io](#).

Sebbene LVGL sia disponibile gratuitamente, dietro c'è un enorme lavoro. È stato creato da un gruppo di volontari che lo hanno reso disponibile a tutti nel loro tempo libero.

Per rendere sostenibile il progetto LVGL, preghiamo di considerare la possibilità di [Contribuire](#). Si può scegliere tra diversi modi. Consultare [Contribuire](#), ad esempio scrivendo semplicemente un tweet su di sé usando LVGL, correggendo bug, traducendo la documentazione o addirittura diventando un manutentore.

1.1.3 FAQ

Dove posso porre domande?

Si possono porre domande nel forum: <https://forum.lvgl.io/>.

Utilizziamo [GitHub issues](#) per le discussioni relative allo sviluppo. Si dovrebbero utilizzarli solo se la domanda o il problema è strettamente correlato allo sviluppo della libreria.

Prima di postare una domanda, leggere questa sezione FAQ, poiché si potrebbe trovare la risposta al problema anche qui.

Questo MCU/hardware è supportato?

Tutti i MCU in grado di pilotare un display tramite porta parallela, SPI, interfaccia RGB o altro e che soddisfano i [Requisiti](#) sono supportati da LVGL.

Questo include:

- MCU "comuni" come STM32F, STM32H, NXP Kinetis, LPC, iMX, dsPIC33, PIC32, SWM341 ecc.
- Moduli Bluetooth, GSM e Wi-Fi come Nordic NRF, Espressif ESP32 e Raspberry Pi Pico W
- Linux con device frame buffer come /dev/fb0. Questo include computer Single-board come il Raspberry Pi
- Qualsiasi altro dispositivo con una MCU sufficientemente potente e una periferica per pilotare un display

Questo display è supportato?

LVGL richiede solo una semplice funzione driver per copiare un array di pixel in una determinata area del display. Se lo si può fare col proprio display, lo si può usare con LVGL.

Alcuni esempi dei tipi di display supportati:

- TFT con profondità di colore a 16 o 24 bit
- Monitor con porta HDMI
- Piccoli display monocromatici
- Display in scala di grigi
- anche matrici di LED
- o qualsiasi altro display in cui sia possibile controllare il colore/stato dei pixel

Consultare la sezione [Display \(lv_display\)](#) per ulteriori informazioni.

LVGL non si avvia, si blocca casualmente o non viene visualizzato nulla sul display. Quale potrebbe essere il problema?

- Provare ad aumentare [LV_MEM_SIZE](#).
- Assicurarsi che il display funzioni senza LVGL. Ad esempio, colorarlo in rosso all'avvio.
- Abilitare [Logging](#).
- Abilitare le asserzioni in `lv_conf.h` (`LV_USE_ASSERT_...`).
- Se si usa un RTOS:
 - Aumentare la dimensione dello stack del task che chiama `lv_timer_handler()`.
 - Utilizzare uno dei metodi per la gestione dei thread descritti in [Considerazioni sui Thread](#).

Il driver video non viene chiamato. Cos'è sfuggito?

Chiamare `lv_tick_inc(x)` come prescritto in [Interfaccia ai Tick](#) e chiamare `lv_timer_handler()` come stabilito in [Gestione del Timer](#).

Per saperne di più, consultare le sezioni [Interfaccia ai Tick](#) e [Gestione del Timer](#).

Perché il driver video viene chiamato una sola volta? Viene aggiornata solo la parte superiore del display.

Chiamare `lv_display_flush_ready(drv)` alla fine del proprio "display flush callback" come indicato nella sezione [Callback Flush](#).

Perché si vede solo spazzatura sullo schermo?

Probabilmente c'è un bug nel driver del display. Provare il seguente codice senza usare LVGL. Si dovrebbe vedere un quadrato con una sfumatura rosso-blu.

```
#define BUF_WIDTH 255
uint16_t buf[BUF_WIDTH];
uint32_t i;
for(i = 0; i < BUF_WIDTH; i++) {
    lv_color_t c = lv_color_mix(lv_color_hex(0xffff0000), lv_color_hex(0x00ffff), i);
    buf[i] = lv_color_to_u16(c);

    lv_area_t a;
    a.x1 = 5;
    a.x2 = a.x1 + BUF_WIDTH - 1;
    a.y1 = 10 + i;
    a.y2 = 10 + i;
    my_flush_cb(NULL, &a, (void*) buf);
}
```

Perché si vedono colori senza senso sullo schermo?

Il formato colore LVGL configurato, probabilmente, non è compatibile con il formato colore del display. Controllare `LV_COLOR_DEPTH` in `lv_conf.h`.

Come si può velocizzare l'interfaccia utente?

- Attivare l'ottimizzazione del compilatore e abilitare la memorizzazione nella cache di istruzioni e dati, se disponibile sull'MCU.
- Aumentare la dimensione del buffer di visualizzazione.
- Utilizzare due buffer di visualizzazione e svuotare il buffer con DMA (o una periferica simile) in background.
- Aumentare la velocità del clock della porta SPI o della parallela se si utilizzano per pilotare il display.
- Se il display ha una porta SPI, valutare la possibilità di passare a un modello con interfaccia parallela, poiché offre una velocità di elaborazione molto più elevata.
- Mantenere il buffer di visualizzazione nella RAM interna (non nella SRAM esterna) perché LVGL la utilizza molto e dovrebbe avere tempi di accesso rapidi.
- Si consiglia di ridurre al minimo il sovraccarico della CPU LVGL aggiornando i Widget:
 - solo una volta, subito prima di ogni aggiornamento dello schermo, e
 - solo quando ciò cambierà ciò che l'utente finale vedrà.

Come posso ridurre l'utilizzo di flash/ROM?

Si possono disabilitare le funzionalità inutilizzate (come animazioni, file system, GPU, ecc.) e i tipi di widget in `lv_conf.h`.

Se si utilizza GCC/CLANG, è possibile aggiungere i flag di compilazione `-fdata-sections -ffunction-sections` e il flag del linker `--gc-sections` per rimuovere funzioni e variabili non utilizzate dal binario finale. Se possibile, aggiungere il flag del compilatore `-flto` per abilitare l'ottimizzazione in fase di link insieme a `-Os` per GCC o `-Oz` per CLANG e le versioni più recenti di GCC.

Come ridurre l'utilizzo di RAM?

- Ridurre la dimensione del *Display buffer*.
- Ridurre *LV_MEM_SIZE* in *lv_conf.h*. Questa memoria viene utilizzata quando si creano widget come pulsanti, etichette, ecc.
- Per lavorare con *LV_MEM_SIZE* più bassi si possono creare Widget solo quando necessario ed eliminarli quando non lo sono più.

Come si utilizza LVGL con un sistema operativo?

Per lavorare con un sistema operativo in cui le attività possono interrompersi a vicenda (in modo preventivo), è necessario assicurarsi che nessuna chiamata di funzione LVGL venga chiamata mentre è in corso un'altra chiamata LVGL. Esistono diversi modi per farlo. Consultare la sezione *Considerazioni sui Thread* per saperne di più.

1.1.4 Il Repository LVGL

Layout del repository

Tutti i repository del progetto LVGL sono ospitati su [GitHub](#).

Questi repository si trovano nei seguenti posti:

- [lvgl](#): La libreria stessa con numerosi [esempi](#) e [demo](#).
- [blog](#): Sorgente del sito del blog
- [sim](#): Sorgente del sito del simulatore online
- [lv_port_*](#): Porting di LVGL per schede di sviluppo e altre piattaforme
- [lv_binding_*](#): Binding ad altri linguaggi

Policy di rilascio

I repository principali seguono le regole del [Semantic Versioning](#):

- Versione major: modifiche API incompatibili. Es. v5.0.0, v6.0.0
- Versione minor: funzionalità nuove ma retrocompatibili. Es. v6.1.0, v6.2.0
- Versione patch: correzioni di bug retrocompatibili. Es. v6.1.1, v6.1.2

Tag come vX.Y.Z vengono creati per ogni release.

Ciclo di rilascio

- Correzioni di bug: rilasciate su richiesta, anche settimanalmente
- Release minor: ogni 3-4 mesi
- Release major: circa una volta all'anno

Branch

I repository principali hanno almeno i seguenti branch:

- [master](#): versione più recente, le patch vengono unite direttamente qui
- [release/vX.Y](#): versioni stabili delle release minor

Log delle modifiche

Le modifiche sono loggatetengono scritte nei *Log delle Modifiche*.

Versioni supportate

Prima della v8, l'ultima release minor di ogni serie principale era supportata per 1 anno. A partire dalla v8, ogni release minor è supportata per 1 anno.

Versione	Data di rilascio	Fine del supporto	Attivo
v5.3	1 Feb, 2019	1 Feb, 2020	No
v6.1	26 Nov, 2019	26 Nov, 2020	No
v7.11	16 Mar, 2021	16 Mar, 2022	No
v8.0	1 Giu, 2021	1 Giu, 2022	No
v8.1	10 Nov, 2021	10 Nov, 2022	No
v8.2	31 Gen, 2022	31 Gen, 2023	No
v8.3	6 Lug, 2022	1 Gen, 2025	No
v8.4	19 Mar, 2024	20 Mar, 2025	No
v9.0	22 Gen, 2024	22 Gen, 2025	No
v9.1	20 Mar, 2024	20 Mar, 2025	No
v9.2	26 Ago, 2024	26 Ago, 2025	Sì
v9.3	3 giugno 2025	3 giugno 2026	Sì

CAPITOLO 2

Per iniziare

2.1 Imparare le Basi

2.1.1 Panoramica

LVGL (Light and Versatile Graphics Library) è una libreria grafica gratuita e open source che fornisce tutto il necessario per creare un'interfaccia utente grafica embedded con elementi grafici facili da usare, simili a quelli dei telefoni cellulari, splendidi effetti visivi e un ridotto utilizzo della memoria.

LVGL può essere considerato come una raccolta di file C e H inseribili in qualsiasi progetto per aggiungere funzionalità di interfaccia utente al prodotto.

Con l'aiuto di funzioni API coerenti e facili da apprendere, è possibile creare widget (pulsanti, slider, grafici, ecc.), personalizzarli, aggiungere eventi, layout o animazioni.

In base a queste impostazioni, LVGL creerà un'immagine (utilizzando il suo motore di rendering software nativo o una GPU) e chiamerà una funzione di callback per visualizzare l'immagine renderizzata sul display. Questa funzione di callback è l'interfaccia principale tra LVGL e il display. La maggior parte del lavoro relativo al porting si concentra sulla scrittura di tale callback in modo efficace.

Questo capitolo illustrerà le basi per dare un'idea del funzionamento di LVGL e di come può essere utilizzato. Per maggiori dettagli su ciascuna funzionalità, visitare la pagina di documentazione dedicata.

Panoramica sull'Integrazione di LVGL

Di seguito è riportata una panoramica su come integrare LVGL nel proprio progetto. I dettagli completi sono disponibili in [Panoramica](#).

Inizializzazione del Driver

È responsabilità dell'utente configurare il clock, i timer, le periferiche, ecc.

Chiamare lv_init()

Inizializzare LVGL stesso.

Creare dispositivi di visualizzazione e di input

Creare display (`lv_display_t`) e dispositivi di input (`lv_indev_t`) e impostarne le callback.

Creare l'Interfaccia Utente

Chiamare le funzioni LVGL per creare schermate, widget, stili, animazioni, eventi, ecc.

Chiamare lv_timer_handler() in un ciclo

Questo comando gestisce tutte le attività relative a LVGL:

- aggiornare i(l) display,
- leggere i dispositivi di input,
- generare eventi in base all'input dell'utente (e altre cose),
- eseguire eventuali animazioni ed
- eseguire i timer creati dall'utente.

Esempio

Questo è solo un breve esempio di come aggiungere LVGL a un nuovo progetto. Per maggiori dettagli consultare [Integration Overview](#).

```
void main(void)
{
    your_driver_init();

    lv_init();

    lv_tick_set_cb(my_get_millis);

    lv_display_t * display = lv_display_create(320, 240);

    /* LVGL will render to this 1/10 screen sized buffer for 2 bytes/pixel */
    static uint8_t buf[320 * 240 / 10 * 2];
    lv_display_set_buffers(display, buf, NULL, LV_DISPLAY_RENDER_MODE_PARTIAL);

    /* This callback will display the rendered image */
    lv_display_set_flush_cb(display, my_flush_cb);

    /* Create widgets */
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Hello LVGL!");

    /* Make LVGL periodically execute its tasks */
    while(1) {
        /* Provide updates to currently-displayed Widgets here. */
        lv_timer_handler();
        my_sleep(5); /*Wait 5 milliseconds before processing LVGL timer again*/
    }
}

/* Return the elapsed milliseconds since startup.
 * It needs to be implemented by the user */
uint32_t my_get_millis(void)
{
    return my_tick_ms;
}

/* Copy rendered image to screen.
 * This needs to be implemented by the user. */
void my_flush_cb(lv_display_t * disp, const lv_area_t * area, uint8_t * px_buf)
{
    /* Show the rendered image on the display */
    my_display_update(area, px_buf);

    /* Indicate that the buffer is available.
     * If DMA were used, call in the DMA complete interrupt. */
    lv_display_flush_ready();
}
```

2.1.2 Display

Display si riferisce all'hardware effettivo. Per collegare LVGL all'hardware, è necessario creare e inizializzare un oggetto `lv_display_t`.

LVGL supporta molti driver integrati (vedere [Integrazione di LVGL](#)), ma è anche facile inizializzare un display da zero (come mostrato sopra).

LVGL gestisce anche più display contemporaneamente.

2.1.3 Screen

Uno *Screen* è un widget LVGL creato su un *Display*. È un contenitore logico per altri widget. Un display può avere più screen, ma ce n'è sempre uno solo attivo, che può essere recuperato tramite `lv_screen_active()`. Restituisce un puntatore `lv_obj_t *`. Consultare [Lo Screen Attivo](#) per ulteriori informazioni.

Il modo più comune per creare uno screen è creare un [Widget di Base](#) con un genitore `NULL`. Ad esempio,

```
lv_obj_t * my_screen = lv_obj_create(NULL);
```

Uno screen può essere caricato in questo modo: `lv_screen_load(my_screen)`

2.1.4 | Widget

I widget sono gli elementi costitutivi fondamentali dell'interfaccia utente. Per esempio: *Pulsante* (`lv_button`), *Slider* (`lv_slider`), *Drop-Down List* (`lv_dropdown`), *Grafici* (`lv_chart`), ecc.

I widget possono essere creati dinamicamente chiamando le rispettive funzioni *create*. La funzione *create* restituisce un puntatore `lv_obj_t *` utilizzabile per configurare il widget in un secondo momento.

Ogni funzione *create* ha un singolo argomento *parent* che definisce a quale widget verrà aggiunto quello nuovo.

Ad esempio:

```
lv_obj_t * my_button1 = lv_button_create(lv_screen_active());
lv_obj_t * my_label1 = lv_label_create(my_button1);
```

Se un widget o una schermata non sono più necessari, possono essere rimossi chiamando `lv_obj_delete(my_button1)`

Per modificare le proprietà dei widget, è possibile utilizzare due set di funzioni:

- Funzioni `lv_obj_...()` per proprietà comuni, ad esempio `lv_obj_set_width()`, `lv_obj_add_style()`, ecc. Queste sono trattate in [Funzionalità Comuni dei Widget](#).
- Funzioni `lv_<widget_type>_...()` per proprietà specifiche del tipo, ad esempio `lv_label_set_text()`, `lv_slider_set_value()`, ecc.

Ecco un esempio che mostra anche alcune unità di misura *non-pixel* per le dimensioni:

```
lv_obj_t * my_button1 = lv_button_create(lv_screen_active());
/* Set parent-sized width, and content-sized height */
lv_obj_set_size(my_button1, lv_pct(100), LV_SIZE_CONTENT);
/* Align to the right center with 20px offset horizontally */
lv_obj_align(my_button1, LV_ALIGN_RIGHT_MID, -20, 0);

lv_obj_t * my_label1 = lv_label_create(my_button1);
lv_label_set_text_fmt(my_label1, "Click me!");
lv_obj_set_style_text_color(my_label1, lv_color_hex(0xff0000), 0);
/* Make the text red */
```

Per visualizzare l'API completa di un widget, consultare la relativa documentazione in [Tutti i Widget](#) o controllare il relativo file header nel codice sorgente.

2.1.5 | Eventi

Gli eventi vengono utilizzati per informare l'applicazione che si è verificato qualcosa con un widget. È possibile assegnare una o più callback a un Widget, che verranno richiamate quando il Widget verrà cliccato, rilasciato, trascinato, eliminato, ecc.

Una callback viene assegnata in questo modo:

```
lv_obj_add_event_cb(btn, my_btn_event_cb, LV_EVENT_CLICKED, NULL);
...
void my_btn_event_cb(lv_event_t * e)
{
    printf("Clicked\n");
}
```

È possibile utilizzare `LV_EVENT_ALL` al posto di `LV_EVENT_CLICKED` per richiamare la callback per tutti gli eventi.

Le callback degli eventi ricevono l'argomento `lv_event_t *` e contenente il codice dell'evento corrente e altre informazioni relative all'evento. Il codice dell'evento corrente può essere recuperato con:

```
lv_event_code_t code = lv_event_get_code(e);
```

Il Widget che ha attivato l'evento può essere recuperato con:

```
lv_obj_t * widget = lv_event_get_target_obj(e);
```

Si può scoprire tutto sugli eventi nella sezione [Eventi](#).

2.1.6 Parti e Stati

Parti

I Widget sono costituiti da una o più *parti*. Ad esempio, un pulsante ha una sola *parte* chiamata `LV_PART_MAIN`. Tuttavia, uno *Slider* (`lv_slider`) ha `LV_PART_MAIN`, `LV_PART_INDICATOR` e `LV_PART_KNOB`.

Utilizzando le *parti* è possibile applicare diversi *stili* alle parti di un widget.

Consultare la documentazione del Widget per scoprire quali parti utilizza.

Stati

I Widget possono trovarsi in una combinazione dei seguenti stati:

- `LV_STATE_DEFAULT`: Normale, stato "a riposo"
- `LV_STATE_CHECKED`: Stato attivato o selezionato
- `LV_STATE_FOCUSED`: Focus tramite tastiera, encoder o clic tramite touchpad/mouse
- `LV_STATE_FOCUS_KEY`: Focus tramite tastiera o encoder ma non tramite touchpad/mouse
- `LV_STATE_EDITED`: Modificato tramite encoder
- `LV_STATE_HOVERED`: Passaggio del mouse
- `LV_STATE_PRESSED`: Premuto
- `LV_STATE_SCROLLED`: In corso di scrolling
- `LV_STATE_DISABLED`: Disabilitato

Ad esempio, se si preme un Widget, questo passerà automaticamente agli stati `LV_STATE_FOCUSED` e `LV_STATE_PRESSED`. Quando lo si rilascia, lo stato `LV_STATE_PRESSED` verrà rimosso, mentre lo stato `LV_STATE_FOCUSED` rimarrà attivo.

Per verificare se un Widget si trova in un determinato stato, utilizzare `lv_obj_has_state(widget, LV_STATE_...)`. Restituirà `true` se il widget si trova attualmente in quello stato.

Per aggiungere o rimuovere stati a livello di codice, utilizzare:

```
lv_obj_add_state(widget, LV_STATE_...);
lv_obj_remove_state(widget, LV_STATE_...);
```

2.1.7 Stili

Inizializzazione degli stili

Gli stili sono contenuti in oggetti `lv_style_t`. Contengono proprietà come colore di sfondo, spessore del bordo, font, ecc.

Gli stili possono essere aggiunti alla *Parte* e allo *Stato*. Solo il loro puntatore viene salvato nei Widget, quindi devono essere definiti come variabili statiche o globali.

Prima di utilizzare uno stile è necessario inizializzarlo con `lv_style_init(&style1)`. In seguito, è possibile aggiungere proprietà per configurare lo stile. Ad esempio:

```
static lv_style_t style1;
lv_style_init(&style1);
lv_style_set_bg_color(&style1, lv_color_hex(0xa03080));
lv_style_set_border_width(&style1, 2);
```

Vedere *Panoramica sugli Stili* per maggiori dettagli.

Proprietà di Stile per vedere l'elenco completo.

Aggiungere stili ai widget

Successivamente è possibile aggiungere ai widget:

```
lv_obj_add_style(my_button1, &style1, 0); /*0 means add to the main part and default state*/
lv_obj_add_style(my_checkbox1, &style1, LV_STATE_DISABLED); /*Add to checkbox's disabled state*/
lv_obj_add_style(my_slider1, &style1, LV_PART_KNOB | LV_STATE_PRESSED); /*Add to the slider's knob pressed state*/
```

Ereditarietà

Alcune proprietà (in particolare quelle relative al testo) possono essere ereditate. Ciò significa che, se una proprietà non è impostata in un Widget, verrà cercata nei Widget genitore. Ad esempio, è possibile impostare il font una volta nello stile della schermata e tutto il testo su quella schermata lo erediterà per default, a meno che il font non sia specificato nel widget o in uno dei suoi genitori.

Stili Locali

È anche possibile aggiungere proprietà di stile locali ai Widget. Questo crea uno stile che risiede all'interno del Widget e viene utilizzato solo da quel Widget:

```
lv_obj_set_style_bg_color(slider1, lv_color_hex(0x2080bb), LV_PART_INDICATOR | LV_STATE_PRESSED);
```

Per tutti i dettagli vedere [Stili](#).

2.1.8 Soggetti e Osservatori

Soggetti e Osservatori sono strumenti potenti per creare facilmente data binding.

I soggetti sono variabili globali `lv_subject_t` che memorizzano valori interi, colori, stringhe, ecc.

L'interfaccia utente o l'applicazione possono sottoscrivere questi soggetti creando *callback dell'osservatore* che vengono notificati quando il soggetto cambia.

Anche un widget può sottoscrivere un soggetto. In questo modo, quando il widget viene eliminato, la sua sottoscrizione verrà automaticamente annullata.

Per alcuni widget, le funzioni di supporto semplificano la connessione ai soggetti. Ad esempio: `lv_slider_bind_value()`, `lv_label_bind_text()`.

In generale, l'utilizzo di *soggetti e osservatori* è un modo per collegare varie parti dell'interfaccia utente e farle reagire dinamicamente alle modifiche dei dati dell'applicazione, oppure consentire all'applicazione di reagire alle modifiche dell'interfaccia utente.

```
static void label_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    lv_obj_t * label = lv_observer_get_target_obj(observer);
    lv_label_set_text_fmt(label, "Progress: %d", lv_subject_get_int(subject));
}

...

static lv_subject_t subject1;
lv_subject_init_int(&subject1, 10);

lv_obj_t * label1 = lv_label_create(lv_screen_active());
/*lv_label_bind_text could have been used too*/
lv_subject_add_observer_obj(&subject1, label_observer_cb, label1, NULL);

lv_obj_t * slider1 = lv_slider_create(lv_screen_active());
lv_slider_bind_value(slider1, &subject1);
lv_obj_set_y(slider1, 30);

lv_subject_set_int(&subject1, 30);
```

Ulteriori approfondimenti nella pagina di documentazione degli [Osservatori](#).

2.2 Esempi di Base

Di seguito sono riportati diversi esempi di base. Includono il codice applicativo che genera il "Widget Tree" [gerarchia dei widget] necessario per far sì che LVGL esegua il rendering degli esempi mostrati. Ogni esempio presuppone che LVGL abbia subito una normale inizializzazione, ovvero che sia stato creato un oggetto `lv_display_t` e che quindi abbia [Lo Screen Attivo](#).

2.2.1 Un'etichetta *hello world* molto semplice

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

/***
 * Basic example to create a "Hello world" label
 */
void lv_example_get_started_1(void)
{
    /*Change the active screen's background color/
    lv_obj_set_style_bg_color(lv_screen_active(), lv_color_hex(0x003a57), LV_PART_MAIN);

    /*Create a white label, set its text and align it to the center/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Hello world");
    lv_obj_set_style_text_color(lv_screen_active(), lv_color_hex(0xffffffff), LV_PART_MAIN);
    lv_obj_align(label, LV_ALIGN_CENTER, 0, 0);
}

#endif
```

2.2.2 Un pulsante con un'etichetta e un evento di reazione al clic

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_BUTTON

static void btn_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * btn = lv_event_get_target_obj(e);
    if(code == LV_EVENT_CLICKED) {
        static uint8_t cnt = 0;
        cnt++;

        /*Get the first child of the button which is the label and change its text*/
        lv_obj_t * label = lv_obj_get_child(btn, 0);
        lv_label_set_text_fmt(label, "Button: %d", cnt);
    }
}

/***
 * Create a button with a label and react on click event.
 */
void lv_example_get_started_2(void)
{
    lv_obj_t * btn = lv_button_create(lv_screen_active()); /*Add a button to the current screen*/
    lv_obj_set_pos(btn, 10, 10); /*Set its position*/
    lv_obj_set_size(btn, 120, 50); /*Set its size*/
    lv_obj_add_event_cb(btn, btn_event_cb, LV_EVENT_ALL, NULL); /*Assign a callback to the button*/

    lv_obj_t * label = lv_label_create(btn); /*Add a label to the button*/
    lv_label_set_text(label, "Button"); /*Set the label's text*/
    lv_obj_center(label);
}

#endif
```

2.2.3 Creare stili da zero per i pulsanti

```
#include "../lv_examples.h"
#if LV_USE_BUTTON && LV_BUILD_EXAMPLES

static lv_style_t style_btn;
static lv_style_t style_button_pressed;
static lv_style_t style_button_red;

static lv_color_t darken(const lv_color_filter_dsc_t * dsc, lv_color_t color, lv_opa_t opa)
{
    LV_UNUSED(dsc);
    return lv_color_darken(color, opa);
}

static void style_init(void)
{
    /*Create a simple button style*/
    lv_style_init(&style_btn);
    lv_style_set_radius(&style_btn, 10);
    lv_style_set_bg_opa(&style_btn, LV_OPA_COVER);
    lv_style_set_bg_color(&style_btn, lv_palette_lighten(LV_PALETTE_GREY, 3));
    lv_style_set_bg_grad_color(&style_btn, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_bg_grad_dir(&style_btn, LV_GRAD_DIR_VER);

    lv_style_set_border_color(&style_btn, lv_color_black());
    lv_style_set_border_opa(&style_btn, LV_OPA_20);
    lv_style_set_border_width(&style_btn, 2);

    lv_style_set_text_color(&style_btn, lv_color_black());

    /*Create a style for the pressed state.
     *Use a color filter to simply modify all colors in this state*/
    static lv_color_filter_dsc_t color_filter;
    lv_color_filter_dsc_init(&color_filter, darken);
    lv_style_init(&style_button_pressed);
    lv_style_set_color_filter_dsc(&style_button_pressed, &color_filter);
    lv_style_set_color_opa(&style_button_pressed, LV_OPA_20);

    /*Create a red style. Change only some colors.*/
    lv_style_init(&style_button_red);
    lv_style_set_bg_color(&style_button_red, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_bg_grad_color(&style_button_red, lv_palette_lighten(LV_PALETTE_RED, 3));
}

/***
```

(continues on next page)

(continua dalla pagina precedente)

```

/* Create styles from scratch for buttons.
*/
void lv_example_get_started_3(void)
{
    /*Initialize the style*/
    style_init();

    /*Create a button and use the new styles*/
    lv_obj_t * btn = lv_button_create(lv_screen_active());
    /* Remove the styles coming from the theme
     * Note that size and position are also stored as style properties
     * so lv_obj_remove_style_all(btn) will remove the set size and position too */
    lv_obj_remove_style_all(btn);
    lv_obj_set_pos(btn, 10, 10);
    lv_obj_set_size(btn, 120, 50);
    lv_obj_add_style(btn, &style_btn, 0);
    lv_obj_add_style(btn, &style_button_pressed, LV_STATE_PRESSED);

    /*Add a label to the button*/
    lv_obj_t * label = lv_label_create(btn);
    lv_label_set_text(label, "Button");
    lv_obj_center(label);

    /*Create another button and use the red style too*/
    lv_obj_t * btn2 = lv_button_create(lv_screen_active());
    lv_obj_remove_style_all(btn2);                                /*Remove the styles coming from the theme*/
    lv_obj_set_pos(btn2, 10, 80);
    lv_obj_set_size(btn2, 120, 50);
    lv_obj_add_style(btn2, &style_btn, 0);
    lv_obj_add_style(btn2, &style_button_red, 0);
    lv_obj_add_style(btn2, &style_button_pressed, LV_STATE_PRESSED);
    lv_obj_set_style_radius(btn2, LV_RADIUS_CIRCLE, 0); /*Add a local style too*/

    label = lv_label_create(btn2);
    lv_label_set_text(label, "Button 2");
    lv_obj_center(label);
}

#endif

```

2.2.4 Creare uno slider e scriverne il valore su un'etichetta

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SLIDER

static lv_obj_t * label;

static void slider_event_cb(lv_event_t * e)
{
    lv_obj_t * slider = lv_event_get_target_obj(e);

    /*Refresh the text*/
    lv_label_set_text_fmt(label, "%" LV_PRIId32, lv_slider_get_value(slider));
    lv_obj_align_to(label, slider, LV_ALIGN_OUT_TOP_MID, 0, -15); /*Align top of the slider*/
}

/***
 * Create a slider and write its value on a label.
 */
void lv_example_get_started_4(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_obj_set_width(slider, 200);           /*Set the width*/
    lv_obj_center(slider);                 /*Align to the center of the parent (screen)*/
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL); /*Assign an event function*/

    /*Create a label above the slider*/
    label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "0");
    lv_obj_align_to(label, slider, LV_ALIGN_OUT_TOP_MID, 0, -15); /*Align top of the slider*/
}

#endif

```

2.3 Prossimi Passi

Esistono diversi ottimi modi per approfondire la conoscenza di LVGL. Ecco un ordine consigliato di documenti da leggere e di spunti su cui esercitarsi nell'approfondire le proprie conoscenze:

1. Se non è stato già fatto, iniziare con la pagina [Introduzione](#) della documentazione. (5 minuti)
2. Consultate le [Online Demos](#) per vedere LVGL in azione. (3 minuti)
3. Se non è stato già fatto, leggere [Imparare le Basi](#) (sopra). (15 minuti)
4. Configurare un [Esecuzione su un PC](#) LVGL. (10 minuti)
5. Consultare alcuni [Esempi](#) e il relativo codice.
6. Consultare [Panoramica dell'Integrazione](#) o vedere nei [ready-to-use Projects](#).
7. Leggere le pagine di [Moduli Principali](#) per comprendere meglio la libreria. (2-3 ore)

8. Consultare la documentazione di [Tutti i Widget](#) per vedere cosa è disponibile.
9. Se ci sono domande, visitare il [Forum](#).
10. Leggere la guida per scoprire come [Contribuire](#) a migliorare LVGL. (15 minuti)

CAPITOLO 3

Esempi

3.1 Per Iniziare

3.1.1 Un'etichetta *hello world* molto semplice

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

/***
 * Basic example to create a "Hello world" label
 */
void lv_example_get_started_1(void)
{
    /*Change the active screen's background color*/
    lv_obj_set_style_bg_color(lv_screen_active(), lv_color_hex(0x003a57), LV_PART_MAIN);

    /*Create a white label, set its text and align it to the center*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Hello world");
    lv_obj_set_style_text_color(lv_screen_active(), lv_color_hex(0xffffffff), LV_PART_MAIN);
    lv_obj_align(label, LV_ALIGN_CENTER, 0, 0);
}

#endif
```

3.1.2 Un pulsante con un'etichetta e un evento di reazione al clic

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_BUTTON

static void btn_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * btn = lv_event_get_target_obj(e);
    if(code == LV_EVENT_CLICKED) {
        static uint8_t cnt = 0;
        cnt++;

        /*Get the first child of the button which is the label and change its text*/
        lv_obj_t * label = lv_obj_get_child(btn, 0);
        lv_label_set_text_fmt(label, "Button: %d", cnt);
    }
}

/***
 * Create a button with a label and react on click event.
 */
void lv_example_get_started_2(void)
{
    lv_obj_t * btn = lv_button_create(lv_screen_active()); /*Add a button to the current screen*/
    lv_obj_set_pos(btn, 10, 10); /*Set its position*/
    lv_obj_set_size(btn, 120, 50); /*Set its size*/
    lv_obj_add_event_cb(btn, btn_event_cb, LV_EVENT_ALL, NULL); /*Assign a callback to the button*/

    lv_obj_t * label = lv_label_create(btn); /*Add a label to the button*/
    lv_label_set_text(label, "Button"); /*Set the label's text*/
    lv_obj_center(label);
}

#endif
```

3.1.3 Creare stili da zero per i pulsanti

```
#include "../lv_examples.h"
#if LV_USE_BUTTON && LV_BUILD_EXAMPLES

static lv_style_t style_btn;
static lv_style_t style_button_pressed;
static lv_style_t style_button_red;

static lv_color_t darken(const lv_color_filter_dsc_t * dsc, lv_color_t color, lv_opa_t opa)
{
    LV_UNUSED(dsc);
    return lv_color_darken(color, opa);
}

static void style_init(void)
{
    /*Create a simple button style*/
    lv_style_init(&style_btn);
    lv_style_set_radius(&style_btn, 10);
    lv_style_set_bg_opa(&style_btn, LV_OPA_COVER);
    lv_style_set_bg_color(&style_btn, lv_palette_lighten(LV_PALETTE_GREY, 3));
    lv_style_set_bg_grad_color(&style_btn, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_bg_grad_dir(&style_btn, LV_GRAD_DIR_VER);

    lv_style_set_border_color(&style_btn, lv_color_black());
    lv_style_set_border_opa(&style_btn, LV_OPA_20);
    lv_style_set_border_width(&style_btn, 2);

    lv_style_set_text_color(&style_btn, lv_color_black());

    /*Create a style for the pressed state.
     *Use a color filter to simply modify all colors in this state*/
    static lv_color_filter_dsc_t color_filter;
    lv_color_filter_dsc_init(&color_filter, darken);
    lv_style_init(&style_button_pressed);
    lv_style_set_color_filter_dsc(&style_button_pressed, &color_filter);
    lv_style_set_color_filter_opa(&style_button_pressed, LV_OPA_20);

    /*Create a red style. Change only some colors.*/
    lv_style_init(&style_button_red);
    lv_style_set_bg_color(&style_button_red, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_bg_grad_color(&style_button_red, lv_palette_lighten(LV_PALETTE_RED, 3));
}

/***
 * Create styles from scratch for buttons.
 */
void lv_example_get_started_3(void)
{
    /*Initialize the style*/
    style_init();

    /*Create a button and use the new styles*/
    lv_obj_t * btn = lv_button_create(lv_screen_active());
    /* Remove the styles coming from the theme
     * Note that size and position are also stored as style properties
     * so lv_obj_remove_style_all will remove the set size and position too */
    lv_obj_remove_style_all(btn);
    lv_obj_set_pos(btn, 10, 10);
    lv_obj_set_size(btn, 120, 50);
    lv_obj_add_style(btn, &style_btn, 0);
    lv_obj_add_style(btn, &style_button_pressed, LV_STATE_PRESSED);

    /*Add a label to the button*/
    lv_obj_t * label = lv_label_create(btn);
    lv_label_set_text(label, "Button");
    lv_obj_center(label);

    /*Create another button and use the red style too*/
    lv_obj_t * btn2 = lv_button_create(lv_screen_active());
    lv_obj_remove_style_all(btn2); /*Remove the styles coming from the theme*/
    lv_obj_set_pos(btn2, 10, 80);
    lv_obj_set_size(btn2, 120, 50);
    lv_obj_add_style(btn2, &style_btn, 0);
    lv_obj_add_style(btn2, &style_button_red, 0);
    lv_obj_add_style(btn2, &style_button_pressed, LV_STATE_PRESSED);
    lv_obj_set_style_radius(btn2, LV_RADIUS_CIRCLE, 0); /*Add a local style too*/

    label = lv_label_create(btn2);
    lv_label_set_text(label, "Button 2");
    lv_obj_center(label);
}

#endif
```

3.1.4 Creare uno slider e scriverne il valore su un'etichetta

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SLIDER

static lv_obj_t * label;

static void slider_event_cb(lv_event_t * e)
{
    lv_obj_t * slider = lv_event_get_target_obj(e);

    /*Refresh the text*/
    lv_label_set_text_fmt(label, "%" LV_PRId32, lv_slider_get_value(slider));
    lv_obj_align_to(label, slider, LV_ALIGN_OUT_TOP_MID, 0, -15); /*Align top of the slider*/
}

/***
 * Create a slider and write its value on a label.
 */
void lv_example_get_started_4(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_width(slider, 200);           /*Set the width*/
lv_obj_center(slider);                 /*Align to the center of the parent (screen)*/
lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL); /*Assign an event function*/

/*Create a label above the slider*/
label = lv_label_create(lv_screen_active());
lv_label_set_text(label, "0");
lv_obj_align_to(label, slider, LV_ALIGN_OUT_TOP_MID, 0, -15); /*Align top of the slider*/
}

#endif

```

3.1.5 Semplice Gradiente Orizzontale

3.1.6 Gradiente (Obliquo) Lineare

3.1.7 Gradiente Radiale

3.1.8 Gradiente Conico

3.2 Stili

3.2.1 Stili per le Dimensioni

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Using the Size, Position and Padding style properties
 */
void lv_example_style_1(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_radius(&style, 5);

    /*Make a gradient*/
    lv_style_set_width(&style, 150);
    lv_style_set_height(&style, LV_SIZE_CONTENT);

    lv_style_set_pad_ver(&style, 20);
    lv_style_set_pad_left(&style, 5);

    lv_style_set_x(&style, lv_pct(50));
    lv_style_set_y(&style, 80);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label, "Hello");
}
#endif

```

3.2.2 Stili per il Background

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the background style properties
 */
void lv_example_style_2(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_radius(&style, 5);

    /*Make a gradient*/
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    static lv_grad_dsc_t grad;
    grad.dir = LV_GRAD_DIR_VER;
    grad.stops_count = 2;
    grad.stops[0].color = lv_palette_lighten(LV_PALETTE_GREY, 1);
    grad.stops[0].opa = LV_OPA_COVER;
    grad.stops[1].color = lv_palette_main(LV_PALETTE_BLUE);
    grad.stops[1].opa = LV_OPA_COVER;

    /*Shift the gradient to the bottom*/
    grad.stops[0].frac = 128;
    grad.stops[1].frac = 192;

    lv_style_set_bg_grad(&style, &grad);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}
#endif

```

3.2.3 Stili per i Bordi

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the border style properties
 */
void lv_example_style_3(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 10);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add border to the bottom+right*/
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_border_width(&style, 5);
    lv_style_set_border_opa(&style, LV_OPA_50);
    lv_style_set_border_side(&style, (lv_border_side_t)(LV_BORDER_SIDE_BOTTOM | LV_BORDER_SIDE_RIGHT));

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

3.2.4 Stili per i Contorni

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the outline style properties
 */
void lv_example_style_4(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add outline*/
    lv_style_set_outline_width(&style, 2);
    lv_style_set_outline_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_outline_pad(&style, 8);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

3.2.5 Stili per le Ombre

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the Shadow style properties
 */
void lv_example_style_5(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add a shadow*/
    lv_style_set_shadow_width(&style, 55);
    lv_style_set_shadow_color(&style, lv_palette_main(LV_PALETTE_BLUE));

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

3.2.6 Stili per le Immagini

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Using the Image style properties
 */
```

(continues on next page)

(continua dalla pagina precedente)

```

/*
void lv_example_style_6(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 3));
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));

    lv_style_set_image_recolor(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_image_recolor_opa(&style, LV_OPA_50);
    lv_style_set_transform_rotation(&style, 300);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_image_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);

    LV_IMAGE_DECLARE(img_cogwheel_argb);
    lv_image_set_src(obj, &img_cogwheel_argb);

    lv_obj_center(obj);
}

#endif

```

3.2.7 Stili per gli Archi

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_ARC

/***
* Using the Arc style properties
*/
void lv_example_style_7(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_arc_color(&style, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_arc_width(&style, 4);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_arc_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}
#endif

```

3.2.8 Stili per il Testo

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

/***
* Using the text style properties
*/
void lv_example_style_8(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 2));
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_pad_all(&style, 10);

    lv_style_set_text_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_text_letter_space(&style, 5);
    lv_style_set_text_line_space(&style, 20);
    lv_style_set_text_decor(&style, LV_TEXT_DECOR_UNDERLINE);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_label_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_label_set_text(obj, "Text of\n"
                        "a label");

    lv_obj_center(obj);
}

#endif

```

3.2.9 Stili per le Linee

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LINE

/***
* Using the line style properties
*/
void lv_example_style_9(void)
{
    static lv_style_t style;

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_init(&style);

lv_style_set_line_color(&style, lv_palette_main(LV_PALETTE_GREY));
lv_style_set_line_width(&style, 6);
lv_style_set_line_rounded(&style, true);

/*Create an object with the new style/
lv_obj_t * obj = lv_line_create(lv_screen_active());
lv_obj_add_style(obj, &style, 0);

static lv_point_precise_t p[] = {{10, 30}, {30, 50}, {100, 0}};
lv_line_set_points(obj, p, 3);

lv_obj_center(obj);
}

#endif

```

3.2.10 Transizione

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Creating a transition
 */
void lv_example_style_10(void)
{
    static const lv_style_prop_t props[] = {LV_STYLE_BG_COLOR, LV_STYLE_BORDER_COLOR, LV_STYLE_BORDER_WIDTH, 0};

    /* A default transition
     * Make it fast (100ms) and start with some delay (200 ms)*/
    static lv_style_transition_dsc_t trans_def;
    lv_style_transition_dsc_init(&trans_def, props, lv_anim_path_linear, 100, 200, NULL);

    /* A special transition when going to pressed state
     * Make it slow (500 ms) but start without delay*/
    static lv_style_transition_dsc_t trans_pr;
    lv_style_transition_dsc_init(&trans_pr, props, lv_anim_path_linear, 500, 0, NULL);

    static lv_style_t style_def;
    lv_style_init(&style_def);
    lv_style_set_transition(&style_def, &trans_def);

    static lv_style_t style_pr;
    lv_style_init(&style_pr);
    lv_style_set_bg_color(&style_pr, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_border_width(&style_pr, 6);
    lv_style_set_border_color(&style_pr, lv_palette_darken(LV_PALETTE_RED, 3));
    lv_style_set_transition(&style_pr, &trans_pr);

    /*Create an object with the new style_pr*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style_def, 0);
    lv_obj_add_style(obj, &style_pr, LV_STATE_PRESSED);

    lv_obj_center(obj);
}
#endif

```

3.2.11 Utilizzo di stili multipli

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Using multiple styles
 */
void lv_example_style_11(void)
{
    /*A base style*/
    static lv_style_t style_base;
    lv_style_init(&style_base);
    lv_style_set_bg_color(&style_base, lv_palette_main(LV_PALETTE_LIGHT_BLUE));
    lv_style_set_border_color(&style_base, lv_palette_darken(LV_PALETTE_LIGHT_BLUE, 3));
    lv_style_set_border_width(&style_base, 2);
    lv_style_set_radius(&style_base, 10);
    lv_style_set_shadow_width(&style_base, 10);
    lv_style_set_shadow_offset_y(&style_base, 5);
    lv_style_set_shadow_opa(&style_base, LV_OPA_50);
    lv_style_set_text_color(&style_base, lv_color_white());
    lv_style_set_width(&style_base, 100);
    lv_style_set_height(&style_base, LV_SIZE_CONTENT);

    /*Set only the properties that should be different*/
    static lv_style_t style_warning;
    lv_style_init(&style_warning);
    lv_style_set_bg_color(&style_warning, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_border_color(&style_warning, lv_palette_darken(LV_PALETTE_YELLOW, 3));
    lv_style_set_text_color(&style_warning, lv_palette_darken(LV_PALETTE_YELLOW, 4));

    /*Create an object with the base style only*/
    lv_obj_t * obj_base = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj_base, &style_base, 0);
    lv_obj_align(obj_base, LV_ALIGN_LEFT_MID, 20, 0);

    lv_obj_t * label = lv_label_create(obj_base);
    lv_label_set_text(label, "Base");
    lv_obj_center(label);

    /*Create another object with the base style and earnings style too*/
    lv_obj_t * obj_warning = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj_warning, &style_base, 0);

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_add_style(obj_warning, &style_warning, 0);
lv_obj_align(obj_warning, LV_ALIGN_RIGHT_MID, -20, 0);

label = lv_label_create(obj_warning);
lv_label_set_text(label, "Warning");
lv_obj_center(label);
}

#endif

```

3.2.12 Stili Locali

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Local styles
 */
void lv_example_style_12(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_GREEN));
    lv_style_set_border_color(&style, lv_palette_lighten(LV_PALETTE_GREEN, 3));
    lv_style_set_border_width(&style, 3);

    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);

    /*Overwrite the background color locally*/
    lv_obj_set_style_bg_color(obj, lv_palette_main(LV_PALETTE_ORANGE), LV_PART_MAIN);

    lv_obj_center(obj);
}
#endif

```

3.2.13 Aggiunta di stili a parti e a stati

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Add styles to parts and states
 */
void lv_example_style_13(void)
{
    static lv_style_t style_indic;
    lv_style_init(&style_indic);
    lv_style_set_bg_color(&style_indic, lv_palette_lighten(LV_PALETTE_RED, 3));
    lv_style_set_bg_grad_color(&style_indic, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_bg_grad_dir(&style_indic, LV_GRAD_DIR_HOR);

    static lv_style_t style_indic_pr;
    lv_style_init(&style_indic_pr);
    lv_style_set_shadow_color(&style_indic_pr, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_shadow_width(&style_indic_pr, 10);
    lv_style_set_shadow_spread(&style_indic_pr, 3);

    /*Create an object with the new style_pr*/
    lv_obj_t * obj = lv_slider_create(lv_screen_active());
    lv_obj_add_style(obj, &style_indic, LV_PART_INDICATOR);
    lv_obj_add_style(obj, &style_indic_pr, LV_PART_INDICATOR | LV_STATE_PRESSED);
    lv_slider_set_value(obj, 70, LV_ANIM_OFF);
    lv_obj_center(obj);
}
#endif

```

3.2.14 Estensione del tema corrente

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE
#include "../lvgl_private.h"

static lv_style_t style_btn;

/*Will be called when the styles of the base theme are already added
to add new styles*/
static void new_theme_apply_cb(lv_theme_t * th, lv_obj_t * obj)
{
    LV_UNUSED(th);

    if(lv_obj_check_type(obj, &lv_button_class)) {
        lv_obj_add_style(obj, &style_btn, 0);
    }
}

static void new_theme_init_and_set(void)
{
    /*Initialize the styles*/
    lv_style_init(&style_btn);
    lv_style_set_bg_color(&style_btn, lv_palette_main(LV_PALETTE_GREEN));
    lv_style_set_border_color(&style_btn, lv_palette_darken(LV_PALETTE_GREEN, 3));
    lv_style_set_border_width(&style_btn, 3);

    /*Initialize the new theme from the current theme*/
    lv_theme_t * th_act = lv_display_get_theme(NULL);
    static lv_theme_t th_new;

```

(continues on next page)

(continua dalla pagina precedente)

```

th_new = *th_act;

/*Set the parent theme and the style apply callback for the new theme*/
lv_theme_set_parent(th_new, th_act);
lv_theme_set_apply_cb(th_new, new_theme_apply_cb);

/*Assign the new theme to the current display*/
lv_display_set_theme(NULL, &th_new);
}

/***
 * Extending the current theme
 */
void lv_example_style_14(void)
{
    lv_obj_t * btn;
    lv_obj_t * label;

    btn = lv_button_create(lv_screen_active());
    lv_obj_align(btn, LV_ALIGN_TOP_MID, 0, 20);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Original theme");

    new_theme_init_and_set();

    btn = lv_button_create(lv_screen_active());
    lv_obj_align(btn, LV_ALIGN_BOTTOM_MID, 0, -20);

    label = lv_label_create(btn);
    lv_label_set_text(label, "New theme");
}

#endif

```

3.2.15 Opacità e trasformazioni

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_BUTTON && LV_USE_LABEL

/***
 * Opacity and Transformations
 */
void lv_example_style_15(void)
{
    lv_obj_t * btn;
    lv_obj_t * label;

    /*Normal button*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 40);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, -70);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Normal");
    lv_obj_center(label);

    /*Set opacity
     *The button and the label is rendered to a layer first and that layer is blended*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 40);
    lv_obj_set_style_opa(btn, LV_OPA_50, 0);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, 0);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Opa:50%");
    lv_obj_center(label);

    /*Set transformations
     *The button and the label is rendered to a layer first and that layer is transformed*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 40);
    lv_obj_set_style_transform_rotation(btn, 150, 0);      /*15 deg*/
    lv_obj_set_style_transform_scale(btn, 256 + 64, 0);   /*1.25x*/
    lv_obj_set_style_transform_pivot_x(btn, 50, 0);
    lv_obj_set_style_transform_pivot_y(btn, 20, 0);
    lv_obj_set_style_opa(btn, LV_OPA_50, 0);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, 70);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Transf.");
    lv_obj_center(label);
}

#endif

```

3.2.16 Manopola metallica con gradiente conico

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

/***
 * Simulate metallic knob using conical gradient
 * For best effect set LV_GRADIENT_MAX_STOPS to 8 or at least 3
 */
void lv_example_style_16(void)
{
#if LV_GRADIENT_MAX_STOPS >= 8
    static const lv_color_t grad_colors[8] = {
        LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
        LV_COLOR_MAKE(0xff, 0xff, 0xff),
        LV_COLOR_MAKE(0xfa, 0xfa, 0xfa),

```

(continues on next page)

(continua dalla pagina precedente)

```

LV_COLOR_MAKE(0x79, 0x79, 0x79),
LV_COLOR_MAKE(0x4B, 0x4B, 0x4B),
LV_COLOR_MAKE(0x4B, 0x4B, 0x4B),
LV_COLOR_MAKE(0x70, 0x70, 0x70),
LV_COLOR_MAKE(0x8E, 0x8E, 0x8E),
};

#elif LV_GRADIENT_MAX_STOPS >= 3
static const lv_color_t grad_colors[3] = {
    LV_COLOR_MAKE(0xE8, 0xE8, 0xE8),
    LV_COLOR_MAKE(0xFF, 0xFF, 0xFF),
    LV_COLOR_MAKE(0x79, 0x79, 0x79),
};
#else
static const lv_color_t grad_colors[2] = {
    LV_COLOR_MAKE(0xE8, 0xE8, 0xE8),
    LV_COLOR_MAKE(0x79, 0x79, 0x79),
};
#endif

/*Create a style with gradient background and shadow*/
static lv_style_t style;
lv_style_init(&style);
lv_style_set_radius(&style, 500);
lv_style_set_bg_opa(&style, LV_OPA_COVER);
lv_style_set_shadow_color(&style, lv_color_black());
lv_style_set_shadow_width(&style, 50);
lv_style_set_shadow_offset_x(&style, 20);
lv_style_set_shadow_offset_y(&style, 20);
lv_style_set_shadow_opa(&style, LV_OPA_50);

/*First define a color gradient. In this example we use a gray color map with random values.*/
static lv_grad_dsc_t grad;

lv_grad_init_stops(&grad, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));

/*Make a conical gradient with the center in the middle of the object*/
#if LV_GRADIENT_MAX_STOPS >= 8
lv_grad_conical_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, 0, 120, LV_GRAD_EXTEND_REFLECT);
#elif LV_GRADIENT_MAX_STOPS >= 3
lv_grad_conical_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, 45, 125, LV_GRAD_EXTEND_REFLECT);
#else
lv_grad_conical_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, 45, 110, LV_GRAD_EXTEND_REFLECT);
#endif

/*Set gradient as background*/
lv_style_set_bg_grad(&style, &grad);

/*Create an object with the new style*/
lv_obj_t * obj = lv_obj_create(lv_screen_active());
lv_obj_add_style(obj, &style, 0);
lv_obj_set_size(obj, 200, 200);
lv_obj_center(obj);
}

#else

void lv_example_style_16(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, LV_PCT(80));
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS is not enabled");
    lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif /*LV_BUILD_EXAMPLES*/

```

3.2.17 Gradiente radiale come sfondo

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

/***
 * Using radial gradient as background
 */
void lv_example_style_17(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0x9B, 0x18, 0x42),
        LV_COLOR_MAKE(0x00, 0x00, 0x00),
    };

    int32_t width = lv_display_get_horizontal_resolution(NULL);
    int32_t height = lv_display_get_vertical_resolution(NULL);

    static lv_style_t style;
    lv_style_init(&style);

    /*First define a color gradient. In this example we use a purple to black color map.*/
    static lv_grad_dsc_t grad;

    lv_grad_init_stops(&grad, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));

    /*Make a radial gradient with the center in the middle of the object, extending to the farthest corner*/
    lv_grad_radial_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, LV_GRAD_RIGHT, LV_GRAD_BOTTOM, LV_GRAD_EXTEND_PAD);

    /*Set gradient as background*/
    lv_style_set_bg_grad(&style, &grad);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_set_size(obj, width, height);
    lv_obj_center(obj);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

}

#ifndef LV_USE_DRAW_SW_COMPLEX_GRADIENTS
void lv_example_style_17(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, LV_PCT(80));
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS is not enabled");
    lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif /*LV_BUILD_EXAMPLES*/

```

3.2.18 Gradienti per lo sfondo del pulsante

```

#include "../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES

#ifndef LV_USE_DRAW_SW_COMPLEX_GRADIENTS

/***
 * Using various gradients for button background
 */
void lv_example_style_18(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0x26, 0xa0, 0xda),
        LV_COLOR_MAKE(0x31, 0x47, 0x55),
    };

    /*Create a linear gradient going from the top left corner to the bottom at an angle, with reflected color map*/
    static lv_style_t style_with_linear_gradient_bg;
    static lv_grad_desc_t linear_gradient_desc; /*NOTE: the gradient descriptor must be static or global variable!*/

    lv_style_init(&style_with_linear_gradient_bg);
    lv_grad_init_stops(&linear_gradient_desc, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));
    lv_grad_linear_init(&linear_gradient_desc, lv_pct(0), lv_pct(0), lv_pct(20), lv_pct(100), LV_GRAD_EXTEND_REFLECT);
    lv_style_set_bg_grad(&style_with_linear_gradient_bg, &linear_gradient_desc);
    lv_style_set_bg_opa(&style_with_linear_gradient_bg, LV_OPA_COVER);

    /*Create a radial gradient with the center in the top left 1/3rd of the object, extending to the bottom right corner, with reflected color map*/
    static lv_style_t style_with_radial_gradient_bg;
    static lv_grad_desc_t radial_gradient_desc; /*NOTE: the gradient descriptor must be static or global variable!*/

    lv_style_init(&style_with_radial_gradient_bg);
    lv_grad_init_stops(&radial_gradient_desc, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));
    lv_grad_radial_init(&radial_gradient_desc, lv_pct(30), lv_pct(30), lv_pct(100), lv_pct(100), LV_GRAD_EXTEND_REFLECT);
    lv_style_set_bg_grad(&style_with_radial_gradient_bg, &radial_gradient_desc);
    lv_style_set_bg_opa(&style_with_radial_gradient_bg, LV_OPA_COVER);

    /*Create buttons with different gradient styles*/

    lv_obj_t * btn;
    lv_obj_t * label;

    /*Simple horizontal gradient*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_style_bg_color(btn, grad_colors[0], 0);
    lv_obj_set_style_bg_grad_color(btn, grad_colors[1], 0);
    lv_obj_set_style_bg_grad_dir(btn, LV_GRAD_DIR_HOR, 0);
    lv_obj_set_size(btn, 150, 50);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, -100);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Horizontal");
    lv_obj_center(label);

    /*Simple vertical gradient*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_style_bg_color(btn, grad_colors[0], 0);
    lv_obj_set_style_bg_grad_color(btn, grad_colors[1], 0);
    lv_obj_set_style_bg_grad_dir(btn, LV_GRAD_DIR_VER, 0);
    lv_obj_set_size(btn, 150, 50);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, -40);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Vertical");
    lv_obj_center(label);

    /*Complex linear gradient*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_add_style(btn, &style_with_linear_gradient_bg, 0);
    lv_obj_set_size(btn, 150, 50);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, 20);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Linear");
    lv_obj_center(label);

    /*Complex radial gradient*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_add_style(btn, &style_with_radial_gradient_bg, 0);
    lv_obj_set_size(btn, 150, 50);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, 80);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Radial");
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
void lv_example_style_18(void)
{
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * label = lv_label_create(lv_screen_active());
lv_obj_set_width(label, LV_PCT(80));
lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS is not enabled");
lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif /*LV_BUILD_EXAMPLES*/

```

3.2.19 Test tra stile di ricolorazione o sfondo modale pieno

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SLIDER && LV_USE_LOG

/**
 * Test between a full background modal and a recolor modal
 */
void lv_example_style_19(void)
{
    /*Add lv_example_style_11 as background*/
    lv_example_style_11();

    /* Set to 1 to enable recolor overlay instead of solid background */
#if 0
    /* Apply a screen-wide tint using recolor (efficient overlay).
     * This modifies the visual appearance by blending a semi-transparent color
     * over existing content without creating additional objects.
     * It's lighter on performance compared to a full-size background object.*/
    lv_obj_set_style_recolor(lv_screen_active(), lv_color_black(), 0);
    lv_obj_set_style_recolor_opa(lv_screen_active(), LV_OPA_50, 0);
#else
    /* Simulate a modal background by setting a semi-transparent black background
     * on lv_layer_top(), the highest built-in layer.
     * This method creates a new full-screen object and can consume more resources
     * compared to recolor, especially when using images or gradients.*/
    lv_obj_set_style_bg_color(lv_layer_top(), lv_color_black(), 0);
    lv_obj_set_style_bg_opa(lv_layer_top(), LV_OPA_50, 0);
#endif

    lv_obj_t * obj = lv_slider_create(lv_layer_top());
    lv_obj_center(obj);

    lv_refr_now(NULL); /*Update layouts and render*/

    lv_obj_invalidate(lv_screen_active());

    uint32_t t = lv_tick_get();
    lv_refr_now(NULL); /*Render only*/
    LV_LOG_USER("%s LV_PRIu32 %ms\n", lv_tick_elaps(t));
}
#endif

```

3.3 Animazioni

3.3.1 Avviare l'animazione su un evento

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_x((lv_obj_t *) var, v);
}

static void sw_event_cb(lv_event_t * e)
{
    lv_obj_t * sw = lv_event_get_target_obj(e);
    lv_obj_t * label = (lv_obj_t *) lv_event_get_user_data(e);

    if(lv_obj_has_state(sw, LV_STATE_CHECKED)) {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), 100);
        lv_anim_set_duration(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_overshoot);
        lv_anim_start(&a);
    } else {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), -lv_obj_get_width(label));
        lv_anim_set_duration(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_ease_in);
        lv_anim_start(&a);
    }
}

/** Start animation on an event
 */
void lv_example_anim_1(void)
{

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * label = lv_label_create(lv_screen_active());
lv_label_set_text(label, "Hello animations!");
lv_obj_set_pos(label, 100, 10);

lv_obj_t * sw = lv_switch_create(lv_screen_active());
lv_obj_center(sw);
lv_obj_add_state(sw, LV_STATE_CHECKED);
lv_obj_add_event_cb(sw, sw_event_cb, LV_EVENT_VALUE_CHANGED, label);
}

#endif

```

3.3.2 Riproduzione di animazione

```

#include "../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES || !LV_USE_SWITCH

static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_x((lv_obj_t *) var, v);
}

static void anim_size_cb(void * var, int32_t v)
{
    lv_obj_set_size((lv_obj_t *) var, v, v);
}

/***
 * Create a playback animation
 */
void lv_example_anim_2(void)
{
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_remove_flag(obj, LV_OBJ_FLAG_SCROLLABLE);
    lv_obj_set_style_bg_color(obj, lv_palette_main(LV_PALETTE_RED), 0);
    lv_obj_set_style_radius(obj, LV_RADIUS_CIRCLE, 0);

    lv_obj_align(obj, LV_ALIGN_LEFT_MID, 10, 0);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, obj);
    lv_anim_set_values(&a, 10, 50);
    lv_anim_set_duration(&a, 1000);
    lv_anim_set_reverse_delay(&a, 100);
    lv_anim_set_reverse_duration(&a, 300);
    lv_anim_set_repeat_delay(&a, 500);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_set_path_cb(&a, lv_anim_path_ease_in_out);

    lv_anim_set_exec_cb(&a, anim_size_cb);
    lv_anim_start(&a);
    lv_anim_set_exec_cb(&a, anim_x_cb);
    lv_anim_set_values(&a, 10, 240);
    lv_anim_start(&a);
}

#endif

```

3.3.3 Animazione di cubica di Bézier

```

#include "../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES || !LV_USE_SLIDER || !LV_USE_CHART || !LV_USE_BUTTON || !LV_USE_GRID

/***
 * the example show the use of cubic-bezier3 in animation.
 * the control point P1,P2 of cubic-bezier3 can be adjusted by slider.
 * and the chart shows the cubic-bezier3 in real time. you can click
 * run button see animation in current point of cubic-bezier3.
 */

#define CHART_POINTS_NUM 256

struct {
    lv_obj_t * anim_obj;
    lv_obj_t * chart;
    lv_chart_series_t * ser1;
    lv_obj_t * p1_slider;
    lv_obj_t * p2_slider;
    lv_obj_t * run_btn;
    int32_t p1;
    int32_t p2;
    lv_anim_t a;
} ginfo;

static int32_t anim_path_bezier3_cb(const lv_anim_t * a);
static void refer_chart_cubic_bezier(void);
static void run_button_event_handler(lv_event_t * e);
static void slider_event_cb(lv_event_t * e);
static void page_obj_init(lv_obj_t * par);
static void anim_x_cb(void * var, int32_t v);

/***
 * create an animation
 */
void lv_example_anim_3(void)
{
    static int32_t col_dsc[] = {LV_GRID_FR(1), 200, LV_GRID_FR(1), LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {30, 10, 10, LV_GRID_FR(1), LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * cont = lv_obj_create(lv_screen_active());
lv_obj_set_style_pad_all(cont, 2, LV_PART_MAIN);
lv_obj_set_style_pad_column(cont, 10, LV_PART_MAIN);
lv_obj_set_style_pad_row(cont, 10, LV_PART_MAIN);
lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);
lv_obj_set_size(cont, 320, 240);
lv_obj_center(cont);

page_obj_init(cont);

lv_anim_init(&ginfo.a);
lv_anim_set_var(&ginfo.a, ginfo.anim_obj);
int32_t end = lv_obj_get_style_width(cont, LV_PART_MAIN) -
    lv_obj_get_style_width(ginfo.anim_obj, LV_PART_MAIN) - 10;
lv_anim_set_values(&ginfo.a, 5, end);
lv_anim_set_duration(&ginfo.a, 2000);
lv_anim_set_exec_cb(&ginfo.a, anim_x_cb);
lv_anim_set_path_cb(&ginfo.a, anim_path_bezier3_cb);

refer_chart_cubic_bezier();
}

static int32_t anim_path_bezier3_cb(const lv_anim_t * a)
{
    int32_t t = lv_map(a->act_time, 0, a->duration, 0, 1024);
    int32_t step = lv_bezier3(t, 0, ginfo.p1, ginfo.p2, 1024);
    int32_t new_value;
    new_value = step * (a->end_value - a->start_value);
    new_value = new_value >> 10;
    new_value += a->start_value;
    return new_value;
}

static void refer_chart_cubic_bezier(void)
{
    for(uint16_t i = 0; i <= CHART_POINTS_NUM; i++) {
        int32_t t = i * (1024 / CHART_POINTS_NUM);
        int32_t step = lv_bezier3(t, 0, ginfo.p1, ginfo.p2, 1024);
        lv_chart_set_series_value_by_id2(ginfo.chart, ginfo.ser1, i, t, step);
    }
    lv_chart_refresh(ginfo.chart);
}

static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_style_translate_x((lv_obj_t *)var, v, LV_PART_MAIN);
}

static void run_button_event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_CLICKED) {
        lv_anim_start(&ginfo.a);
    }
}

static void slider_event_cb(lv_event_t * e)
{
    char buf[16];
    lv_obj_t * label;
    lv_obj_t * slider = lv_event_get_target_obj(e);

    if(slider == ginfo.p1_slider) {
        label = ginfo.p1_label;
        ginfo.p1 = lv_slider_get_value(slider);
        lv_snprintf(buf, sizeof(buf), "p1:%d", ginfo.p1);
    }
    else {
        label = ginfo.p2_label;
        ginfo.p2 = lv_slider_get_value(slider);
        lv_snprintf(buf, sizeof(buf), "p2:%d", ginfo.p2);
    }

    lv_label_set_text(label, buf);
    refer_chart_cubic_bezier();
}

static void page_obj_init(lv_obj_t * par)
{
    ginfo.anim_obj = lv_obj_create(par);
    lv_obj_set_size(ginfo.anim_obj, 30, 30);
    lv_obj_set_align(ginfo.anim_obj, LV_ALIGN_TOP_LEFT);
    lv_obj_remove_flag(ginfo.anim_obj, LV_OB_FLAG_SCROLLABLE);
    lv_obj_set_style_bg_color(ginfo.anim_obj, lv_palette_main(LV_PALETTE_RED), LV_PART_MAIN);
    lv_obj_set_grid_cell(ginfo.anim_obj, LV_GRID_ALIGN_START, 0, 1, LV_GRID_ALIGN_START, 0, 1);

    ginfo.p1_label = lv_label_create(par);
    ginfo.p2_label = lv_label_create(par);
    lv_label_set_text(ginfo.p1_label, "p1:0");
    lv_label_set_text(ginfo.p2_label, "p2:0");
    lv_obj_set_grid_cell(ginfo.p1_label, LV_GRID_ALIGN_START, 0, 1, LV_GRID_ALIGN_START, 1, 1);
    lv_obj_set_grid_cell(ginfo.p2_label, LV_GRID_ALIGN_START, 0, 1, LV_GRID_ALIGN_START, 2, 1);

    ginfo.p1_slider = lv_slider_create(par);
    ginfo.p2_slider = lv_slider_create(par);
    lv_slider_set_range(ginfo.p1_slider, 0, 1024);
    lv_slider_set_range(ginfo.p2_slider, 0, 1024);
    lv_obj_set_style_pad_all(ginfo.p1_slider, 2, LV_PART_KNOB);
    lv_obj_add_event_cb(ginfo.p1_slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_add_event_cb(ginfo.p2_slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_set_grid_cell(ginfo.p1_slider, LV_GRID_ALIGN_STRETCH, 1, 1, LV_GRID_ALIGN_START, 1, 1);
    lv_obj_set_grid_cell(ginfo.p2_slider, LV_GRID_ALIGN_STRETCH, 1, 1, LV_GRID_ALIGN_START, 2, 1);

    ginfo.run_btn = lv_button_create(par);
    lv_obj_add_event_cb(ginfo.run_btn, run_button_event_handler, LV_EVENT_CLICKED, NULL);
    lv_obj_t * btn_label = lv_label_create(ginfo.run_btn);
    lv_label_set_text(btn_label, LV_SYMBOL_PLAY);
    lv_obj_center(btn_label);
    lv_obj_set_grid_cell(ginfo.run_btn, LV_GRID_ALIGN_STRETCH, 2, 1, LV_GRID_ALIGN_STRETCH, 1, 2);

    ginfo.chart = lv_chart_create(par);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_style_pad_all(ginfo.chart, 0, LV_PART_MAIN);
lv_obj_set_size(ginfo.chart, 0, 0, LV_PART_INDICATOR);
lv_chart_set_type(ginfo.chart, LV_CHART_TYPE_SCATTER);
ginfo.series1 = lv_chart_add_series(ginfo.chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
lv_chart_set_axis_range(ginfo.chart, LV_CHART_AXIS_PRIMARY_Y, 0, 1024);
lv_chart_set_axis_range(ginfo.chart, LV_CHART_AXIS_PRIMARY_X, 0, 1024);
lv_chart_set_point_count(ginfo.chart, CHART_POINTS_NUM);
lv_obj_set_grid_cell(ginfo.chart, LV_GRID_ALIGN_STRETCH, 0, 3, LV_GRID_ALIGN_STRETCH, 3, 1);
}

#endif

```

3.3.4 Pausa animazione

```

#include "../lv_examples.h"
#ifndef LV_USE_SWITCH
static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_x((lv_obj_t *) var, v);
}

static void timer_cb(lv_timer_t * timer)
{
    lv_anim_t * anim = (lv_anim_t *) lv_timer_get_user_data(timer);
    lv_anim_pause_for(anim, 1000);
    lv_timer_delete(timer);
}
static void sw_event_cb(lv_event_t * e)
{
    lv_obj_t * sw = lv_event_get_target_obj(e);
    lv_obj_t * label = (lv_obj_t *) lv_event_get_user_data(e);

    if(lv_obj_has_state(sw, LV_STATE_CHECKED)) {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), 100);
        lv_anim_set_duration(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_overshoot);
        lv_timer_create(timer_cb, 200, lv_anim_start(&a));
    }
    else {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), -lv_obj_get_width(label));
        lv_anim_set_duration(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_ease_in);
        lv_timer_create(timer_cb, 200, lv_anim_start(&a));
    }
}
/** Start animation on an event */
void lv_example_anim_4(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Hello animations!");
    lv_obj_set_pos(label, 100, 10);

    lv_obj_t * sw = lv_switch_create(lv_screen_active());
    lv_obj_center(sw);
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_event_cb(sw, sw_event_cb, LV_EVENT_VALUE_CHANGED, label);
}
#endif

```

3.3.5 Timeline animazione

```

#include "../lv_examples.h"
#ifndef LV_USE_FLEX
static const int32_t obj_width = 90;
static const int32_t obj_height = 70;

static void set_width(lv_anim_t * var, int32_t v)
{
    lv_obj_set_width((lv_obj_t *) var->var, v);
}

static void set_height(lv_anim_t * var, int32_t v)
{
    lv_obj_set_height((lv_obj_t *) var->var, v);
}

static void set_slider_value(lv_anim_t * var, int32_t v)
{
    lv_slider_set_value((lv_obj_t *) var->var, v, LV_ANIM_OFF);
}

static void btn_start_event_handler(lv_event_t * e)
{
    lv_obj_t * btn = lv_event_get_current_target_obj(e);
    lv_anim_timeline_t * anim_timeline = (lv_anim_timeline_t *) lv_event_get_user_data(e);

    bool reverse = lv_obj_has_state(btn, LV_STATE_CHECKED);

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_anim_timeline_set_reverse(anim_timeline, reverse);
lv_anim_timeline_start(anim_timeline);
}

static void btn_pause_event_handler(lv_event_t * e)
{
    lv_anim_timeline_t * anim_timeline = (lv_anim_timeline_t *) lv_event_get_user_data(e);
    lv_anim_timeline_pause(anim_timeline);
}

static void slider_prg_event_handler(lv_event_t * e)
{
    lv_obj_t * slider = lv_event_get_current_target_obj(e);
    lv_anim_timeline_t * anim_timeline = (lv_anim_timeline_t *) lv_event_get_user_data(e);
    int32_t progress = lv_slider_get_value(slider);
    lv_anim_timeline_set_progress(anim_timeline, (uint16_t) progress);
}

/**
 * Create an animation timeline
 */
void lv_example_anim_timeline_1(void)
{
    /* Create anim timeline */
    lv_anim_timeline_t * anim_timeline = lv_anim_timeline_create();

    lv_obj_t * par = lv_screen_active();
    lv_obj_set_flex_flow(par, LV_FLEX_FLOW_ROW);
    lv_obj_set_flex_align(par, LV_FLEX_ALIGN_SPACE_AROUND, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    /* create btn_start */
    lv_obj_t * btn_start = lv_button_create(par);
    lv_obj_add_event_cb(btn_start, btn_start_event_handler, LV_EVENT_VALUE_CHANGED, anim_timeline);
    lv_obj_add_flag(btn_start, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj_add_flag(btn_start, LV_OBJ_FLAG_CHECKABLE);
    lv_obj_align(btn_start, LV_ALIGN_TOP_MID, -100, 20);

    lv_obj_t * label_start = lv_label_create(btn_start);
    lv_label_set_text(label_start, "Start");
    lv_obj_center(label_start);

    /* create btn_pause */
    lv_obj_t * btn_pause = lv_button_create(par);
    lv_obj_add_event_cb(btn_pause, btn_pause_event_handler, LV_EVENT_CLICKED, anim_timeline);
    lv_obj_add_flag(btn_pause, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj_align(btn_pause, LV_ALIGN_TOP_MID, 100, 20);

    lv_obj_t * label_pause = lv_label_create(btn_pause);
    lv_label_set_text(label_pause, "Pause");
    lv_obj_center(label_pause);

    /* create slider_prg */
    lv_obj_t * slider_prg = lv_slider_create(par);
    lv_obj_add_event_cb(slider_prg, slider_prg_event_handler, LV_EVENT_VALUE_CHANGED, anim_timeline);
    lv_obj_add_flag(slider_prg, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj_align(slider_prg, LV_ALIGN_BOTTOM_MID, 0, -20);
    lv_slider_set_range(slider_prg, 0, LV_ANIM_TIMELINE_PROGRESS_MAX);

    /* create 3 objects */
    lv_obj_t * obj1 = lv_obj_create(par);
    lv_obj_set_size(obj1, obj_width, obj_height);
    lv_obj_set_scrollbar_mode(obj1, LV_SCROLLBAR_MODE_OFF);

    lv_obj_t * obj2 = lv_obj_create(par);
    lv_obj_set_size(obj2, obj_width, obj_height);
    lv_obj_set_scrollbar_mode(obj2, LV_SCROLLBAR_MODE_OFF);

    lv_obj_t * obj3 = lv_obj_create(par);
    lv_obj_set_size(obj3, obj_width, obj_height);
    lv_obj_set_scrollbar_mode(obj3, LV_SCROLLBAR_MODE_OFF);

    /* anim-slider */
    lv_anim_t a_slider;
    lv_anim_init(&a_slider);
    lv_anim_set_var(&a_slider, slider_prg);
    lv_anim_set_values(&a_slider, 0, LV_ANIM_TIMELINE_PROGRESS_MAX);
    lv_anim_set_custom_exec_cb(&a_slider, set_slider_value);
    lv_anim_set_path_cb(&a_slider, lv_anim_path_linear);
    lv_anim_set_duration(&a_slider, 700);

    /* anim-obj1 */
    lv_anim_t a1;
    lv_anim_init(&a1);
    lv_anim_set_var(&a1, obj1);
    lv_anim_set_values(&a1, 0, obj_width);
    lv_anim_set_custom_exec_cb(&a1, set_width);
    lv_anim_set_path_cb(&a1, lv_anim_path_overshoot);
    lv_anim_set_duration(&a1, 300);

    lv_anim_t a2;
    lv_anim_init(&a2);
    lv_anim_set_var(&a2, obj1);
    lv_anim_set_values(&a2, 0, obj_height);
    lv_anim_set_custom_exec_cb(&a2, set_height);
    lv_anim_set_path_cb(&a2, lv_anim_path_ease_out);
    lv_anim_set_duration(&a2, 300);

    /* anim-obj2 */
    lv_anim_t a3;
    lv_anim_init(&a3);
    lv_anim_set_var(&a3, obj2);
    lv_anim_set_values(&a3, 0, obj_width);
    lv_anim_set_custom_exec_cb(&a3, set_width);
    lv_anim_set_path_cb(&a3, lv_anim_path_overshoot);
    lv_anim_set_duration(&a3, 300);

    lv_anim_t a4;
    lv_anim_init(&a4);
    lv_anim_set_var(&a4, obj2);
    lv_anim_set_values(&a4, 0, obj_height);
    lv_anim_set_custom_exec_cb(&a4, set_height);
    lv_anim_set_path_cb(&a4, lv_anim_path_ease_out);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_anim_set_duration(&a4, 300);

/* anim-obj3 */
lv_anim_t a5;
lv_anim_init(&a5);
lv_anim_set_var(&a5, obj3);
lv_anim_set_values(&a5, 0, obj_width);
lv_anim_set_custom_exec_cb(&a5, set_width);
lv_anim_set_path_cb(&a5, lv_anim_path_overshoot);
lv_anim_set_duration(&a5, 300);

lv_anim_t a6;
lv_anim_init(&a6);
lv_anim_set_var(&a6, obj3);
lv_anim_set_values(&a6, 0, obj_height);
lv_anim_set_custom_exec_cb(&a6, set_height);
lv_anim_set_path_cb(&a6, lv_anim_path_ease_out);
lv_anim_set_duration(&a6, 300);

/* add animations to timeline */
lv_anim_timeline_add(anim_timeline, 0, &a1);
lv_anim_timeline.add(anim_timeline, 0, &a2);
lv_anim_timeline.add(anim_timeline, 200, &a3);
lv_anim_timeline.add(anim_timeline, 200, &a4);
lv_anim_timeline.add(anim_timeline, 400, &a5);
lv_anim_timeline.add(anim_timeline, 400, &a6);

lv_anim_timeline_set_progress(anim_timeline, LV_ANIM_TIMELINE_PROGRESS_MAX);
}

#endif

```

3.4 Eventi

3.4.1 Evento clic pulsante

```

#include "../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES || !LV_USE_SWITCH

static void event_cb(lv_event_t * e)
{
    LV_LOG_USER("Clicked");

    static uint32_t cnt = 1;
    lv_obj_t * btn = lv_event_get_target_obj(e);
    lv_obj_t * label = lv_obj_get_child(btn, 0);
    lv_label_set_text_fmt(label, "%" LV_PRIu32, cnt);
    cnt++;
}

/**
 * Add click event to a button
 */
void lv_example_event_click(void)
{
    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 50);
    lv_obj_center(btn);
    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_CLICKED, NULL);

    lv_obj_t * label = lv_label_create(btn);
    lv_label_set_text(label, "Click me!");
    lv_obj_center(label);
}
#endif

```

3.4.2 Sequenze di clic

```

#include "../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES || !LV_USE_LABEL

static void short_click_event_cb(lv_event_t * e)
{
    LV_LOG_USER("Short clicked");

    lv_obj_t * info_label = (lv_obj_t *) lv_event_get_user_data(e);
    lv_indev_t * indev = (lv_indev_t *) lv_event_get_param(e);
    uint8_t cnt = lv_indev_get_short_click_streak(indev);
    lv_label_set_text_fmt(info_label, "Short click streak: %u", cnt);
}

static void streak_event_cb(lv_event_t * e)
{
    lv_obj_t * btn = lv_event_get_target_obj(e);
    lv_obj_t * label = lv_obj_get_child(btn, 0);
    const char * text = (const char *) lv_event_get_user_data(e);
    lv_label_set_text(label, text);
}

void lv_example_event_streak(void)
{
    lv_obj_t * info_label = lv_label_create(lv_screen_active());
    lv_label_set_text(info_label, "No events yet");

    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 50);
    lv_obj_center(btn);
    lv_obj_add_event_cb(btn, short_click_event_cb, LV_EVENT_SHORT_CLICKED, info_label);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_add_event_cb(btn, streak_event_cb, LV_EVENT_SINGLE_CLICKED, (void *) "Single clicked");
lv_obj_add_event_cb(btn, streak_event_cb, LV_EVENT_DOUBLE_CLICKED, (void *) "Double clicked");
lv_obj_add_event_cb(btn, streak_event_cb, LV_EVENT_TRIPLE_CLICKED, (void *) "Triple clicked");

lv_obj_t * label = lv_label_create(btn);
lv_label_set_text(label, "Click me!");
lv_obj_center(label);
}

#endif

```

3.4.3 Gestione di eventi multipli

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * label = (lv_obj_t *) lv_event_get_user_data(e);

    switch(code) {
        case LV_EVENT_PRESSED:
            lv_label_set_text(label, "The last button event:\nLV_EVENT_PRESSED");
            break;
        case LV_EVENT_CLICKED:
            lv_label_set_text(label, "The last button event:\nLV_EVENT_CLICKED");
            break;
        case LV_EVENT_LONG_PRESSED:
            lv_label_set_text(label, "The last button event:\nLV_EVENT_LONG_PRESSED");
            break;
        case LV_EVENT_LONG_PRESSED_REPEAT:
            lv_label_set_text(label, "The last button event:\nLV_EVENT_LONG_PRESSED_REPEAT");
            break;
        default:
            break;
    }
}

/**
 * Handle multiple events
 */
void lv_example_event_button(void)
{
    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 50);
    lv_obj_center(btn);

    lv_obj_t * btn_label = lv_label_create(btn);
    lv_label_set_text(btn_label, "Click me!");
    lv_obj_center(btn_label);

    lv_obj_t * info_label = lv_label_create(lv_screen_active());
    lv_label_set_text(info_label, "The last button event:\nNone");

    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_ALL, info_label);
}

#endif

```

3.4.4 Evento "bubbling"

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static void event_cb(lv_event_t * e)
{
    /*The original target of the event. Can be the buttons or the container*/
    lv_obj_t * target = lv_event_get_target_obj(e);

    /*The current target is always the container as the event is added to it*/
    lv_obj_t * cont = lv_event_get_current_target_obj(e);

    /*If container was clicked do nothing*/
    if(target == cont) return;

    /*Make the clicked buttons red*/
    lv_obj_set_style_bg_color(target, lv_palette_main(LV_PALETTE_RED), 0);
}

/**
 * Demonstrate event bubbling
 */
void lv_example_event_bubble(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 290, 200);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    uint32_t i;
    for(i = 0; i < 30; i++) {
        lv_obj_t * btn = lv_button_create(cont);
        lv_obj_set_size(btn, 70, 50);
        lv_obj_add_flag(btn, LV_OBJ_FLAG_EVENT_BUBBLE);

        lv_obj_t * label = lv_label_create(btn);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);
        lv_obj_center(label);
    }

    lv_obj_add_event_cb(cont, event_cb, LV_EVENT_CLICKED, NULL);
}

```

(continues on next page)

(continua dalla pagina precedente)

#endif

3.4.5 Evento "trickle-down"

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

/***
 * Demonstrate event trickle
 */
void lv_example_event_trickle(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 290, 200);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    static lv_style_t style_black;
    lv_style_init(&style_black);
    lv_style_set_text_color(&style_black, lv_color_white());
    lv_style_set_bg_color(&style_black, lv_color_black());

    /*Enable event trickle-down on the container*/
    lv_obj_add_flag(cont, LV_OBJ_FLAG_EVENT_TRICKLE);

    lv_obj_add_style(cont, &style_black, LV_STATE_PRESSED);

    uint32_t i;
    for(i = 0; i < 9; i++) {
        lv_obj_t * subcont = lv_obj_create(cont);
        lv_obj_set_size(subcont, 70, 50);
        lv_obj_t * label = lv_label_create(subcont);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);

        /*Add style to the label when clicked*/
        lv_obj_add_style(subcont, &style_black, LV_STATE_FOCUSED);
    }
}

#endif
```

3.4.6 Evento "Draw"

```
#include "../lv_examples.h"

#if LV_BUILD_EXAMPLES

static int32_t size = 0;
static bool size_dec = false;

static void timer_cb(lv_timer_t * timer)
{
    lv_obj_invalidate((lv_obj_t *) lv_timer_get_user_data(timer));
    if(size_dec) size--;
    else size++;

    if(size == 50) size_dec = true;
    else if(size == 0) size_dec = false;
}

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *) lv_draw_task_get_draw_dsc(draw_task);
    if(base_dsc->part == LV_PART_MAIN) {
        lv_draw_rect_dsc_t draw_dsc;
        lv_draw_rect_dsc_init(&draw_dsc);
        draw_dsc.bg_color = lv_color_hex(0xffffaaa);
        draw_dsc.radius = LV_RADIUS_CIRCLE;
        draw_dsc.border_color = lv_color_hex(0xffff5555);
        draw_dsc.border_width = 2;
        draw_dsc.outline_color = lv_color_hex(0xffff0000);
        draw_dsc.outline_paa = 3;
        draw_dsc.outline_width = 2;

        lv_area_t a;
        a.x1 = 0;
        a.y1 = 0;
        a.x2 = size;
        a.y2 = size;
        lv_area_t obj_coords;
        lv_obj_get_coords(obj, &obj_coords);
        lv_area_align(&obj_coords, &a, LV_ALIGN_CENTER, 0, 0);

        lv_draw_rect(base_dsc->layer, &draw_dsc, &a);
    }
}

/***
 * Demonstrate the usage of draw event
 */
void lv_example_event_draw(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 200, 200);
    lv_obj_center(cont);
    lv_obj_add_event_cb(cont, event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(cont, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
    lv_timer_create(timer_cb, 30, cont);
}

#endif
```

3.5 Layout

3.5.1 Flex

Un semplice layout di riga e colonna con flexbox

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/***
 * A simple row and a column layout with flexbox
 */
void lv_example_flex_1(void)
{
    /*Create a container with ROW flex direction*/
    lv_obj_t * cont_row = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont_row, 300, 75);
    lv_obj_align(cont_row, LV_ALIGN_TOP_MID, 0, 5);
    lv_obj_set_flex_flow(cont_row, LV_FLEX_FLOW_ROW);

    /*Create a container with COLUMN flex direction*/
    lv_obj_t * cont_col = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont_col, 200, 150);
    lv_obj_align_to(cont_col, cont_row, LV_ALIGN_OUT_BOTTOM_MID, 0, 5);
    lv_obj_set_flex_flow(cont_col, LV_FLEX_FLOW_COLUMN);

    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_obj_t * obj;
        lv_obj_t * label;

        /*Add items to the row*/
        obj = lv_button_create(cont_row);
        lv_obj_set_size(obj, 100, LV_PCT(100));

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item: %" LV_PRIu32 "", i);
        lv_obj_center(label);

        /*Add items to the column*/
        obj = lv_button_create(cont_col);
        lv_obj_set_size(obj, LV_PCT(100), LV_SIZE_CONTENT);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item: %" LV_PRIu32, i);
        lv_obj_center(label);
    }
}

#endif
```

Disporre gli elementi in righe con "wrap" [riavvolgimento] e spaziatura uniforme

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/***
 * Arrange items in rows with wrap and place the items to get even space around them.
 */
void lv_example_flex_2(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_flex_flow(&style, LV_FLEX_FLOW_ROW_WRAP);
    lv_style_set_flex_main_place(&style, LV_FLEX_ALIGN_SPACE_EVENLY);
    lv_style_set_layout(&style, LV_LAYOUT_FLEX);

    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_add_style(cont, &style, 0);

    uint32_t i;
    for(i = 0; i < 8; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);
        lv_obj_add_flag(obj, LV_OBJ_FLAG_CHECKABLE);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);
        lv_obj_center(label);
    }
}

#endif
```

Dimostrazione della flex grow.

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/***
 * Demonstrate flex grow.
 */
void lv_example_flex_3(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * obj;
obj = lv_obj_create(cont);
lv_obj_set_size(obj, 40, 40); /*Fix size*/

obj = lv_obj_create(cont);
lv_obj_set_height(obj, 40);
lv_obj_set_flex_grow(obj, 1); /*1 portion from the free space*/

obj = lv_obj_create(cont);
lv_obj_set_height(obj, 40);
lv_obj_set_flex_grow(obj, 2); /*2 portion from the free space*/

obj = lv_obj_create(cont);
lv_obj_set_size(obj, 40, 40); /*Fix size. It is flushed to the right by the "grow" items*/
}

#endif

```

Dimostrazione della flex grow.

```

#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/***
 * Reverse the order of flex items
 */
void lv_example_flex_4(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_COLUMN_REVERSE);

    uint32_t i;
    for(i = 0; i < 6; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 100, 50);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item: %" LV_PRIu32, i);
        lv_obj_center(label);
    }
}
#endif

```

Dimostrazione delle proprietà di stile degli spazi tra colonne e righe

```

#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

static void row_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_row((lv_obj_t *) obj, v, 0);
}

static void column_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_column((lv_obj_t *) obj, v, 0);
}

/***
 * Demonstrate the effect of column and row gap style properties
 */
void lv_example_flex_5(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    uint32_t i;
    for(i = 0; i < 9; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);
        lv_obj_center(label);
    }

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, cont);
    lv_anim_set_values(&a, 0, 10);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);

    lv_anim_set_exec_cb(&a, row_gap_anim);
    lv_anim_set_duration(&a, 500);
    lv_anim_set_reverse_duration(&a, 500);
    lv_anim_start(&a);

    lv_anim_set_exec_cb(&a, column_gap_anim);
    lv_anim_set_duration(&a, 3000);
    lv_anim_set_reverse_duration(&a, 3000);
    lv_anim_start(&a);
}
#endif

```

La direzione di base RTL modifica l'ordine degli elementi

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/***
 * RTL base direction changes order of the items.
 * Also demonstrate how horizontal scrolling works with RTL.
 */
void lv_example_flex_6(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_style_base_dir(cont, LV_BASE_DIR_RTL, 0);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    uint32_t i;
    for(i = 0; i < 20; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);
        lv_obj_center(label);
    }
}
#endif
```

3.5.2 Grid

Una semplice grid

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/***
 * A simple grid
 */
void lv_example_grid_1(void)
{
    static int32_t col_dsc[] = {70, 70, 70, LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {50, 50, 50, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_style_grid_column_dsc_array(cont, col_dsc, 0);
    lv_obj_set_style_grid_row_dsc_array(cont, row_dsc, 0);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_layout(cont, LV_LAYOUT_GRID);

    lv_obj_t * label;
    lv_obj_t * obj;

    uint8_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_button_create(cont);
        /*Stretch the cell horizontally and vertically too
         *Set span to 1 to make the cell 1 column/row sized*/
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                            LV_GRID_ALIGN_STRETCH, row, 1);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%d, %d", col, row);
        lv_obj_center(label);
    }
}
#endif
```

Posizionamento e ampiezza delle celle

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/***
 * Demonstrate cell placement and span
 */
void lv_example_grid_2(void)
{
    static int32_t col_dsc[] = {70, 70, 70, LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {50, 50, 50, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);

    lv_obj_t * label;
    lv_obj_t * obj;

    /*Cell to 0:0 and align to to the start (left/top) horizontally and vertically too*/
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_START, 0, 1,
                        LV_GRID_ALIGN_START, 0, 1);
    label = lv_label_create(obj);
    lv_label_set_text(label, "C0, r0");
}
```

(continues on next page)

(continua dalla pagina precedente)

```

/*Cell to 1;0 and align to to the start (left) horizontally and center vertically too*/
obj = lv_obj_create(cont);
lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_START, 1, 1,
                     LV_GRID_ALIGN_CENTER, 0, 1);
label = lv_label_create(obj);
lv_label_set_text(label, "c1, r0");

/*Cell to 2;0 and align to to the start (left) horizontally and end (bottom) vertically too*/
obj = lv_obj_create(cont);
lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_START, 2, 1,
                     LV_GRID_ALIGN_END, 0, 1);
label = lv_label_create(obj);
lv_label_set_text(label, "c2, r0");

/*Cell to 1;1 but 2 column wide (span = 2).Set width and height to stretched.*/
obj = lv_obj_create(cont);
lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, 1, 2,
                     LV_GRID_ALIGN_STRETCH, 1, 1);
label = lv_label_create(obj);
lv_label_set_text(label, "c1-2, r1");

/*Cell to 0;1 but 2 rows tall (span = 2).Set width and height to stretched.*/
obj = lv_obj_create(cont);
lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, 0, 1,
                     LV_GRID_ALIGN_STRETCH, 1, 2);
label = lv_label_create(obj);
lv_label_set_text(label, "c0\nr1-2");
}

#endif

```

La -free unit- della grid

```

#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/**
 * Demonstrate grid's "free unit"
 */
void lv_example_grid_3(void)
{
    /*Column 1: fix width 60 px
     *Column 2: 1 unit from the remaining free space
     *Column 3: 2 unit from the remaining free space*/
    static int32_t col_dsc[] = {60, LV_GRID_FR(1), LV_GRID_FR(2), LV_GRID_TEMPLATE_LAST};

    /*Row 1: fix width 50 px
     *Row 2: 1 unit from the remaining free space
     *Row 3: fix width 50 px*/
    static int32_t row_dsc[] = {50, LV_GRID_FR(1), 50, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);

    lv_obj_t * label;
    lv_obj_t * obj;
    uint8_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_obj_create(cont);
        /*Stretch the cell horizontally and vertically too
         *Set span to 1 to make the cell 1 column/row sized*/
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                            LV_GRID_ALIGN_STRETCH, row, 1);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%d,%d", col, row);
        lv_obj_center(label);
    }
}
#endif

```

Posizionamento delle tracce

```

#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/**
 * Demonstrate track placement
 */
void lv_example_grid_4(void)
{
    static int32_t col_dsc[] = {60, 60, 60, LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {45, 45, 45, LV_GRID_TEMPLATE_LAST};

    /*Add space between the columns and move the rows to the bottom (end)*/

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_grid_align(cont, LV_GRID_ALIGN_SPACE_BETWEEN, LV_GRID_ALIGN_END);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * label;
lv_obj_t * obj;
uint8_t i;
for(i = 0; i < 9; i++) {
    uint8_t col = i % 3;
    uint8_t row = i / 3;

    obj = lv_obj_create(cont);
    /*Stretch the cell horizontally and vertically too
     *Set span to 1 to make the cell 1 column/row sized*/
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                         LV_GRID_ALIGN_STRETCH, row, 1);

    label = lv_label_create(obj);
    lv_label_set_text_fmt(label, "%d,%d", col, row);
    lv_obj_center(label);
}

#endif
}

```

Spazi tra colonne e righe

```

#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

static void row_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_row((lv_obj_t *) obj, v, 0);
}

static void column_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_column((lv_obj_t *) obj, v, 0);
}

/***
 * Demonstrate column and row gap
 */
void lv_example_grid_5(void)
{

/*60x60 cells*/
static int32_t col_dsc[] = {60, 60, 60, LV_GRID_TEMPLATE_LAST};
static int32_t row_dsc[] = {45, 45, 45, LV_GRID_TEMPLATE_LAST};

/*Create a container with grid*/
lv_obj_t * cont = lv_obj_create(lv_screen_active());
lv_obj_set_size(cont, 300, 220);
lv_obj_center(cont);
lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);

lv_obj_t * label;
lv_obj_t * obj;
uint8_t i;
for(i = 0; i < 9; i++) {
    uint8_t col = i % 3;
    uint8_t row = i / 3;

    obj = lv_obj_create(cont);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                         LV_GRID_ALIGN_STRETCH, row, 1);
    label = lv_label_create(obj);
    lv_label_set_text_fmt(label, "%d,%d", col, row);
    lv_obj_center(label);
}

lv_anim_t a;
lv_anim_init(&a);
lv_anim_set_var(&a, cont);
lv_anim_set_values(&a, 0, 10);
lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);

lv_anim_set_exec_cb(&a, row_gap_anim);
lv_anim_set_duration(&a, 500);
lv_anim_set_reverse_duration(&a, 500);
lv_anim_start(&a);

lv_anim_set_exec_cb(&a, column_gap_anim);
lv_anim_set_duration(&a, 3000);
lv_anim_set_reverse_duration(&a, 3000);
lv_anim_start(&a);
}

#endif
}

```

Direzione RTL sulla griglia

```

#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/***
 * Demonstrate RTL direction on grid
 */
void lv_example_grid_6(void)
{

static int32_t col_dsc[] = {60, 60, 60, LV_GRID_TEMPLATE_LAST};
static int32_t row_dsc[] = {45, 45, 45, LV_GRID_TEMPLATE_LAST};

/*Create a container with grid*/
lv_obj_t * cont = lv_obj_create(lv_screen_active());
lv_obj_set_size(cont, 300, 220);

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_center(cont);
lv_obj_set_style_base_dir(cont, LV_BASE_DIRRTL, 0);
lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);

lv_obj_t * label;
lv_obj_t * obj;
uint8_t i;
for(i = 0; i < 9; i++) {
    uint8_t col = i % 3;
    uint8_t row = i / 3;

    obj = lv_obj_create(cont);
    /*Stretch the cell horizontally and vertically too
     *Set span to 1 to make the cell 1 column/row sized*/
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                         LV_GRID_ALIGN_STRETCH, row, 1);

    label = lv_label_create(obj);
    lv_label_set_text_fmt(label, "%d,%d", col, row);
    lv_obj_center(label);
}

#endif
}

```

3.6 Scrolling

3.6.1 Scrolling nidificato

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

static lv_obj_t * panel;
static lv_obj_t * save_button;
static lv_obj_t * restore_button;
static int saved_scroll_x;
static int saved_scroll_y;

static void scroll_update_cb(lv_event_t * e);
static void button_event_cb(lv_event_t * e);

static void scroll_update_cb(lv_event_t * e)
{
    LV_UNUSED(e);

    LV_LOG("scroll info: x:%3" LV_PRId32", y:%3" LV_PRId32", top:%3" LV_PRId32", "
           "bottom:%3" LV_PRId32", left:%3" LV_PRId32", right:%3" LV_PRId32"\n",
           lv_obj_get_scroll_x(panel),
           lv_obj_get_scroll_y(panel),
           lv_obj_get_scroll_top(panel),
           lv_obj_get_scroll_bottom(panel),
           lv_obj_get_scroll_left(panel),
           lv_obj_get_scroll_right(panel)
    );
}

static void button_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);

    if(obj == save_button) {
        saved_scroll_x = lv_obj_get_scroll_x(panel);
        saved_scroll_y = lv_obj_get_scroll_y(panel);
    }
    else {
        lv_obj_scroll_to(panel, saved_scroll_x, saved_scroll_y, LV_ANIM_ON);
    }
}

/***
 * Demonstrate how scrolling appears automatically
 */
void lv_example_scroll_1(void)
{
    /*Create an object with the new style*/
    lv_obj_t * scr;
    scr = lv_screen_active();
    panel = lv_obj_create(scr);
    lv_obj_set_size(panel, 200, 200);
    lv_obj_align(panel, LV_ALIGN_CENTER, 44, 0);

    lv_obj_t * child;
    lv_obj_t * label;

    child = lv_obj_create(panel);
    lv_obj_set_pos(child, 0, 0);
    lv_obj_set_size(child, 70, 70);
    label = lv_label_create(child);
    lv_label_set_text(label, "Zero");
    lv_obj_center(label);

    child = lv_obj_create(panel);
    lv_obj_set_pos(child, 160, 80);
    lv_obj_set_size(child, 80, 80);

    lv_obj_t * child2 = lv_button_create(child);
    lv_obj_set_size(child2, 100, 50);

    label = lv_label_create(child2);
    lv_label_set_text(label, "Right");
    lv_obj_center(label);

    child = lv_obj_create(panel);
    lv_obj_set_pos(child, 40, 160);
    lv_obj_set_size(child, 100, 70);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

label = lv_label_create(child);
lv_label_set_text(label, "Bottom");
lv_obj_center(label);

/* When LV_OBJ_FLAG_SCROLL_ELASTIC is cleared, scrolling does not go past edge boundaries. */
/* lv_obj_remove_flag(panel, LV_OBJ_FLAG_SCROLL_ELASTIC); */

/* Call `scroll_update_cb` while panel is being scrolled. */
lv_obj_add_event_cb(panel, scroll_update_cb, LV_EVENT_SCROLL, NULL);

/* Set up buttons that save and restore scroll position. */
save_button = lv_button_create(scr);
restore_button = lv_button_create(scr);
lv_obj_t * lbl;
lbl = lv_label_create(save_button);
lv_label_set_text_static(lbl, "Save");
lbl = lv_label_create(restore_button);
lv_label_set_text_static(lbl, "Restore");
lv_obj_align_to(save_button, panel, LV_ALIGN_OUT_LEFT_MID, -10, -20);
lv_obj_align_to(restore_button, panel, LV_ALIGN_OUT_LEFT_MID, -10, 20);
lv_obj_add_event_cb(save_button, button_event_cb, LV_EVENT_CLICKED, NULL);
lv_obj_add_event_cb(restore_button, button_event_cb, LV_EVENT_CLICKED, NULL);
}

#endif

```

3.6.2 Aggancio

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static void sw_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * sw = lv_event_get_target_obj(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        lv_obj_t * list = (lv_obj_t *) lv_event_get_user_data(e);

        if(lv_obj_has_state(sw, LV_STATE_CHECKED)) lv_obj_add_flag(list, LV_OBJ_FLAG_SCROLL_ONE);
        else lv_obj_remove_flag(list, LV_OBJ_FLAG_SCROLL_ONE);
    }
}

/***
 * Show an example to scroll snap
 */
void lv_example_scroll_2(void)
{
    lv_obj_t * panel = lv_obj_create(lv_screen_active());
    lv_obj_set_size(panel, 280, 120);
    lv_obj_set_scroll_snap_x(panel, LV_SCROLL_SNAP_CENTER);
    lv_obj_set_flex_flow(panel, LV_FLEX_FLOW_ROW);
    lv_obj_align(panel, LV_ALIGN_CENTER, 0, 20);

    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_obj_t * btn = lv_button_create(panel);
        lv_obj_set_size(btn, 150, lv_pct(100));

        lv_obj_t * label = lv_label_create(btn);
        if(i == 3) {
            lv_label_set_text_fmt(label, "Panel %" LV_PRIu32"\nno snap", i);
            lv_obj_remove_flag(btn, LV_OBJ_FLAG_SNAPPABLE);
        }
        else {
            lv_label_set_text_fmt(label, "Panel %" LV_PRIu32, i);
        }

        lv_obj_center(label);
    }
    lv_obj_update_snap(panel, LV_ANIM_ON);
}

#endif

```

3.6.3 Pulsante mobile

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LIST

static uint32_t btn_cnt = 1;

static void float_button_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * float_btn = lv_event_get_target_obj(e);

    if(code == LV_EVENT_CLICKED) {
        lv_obj_t * list = (lv_obj_t *) lv_event_get_user_data(e);
        char buf[32];
        lv_snprintf(buf, sizeof(buf), "Track %d", (int)btn_cnt);
        lv_obj_t * list_btn = lv_list_add_button(list, LV_SYMBOL_AUDIO, buf);
        btn_cnt++;
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

    /* Move the button to the foreground*/
    lv_obj_move_to_index(float_btn, -1);

    lv_obj_scroll_to_view(list_btn, LV_ANIM_ON);
}

/***
 * Create a list with a floating button
 */
void lv_example_scroll_3(void)
{
    lv_obj_t * list = lv_list_create(lv_screen_active());
    lv_obj_set_size(list, 200, 220);
    lv_obj_center(list);

    for(btn_cnt = 1; btn_cnt <= 2; btn_cnt++) {
        char buf[32];
        lv_snprintf(buf, sizeof(buf), "Track %d", (int)btn_cnt);
        lv_list_add_button(list, LV_SYMBOL_AUDIO, buf);
    }

    lv_obj_t * float_btn = lv_button_create(list);
    lv_obj_set_size(float_btn, 50, 50);
    lv_obj_add_flag(float_btn, LV_OBJ_FLAG_FLOATING);
    lv_obj_align(float_btn, LV_ALIGN_BOTTOM_RIGHT, 0, -lv_obj_get_style_pad_right(list, LV_PART_MAIN));
    lv_obj_add_event_cb(float_btn, float_button_event_cb, LV_EVENT_ALL, list);
    lv_obj_set_style_radius(float_btn, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_bg_image_src(float_btn, LV_SYMBOL_PLUS, 0);
    lv_obj_set_style_text_font(float_btn, lv_theme_get_font_large(float_btn), 0);
}

#endif

```

3.6.4 Stile delle barre di scorrimento

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LIST

/***
 * Styling the scrollbars
 */
void lv_example_scroll_4(void)
{
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_set_size(obj, 200, 100);
    lv_obj_center(obj);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label,
                      "Lorem ipsum dolor sit amet, consectetur adipiscing elit.\n"
                      "Etiam dictum, tortor vestibulum lacinia laoreet, mi neque consectetur neque, vel mattis odio dolor egestas ligula. \n"
                      "Sed vestibulum sapien nulla, id convallis ex porttitor nec. \n"
                      "Duis et massa eu libero accumsan faucibus a in arcu. \n"
                      "Ut pulvinar odio lorem, vel tempus turpis condimentum quis. Nam consectetur condimentum sem in auctor. \n"
                      "Sed nisl augue, venenatis in blandit et, gravida ac tortor. \n"
                      "Etiam dapibus elementum suscipit. \n"
                      "Proin mollis sollicitudin convallis. \n"
                      "Integer dapibus tempus arcu nec viverra. \n"
                      "Donec molestie nulla enim, eu interdum velit placerat quis. \n"
                      "Donec id efficitur risus, at molestie turpis. \n"
                      "Suspendisse vestibulum consectetur nunc ut commodo. \n"
                      "Fusce molestie rhoncus nisi sit amet tincidunt. \n"
                      "Suspendisse a nunc ut magna ornare volutpat.");
}

/*Remove the style of scrollbar to have clean start/
lv_obj_remove_style(obj, NULL, LV_PART_SCROLLBAR | LV_STATE_ANY);

/*Create a transition the animate the some properties on state change/
static const lv_style_prop_t props[] = {LV_STYLE_BG_OPA, LV_STYLE_WIDTH, 0};
static lv_style_transition_dsc_t trans;
lv_style_transition_dsc_init(&trans, props, lv_anim_path_linear, 200, 0, NULL);

/*Create a style for the scrollbars*/
static lv_style_t style;
lv_style_init(&style);
lv_style_set_width(&style, 4);           /*Width of the scrollbar*/
lv_style_set_length(&style, 20);          /*Length of the scrollbar*/
lv_style_set_pad_right(&style, 5);         /*Space from the parallel side*/
lv_style_set_pad_top(&style, 5);          /*Space from the perpendicular side*/

lv_style_set_radius(&style, 2);
lv_style_set_bg_opa(&style, LV_OPA_70);
lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_BLUE));
lv_style_set_border_color(&style, lv_palette_darken(LV_PALETTE_BLUE, 3));
lv_style_set_border_width(&style, 2);
lv_style_set_shadow_width(&style, 8);
lv_style_set_shadow_spread(&style, 2);
lv_style_set_shadow_color(&style, lv_palette_darken(LV_PALETTE_BLUE, 1));

lv_style_set_transition(&style, &trans);

/*Make the scrollbars wider and use 100% opacity when scrolled*/
static lv_style_t style_scrolled;
lv_style_init(&style_scrolled);
lv_style_set_width(&style_scrolled, 8);
lv_style_set_bg_opa(&style_scrolled, LV_OPA_COVER);

lv_obj_add_style(obj, &style, LV_PART_SCROLLBAR);
lv_obj_add_style(obj, &style_scrolled, LV_PART_SCROLLBAR | LV_STATE_SCROLLED);
}

#endif

```

3.6.5 Scorrimento da destra a sinistra

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_FONT_DEJAVU_16_PERSIAN_HEBREW

/***
 * Scrolling with Right To Left base direction
 ***/
void lv_example_scroll_5(void)
{
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_set_style_base_dir(obj, LV_BASE_DIR_RTL, 0);
    lv_obj_set_size(obj, 200, 100);
    lv_obj_center(obj);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label,
        "دیجور و دورو ی امپربو رمیان . ROM) دیندا و خطاو ' مظاوح و RAM) یقدامه سیرتسد ' مظاوح یاراد هک سمرا مدزابریزیار و اونوگ (Microcontroller: میکروکنترولر ) رلوگن (Serial Port میکروکنترولر باکرد و 0) دیجور و دورو . (ایپرس درو و نسخه های دیگر دارج )";
    lv_obj_set_width(label, 400);
    lv_obj_set_style_text_font(label, &lv_font_dejavu_16_persian_hebrew, 0);

}
#endif

```

3.6.6 Traslazione durante lo scorrimento

```

#include "../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES && LV_USE_FLEX
static void scroll_event_cb(lv_event_t * e)
{
    lv_obj_t * cont = lv_event_get_target_obj(e);

    lv_area_t cont_a;
    lv_obj_get_coords(cont, &cont_a);
    int32_t cont_y_center = cont_a.y1 + lv_area_get_height(&cont_a) / 2;

    int32_t r = lv_obj_get_height(cont) * 7 / 10;
    int32_t i;
    int32_t child_cnt = (int32_t)lv_obj_get_child_count(cont);
    for(i = 0; i < child_cnt; i++) {
        lv_obj_t * child = lv_obj_get_child(cont, i);
        lv_area_t child_a;
        lv_obj_get_coords(child, &child_a);

        int32_t child_y_center = child_a.y1 + lv_area_get_height(&child_a) / 2;

        int32_t diff_y = child_y_center - cont_y_center;
        diff_y = LV_ABS(diff_y);

        /*Get the x of diff_y on a circle.*/
        int32_t x;
        /*If diff_y is out of the circle use the last point of the circle (the radius)*/
        if(diff_y >= r) {
            x = r;
        }
        else {
            /*Use Pythagoras theorem to get x from radius and y*/
            uint32_t x_sqr = r * r - diff_y * diff_y;
            lv_sqrt_res_t res;
            lv_sqrt(x_sqr, &res, 0x8000); /*Use lvgl's built in sqrt root function*/
            x = r - res.i;
        }
    }

    /*Translate the item by the calculated X coordinate*/
    lv_obj_set_style_translate_x(child, x, 0);

    /*Use some opacity with larger translations*/
    lv_opa_t opa = (lv_opa_t)lv_map(x, 0, r, LV_OPA_TRANS, LV_OPA_COVER);
    lv_obj_set_style_opa(child, LV_OPA_COVER - opa, 0);
}
}

/*
 * Translate the object as they scroll
 */
void lv_example_scroll_6(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 200, 200);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLOW_COLUMN);
    lv_obj_add_event_cb(cont, scroll_event_cb, LV_EVENT_SCROLL, NULL);
    lv_obj_set_style_radius(cont, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_clip_corner(cont, true, 0);
    lv_obj_set_scroll_dir(cont, LV_DIR_VER);
    lv_obj_set_scroll_snap_y(cont, LV_SCROLL_SNAP_CENTER);
    lv_obj_set_scrollbar_mode(cont, LV_SCROLLBAR_MODE_OFF);

    uint32_t i;
    for(i = 0; i < 20; i++) {
        lv_obj_t * btn = lv_button_create(cont);
        lv_obj_set_width(btn, lv_pct(100));

        lv_obj_t * label = lv_label_create(btn);
        lv_label_set_text_fmt(label, "Button %s LV_PRIu32, %i");
    }

    /*Update the buttons position manually for first*/
    lv_obj_send_event(cont, LV_EVENT_SCROLL, NULL);

    /*Be sure the fist button is in the middle*/
    lv_obj_scroll_to_view(lv_obj_get_child(cont, 0), LV_ANIM_OFF);
}

#endif

```

3.6.7 Scrolling infinito

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static lv_obj_t * high_label;
static lv_obj_t * low_label;
static int32_t top_num;
static int32_t bottom_num;
static bool update_scroll_running = false;

static lv_obj_t * load_item(lv_obj_t * parent, int32_t num)
{
    lv_obj_t * obj = (lv_obj_t *) lv_obj_create(parent);
    lv_obj_set_size(obj, LV_PCT(100), LV_SIZE_CONTENT);
    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text_fmt(label, "%" PRId32, num);
    return obj;
}

static void update_scroll(lv_obj_t * obj)
{
    /* do not re-enter this function when `lv_obj_scroll_by` triggers this callback again.
     */
    if(update_scroll_running) return;
    update_scroll_running = true;

    int32_t top_num_original = top_num;
    int32_t bottom_num_original = bottom_num;

    /* load items we're getting close to */
    while(bottom_num > -30 && lv_obj_get_scroll_bottom(obj) < 200) {
        bottom_num -= 1;
        load_item(obj, bottom_num);
        lv_obj_update_layout(obj);
        LV_LOG_USER("loaded bottom num: %" PRId32, bottom_num);
    }

    while(top_num < 30 && lv_obj_get_scroll_top(obj) < 200) {
        top_num += 1;
        int32_t bottom_before = lv_obj_get_scroll_bottom(obj);
        lv_obj_t * new_item = load_item(obj, top_num);
        lv_obj_move_to_index(new_item, 0);
        lv_obj_update_layout(obj);
        int32_t bottom_after = lv_obj_get_scroll_bottom(obj);
        lv_obj_scroll_by(obj, 0, bottom_before - bottom_after, LV_ANIM_OFF);
        LV_LOG_USER("loaded top num: %" PRId32, top_num);
    }

    /* delete far-away items */
    while(lv_obj_get_scroll_bottom(obj) > 600) {
        bottom_num += 1;
        lv_obj_t * child = lv_obj_get_child(obj, -1);
        lv_obj_delete(child);
        lv_obj_update_layout(obj);
        LV_LOG_USER("deleted bottom num: %" PRId32, bottom_num);
    }

    while(lv_obj_get_scroll_top(obj) > 600) {
        top_num -= 1;
        int32_t bottom_before = lv_obj_get_scroll_bottom(obj);
        lv_obj_t * child = lv_obj_get_child(obj, 0);
        lv_obj_delete(child);
        lv_obj_update_layout(obj);
        int32_t bottom_after = lv_obj_get_scroll_bottom(obj);
        lv_obj_scroll_by(obj, 0, bottom_before - bottom_after, LV_ANIM_OFF);
        LV_LOG_USER("deleted top num: %" PRId32, top_num);
    }

    if(top_num != top_num_original) {
        lv_label_set_text_fmt(high_label, "current largest\nloaded value:\n%" PRId32, top_num);
    }
    if(bottom_num != bottom_num_original) {
        lv_label_set_text_fmt(low_label, "current smallest\nloaded value:\n%" PRId32, bottom_num);
    }
}

update_scroll_running = false;
}

static void scroll_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    update_scroll(obj);
}

static void checkbox_cb(lv_event_t * e)
{
    lv_obj_t * checkbox = lv_event_get_target_obj(e);
    lv_obj_t * obj = (lv_obj_t *) lv_event_get_user_data(e);
    bool checked = lv_obj_has_state(checkbox, LV_STATE_CHECKED);
    lv_obj_set_style_opa(obj, checked ? LV_OPA_COVER : LV_OPA_TRANS, LV_PART_SCROLLBAR);
}

/***
 * Dynamically load widgets while scrolling
 */
void lv_example_scroll_7(void)
{
    lv_obj_t * scr = lv_screen_active();
    lv_obj_t * obj = lv_obj_create(scr);
    lv_obj_set_size(obj, 160, 220);
    lv_obj_align(obj, LV_ALIGN_RIGHT_MID, -10, 0);
    lv_obj_set_flex_flow(obj, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_style_opa(obj, LV_OPA_TRANS, LV_PART_SCROLLBAR);

    high_label = lv_label_create(scr);
    lv_label_set_text_static(high_label, "current largest\nloaded value:");
    lv_obj_align(high_label, LV_ALIGN_TOP_LEFT, 10, 10);

    lv_obj_t * checkbox = lv_checkbox_create(scr);
    lv_checkbox_set_text_static(checkbox, "show\nscrollbar");
    lv_obj_align(checkbox, LV_ALIGN_LEFT_MID, 10, 0);
    lv_obj_add_event_cb(checkbox, checkbox_cb, LV_EVENT_VALUE_CHANGED, obj);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

low_label = lv_label_create(scr);
lv_label_set_text_static(low_label, "current smallest\nloaded value:");
lv_obj_align(low_label, LV_ALIGN_BOTTOM_LEFT, 10, -10);

load_item(obj, 3);
/* These counters hold the highest/lowest number currently loaded. */
top_num = 3;
bottom_num = 3;

lv_obj_update_layout(obj);
update_scroll(obj);
lv_obj_add_event_cb(obj, scroll_cb, LV_EVENT_SCROLL, NULL);
}

#endif

```

3.6.8 Scrolling circolare

```

#include "../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES && LV_USE_FLEX

/*
 * Circular list implementation based on:
 * Reference: https://blog.csdn.net/TQW4321/article/details/145434819
 */

/* Applies to both width (for row items) and height (for column items) */
#define ITEM_SIZE     80

/* Function to calculate the total content width of the container */
static int32_t get_content_width(lv_obj_t * cont)
{
    int32_t w = 0;
    int32_t pad_column = lv_obj_get_style_pad_column(cont, LV_PART_MAIN);
    int32_t total = (int32_t)lv_obj_get_child_count(cont);

    for(int32_t i = 0; i < total; i++) {
        w += lv_obj_get_width(lv_obj_get_child(cont, i));
        if(i < total - 1) w += pad_column;
    }
    return w + lv_obj_get_style_pad_left(cont, LV_PART_MAIN)
        + lv_obj_get_style_pad_right(cont, LV_PART_MAIN);
}

/* Scroll event callback for row layout */
static void cont_row_scroll_event_cb(lv_event_t * e)
{
    static bool is_adjusting = false;
    lv_obj_t * cont = lv_event_get_current_target_obj(e);

    if(!is_adjusting) {
        is_adjusting = true;
        int32_t scroll_x = lv_obj_get_scroll_x(cont);
        int32_t cont_w = lv_obj_get_width(cont);
        int32_t content_w = get_content_width(cont);

        /* Use ITEM_SIZE as horizontal item width */
        const int32_t item_width = ITEM_SIZE;

        if(scroll_x <= 0) {
            lv_obj_t * last_child = lv_obj_get_child(cont, (int32_t)(lv_obj_get_child_count(cont) - 1));
            lv_obj_move_to_index(last_child, 0);
            lv_obj_scroll_to_x(cont, scroll_x + item_width, LV_ANIM_OFF);
        }
        else if(scroll_x > content_w - cont_w) {
            lv_obj_t * first_child = lv_obj_get_child(cont, 0);
            lv_obj_move_to_index(first_child, (int32_t)(lv_obj_get_child_count(cont) - 1));
            lv_obj_scroll_to_x(cont, scroll_x - item_width, LV_ANIM_OFF);
        }
        is_adjusting = false;
    }
}

/* Function to calculate the total content height of the container */
static int32_t get_content_height(lv_obj_t * cont)
{
    int32_t h = 0;
    int32_t pad_row = lv_obj_get_style_pad_row(cont, LV_PART_MAIN);
    int32_t total = (int32_t)lv_obj_get_child_count(cont);

    for(int i = 0; i < total; i++) {
        h += lv_obj_get_height(lv_obj_get_child(cont, i));
        if(i < total - 1) h += pad_row;
    }
    return h + lv_obj_get_style_pad_top(cont, LV_PART_MAIN)
        + lv_obj_get_style_pad_bottom(cont, LV_PART_MAIN);
}

/* Scroll event callback for column layout */
static void cont_col_scroll_event_cb(lv_event_t * e)
{
    static bool is_adjusting = false;
    lv_obj_t * cont = lv_event_get_current_target_obj(e);

    if(!is_adjusting) {
        is_adjusting = true;
        int32_t scroll_y = lv_obj_get_scroll_y(cont);
        int32_t cont_h = lv_obj_get_height(cont);
        int32_t content_h = get_content_height(cont);

        /* Use ITEM_SIZE as vertical item height */
        const int32_t item_height = ITEM_SIZE;

        if(scroll_y <= 0) {
            lv_obj_t * last_child = lv_obj_get_child(cont, (int32_t)(lv_obj_get_child_count(cont) - 1));
            lv_obj_move_to_index(last_child, 0);
            lv_obj_scroll_to_y(cont, scroll_y + item_height, LV_ANIM_OFF);
        }
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

    }
    else if(scroll_y > content_h - cont_h) {
        lv_obj_t * first_child = lv_obj_get_child(cont, 0);
        lv_obj_move_to_index(first_child, (int32_t)(lv_obj_get_child_count(cont) - 1));
        lv_obj_scroll_to_y(cont, scroll_y - item_height, LV_ANIM_OFF);
    }
    is_adjusting = false;
}

void lv_example_scroll_8(void)
{
    /* Create a scroll container with ROW flex direction */
    lv_obj_t * cont_row = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont_row, 300, 75);
    lv_obj_align(cont_row, LV_ALIGN_TOP_MID, 0, 5);
    lv_obj_set_flex_flow(cont_row, LV_FLEX_FLOW_ROW);
    lv_obj_add_event_cb(cont_row, cont_row_scroll_event_cb, LV_EVENT_SCROLL, NULL);
    /* Hide scrollbar visuals */
    lv_obj_set_scrollbar_mode(cont_row, LV_SCROLLBAR_MODE_OFF);

    /* Create a scroll container with COLUMN flex direction */
    lv_obj_t * cont_col = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont_col, 200, 150);
    lv_obj_align_to(cont_col, cont_row, LV_ALIGN_OUT_BOTTOM_MID, 0, 5);
    lv_obj_set_flex_flow(cont_col, LV_FLEX_FLOW_COLUMN);
    lv_obj_add_event_cb(cont_col, cont_col_scroll_event_cb, LV_EVENT_SCROLL, NULL);
    /* Hide scrollbar visuals */
    lv_obj_set_scrollbar_mode(cont_col, LV_SCROLLBAR_MODE_OFF);

    /* If the number of items is less than 3, scrolling may not be needed or may cause unexpected behavior. */
    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_obj_t * obj;
        lv_obj_t * label;

        /*Add items to the row*/
        obj = lv_button_create(cont_row);
        lv_obj_set_size(obj, ITEM_SIZE, LV_PCT(100));

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item %" LV_PRIu32, i + 1);
        lv_obj_center(label);

        /*Add items to the column*/
        obj = lv_button_create(cont_col);
        lv_obj_set_size(obj, LV_PCT(100), ITEM_SIZE);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item %" LV_PRIu32, i + 1);
        lv_obj_center(label);
    }
}

#endif

```

3.6.9 Proprietà dello Scrolling

```

/***
 * @file lv_example_scroll_9.c
 */

#include "lv_example_scroll.h"

#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static lv_obj_t * switch_create(lv_obj_t * parent, const char * title, lv_obj_flag_t flag, bool en);
static void generic_switch_event_cb(lv_event_t * e);

static lv_obj_t * list;

void lv_example_scroll_9(void)
{
    lv_obj_t * panel = lv_obj_create(lv_screen_active());
    lv_obj_set_style_shadow_width(panel, 16, 0);
    lv_obj_set_style_shadow_offset_y(panel, 8, 0);
    lv_obj_set_style_shadow_offset_x(panel, 4, 0);
    lv_obj_set_style_shadow_opa(panel, LV_OPA_40, 0);

    lv_obj_set_size(panel, lv_pct(70), lv_pct(90));
    lv_obj_center(panel);
    lv_obj_set_flex_flow(panel, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(panel, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    list = lv_list_create(panel);
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image1.png");
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image2.png");
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image3.png");
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image4.png");
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image5.png");
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image6.png");
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image7.png");
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image8.png");
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image9.png");
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image10.png");

    switch_create(panel, "Scollable", LV_OBJ_FLAG_SCROLLABLE, true);
    switch_create(panel, "Scroll chain", LV_OBJ_FLAG_SCROLL_CHAIN, true);
    switch_create(panel, "Elastic scroll", LV_OBJ_FLAG_SCROLL_ELASTIC, true);
    switch_create(panel, "Add scroll momentum", LV_OBJ_FLAG_SCROLL_MOMENTUM, true);

    /*Show the switches first*/
    lv_obj_move_to_index(list, -1);
}

static lv_obj_t * switch_create(lv_obj_t * parent, const char * title, lv_obj_flag_t flag, bool en)
{

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * cont = lv_obj_create(parent);
lv_obj_remove_style_all(cont);
lv_obj_set_size(cont, lv_pct(100), LV_SIZE_CONTENT);
lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW);
lv_obj_set_flex_align(cont, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_START);

lv_obj_t * label = lv_label_create(cont);
lv_label_set_text(label, title);
lv_obj_set_flex_grow(label, 1);

lv_obj_t * sw = lv_switch_create(cont);
lv_obj_add_event_cb(sw, generic_switch_event_cb, LV_EVENT_VALUE_CHANGED, (void *)((lv_uintptr_t) flag));
if(en)
{
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_flag(list, flag);
}
else
{
    lv_obj_remove_state(sw, LV_STATE_CHECKED);
    lv_obj_remove_flag(list, flag);
}

return cont;
}

static void generic_switch_event_cb(lv_event_t * e)
{
    lv_obj_t * sw = lv_event_get_target_obj(e);
    lv_obj_flag_t flag = (lv_obj_flag_t)((lv_uintptr_t)lv_event_get_user_data(e));

    if(lv_obj_has_state(sw, LV_STATE_CHECKED)) lv_obj_add_flag(list, flag);
    else lv_obj_remove_flag(list, flag);
}

#endif

```

3.7 | Widget

3.7.1 Widget Base

Oggetti base con stili personalizzati

```

#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES

void lv_example_obj_1(void)
{
    lv_obj_t * obj1;
    obj1 = lv_obj_create(lv_screen_active());
    lv_obj_set_size(obj1, 100, 50);
    lv_obj_align(obj1, LV_ALIGN_CENTER, -60, -30);

    static lv_style_t style_shadow;
    lv_style_init(&style_shadow);
    lv_style_set_shadow_width(&style_shadow, 10);
    lv_style_set_shadow_spread(&style_shadow, 5);
    lv_style_set_shadow_color(&style_shadow, lv_palette_main(LV_PALETTE_BLUE));

    lv_obj_t * obj2;
    obj2 = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj2, &style_shadow, 0);
    lv_obj_align(obj2, LV_ALIGN_CENTER, 60, 30);
}
#endif

```

Creare un oggetto trascinabile

```

#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES

static void drag_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);

    lv_indev_t * indev = lv_indev_active();
    if(indev == NULL) return;

    lv_point_t vect;
    lv_indev_get_vect(indev, &vect);

    int32_t x = lv_obj_get_x_aligned(obj) + vect.x;
    int32_t y = lv_obj_get_y_aligned(obj) + vect.y;
    lv_obj_set_pos(obj, x, y);
}

/***
 * Make an object draggable.
 */
void lv_example_obj_2(void)
{
    lv_obj_t * obj;
    obj = lv_obj_create(lv_screen_active());
    lv_obj_set_size(obj, 150, 100);
    lv_obj_add_event_cb(obj, drag_event_handler, LV_EVENT_PRESSING, NULL);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label, "Drag me");
    lv_obj_center(label);
}
#endif

```

Trasformare un oggetto usando una matrice 3x3

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#define LV_DRAW_TRANSFORM_USE_MATRIX

static void timer_cb(lv_timer_t * timer)
{
    lv_obj_t * obj = (lv_obj_t *) lv_timer_get_user_data(timer);

    static float value = 0.1f;
    lv_matrix_t matrix;
    lv_matrix_identity(&matrix);
    lv_matrix_scale(&matrix, value, 1);
    lv_matrix_rotate(&matrix, value * 360);
    lv_obj_set_transform(obj, &matrix);

    value += 0.01f;

    if(value > 2.0f) {
        lv_obj_reset_transform(obj);
        value = 0.1f;
    }
}

void lv_example_obj_3(void)
{
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_center(obj);

    lv_timer_create(timer_cb, 20, obj);
}

#else

void lv_example_obj_3(void)
{
    lv_label_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text_static(label, "LV_DRAW_TRANSFORM_USE_MATRIX is not enabled");
    lv_obj_center(label);
}

#endif /*LV_DRAW_TRANSFORM_USE_MATRIX*/
#endif /*LV_BUILD_EXAMPLES*/
```

3.7.2 Immagine animata

Immagine animata semplice

```
#include "../../lv_examples.h"
#if LV_USE_ANIMIMG && LV_BUILD_EXAMPLES
LV_IMAGE_DECLARE(animimg001);
LV_IMAGE_DECLARE(animimg002);
LV_IMAGE_DECLARE(animimg003);

static const lv_image_dsc_t * anim_imgs[3] = {
    &animimg001,
    &animimg002,
    &animimg003,
};

void lv_example_animimg_1(void)
{
    lv_obj_t * animimg0 = lv_animimg_create(lv_screen_active());
    lv_obj_center(animimg0);
    lv_animimg_set_src(animimg0, (const void **) anim_imgs, 3);
    lv_animimg_set_duration(animimg0, 1000);
    lv_animimg_set_repeat_count(animimg0, LV_ANIM_REPEAT_INFINITE);
    lv_animimg_start(animimg0);
}

#endif
```

3.7.3 Arco

Arco semplice

```
#include "../../lv_examples.h"
#if LV_USE_ARC && LV_BUILD_EXAMPLES

static void value_changed_event_cb(lv_event_t * e);

void lv_example_arc_1(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());

    /*Create an Arc*/
    lv_obj_t * arc = lv_arc_create(lv_screen_active());
    lv_obj_set_size(arc, 150, 150);
    lv_arc_set_rotation(arc, 135);
    lv_arc_set_bg_angles(arc, 0, 270);
    lv_arc_set_value(arc, 10);
    lv_obj_center(arc);
    lv_obj_add_event_cb(arc, value_changed_event_cb, LV_EVENT_VALUE_CHANGED, label);

    /*Manually update the label for the first time*/
    lv_obj_send_event(arc, LV_EVENT_VALUE_CHANGED, NULL);
}

static void value_changed_event_cb(lv_event_t * e)
```

(continues on next page)

(continua dalla pagina precedente)

```
{
    lv_obj_t * arc = lv_event_get_target_obj(e);
    lv_obj_t * label = (lv_obj_t *)lv_event_get_user_data(e);

    lv_label_set_text_fmt(label, "% LV_PRId32 %%", lv_arc_get_value(arc));

    /*Rotate the label to the current position of the arc*/
    lv_arc_rotate_obj_to_angle(arc, label, 25);
}

#endif
```

Loader con Arc

```
#include "../../lv_examples.h"

#if LV_USE_ARC && LV_BUILD_EXAMPLES

static void set_angle(void * obj, int32_t v)
{
    lv_arc_set_value((lv_obj_t *)obj, v);
}

/***
 * Create an arc which acts as a loader.
 */
void lv_example_arc_2(void)
{
    /*Create an Arc*/
    lv_obj_t * arc = lv_arc_create(lv_screen_active());
    lv_arc_set_rotation(arc, 270);
    lv_arc_set_bg_angles(arc, 0, 360);
    lv_obj_remove_style(arc, NULL, LV_PART_KNOB); /*Be sure the knob is not displayed*/
    lv_obj_remove_flag(arc, LV_OBJ_FLAG_CLICKABLE); /*To not allow adjusting by click*/
    lv_obj_center(arc);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, arc);
    lv_anim_set_exec_cb(&a, set_angle);
    lv_anim_set_duration(&a, 1000);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE); /*Just for the demo*/
    lv_anim_set_repeat_delay(&a, 500);
    lv_anim_set_values(&a, 0, 100);
    lv_anim_start(&a);
}

#endif
```

Grafico a torta con sezioni cliccabili tramite Archi

```
#include "../../lv_examples.h"

#if LV_USE_ARC && LV_BUILD_EXAMPLES

#define CHART_SIZE 150
#define SLICE_OFFSET 20

typedef struct {
    int start_angle;
    int end_angle;
    int mid_angle;
    lv_point_t home;
    bool out;
} slice_info_t;

typedef struct {
    lv_obj_t * obj;
    int start_x;
    int start_y;
    int end_x;
    int end_y;
} slice_anim_data_t;

static float angle_accum = 0.0f;
static slice_info_t * active_info = NULL;
static lv_obj_t * active_arc = NULL;

static void anim_move_cb(void * var, int32_t v)
{
    slice_anim_data_t * d = (slice_anim_data_t *) var;

    int32_t x = d->start_x + ((d->end_x - d->start_x) * v) / 100;
    int32_t y = d->start_y + ((d->end_y - d->start_y) * v) / 100;
    lv_obj_set_pos(d->obj, x, y);
}

static void anim_cleanup_cb(lv_anim_t * a)
{
    lv_free(a->var);
}

static void arc_click_cb(lv_event_t * e)
{
    lv_obj_t * arc = lv_event_get_target_obj(e);
    slice_info_t * info = (slice_info_t *)lv_event_get_user_data(e);

    int32_t x_off = (SLICE_OFFSET * lv_trigo_cos(info->mid_angle)) >> LV_TRIGO_SHIFT;
    int32_t y_off = (SLICE_OFFSET * lv_trigo_sin(info->mid_angle)) >> LV_TRIGO_SHIFT;

    if(active_info && active_info != info && active_info->out) {
        slice_anim_data_t * anim_back = (slice_anim_data_t *) lv_malloc(sizeof(slice_anim_data_t));
        anim_back->obj = active_arc;
```

(continues on next page)

(continua dalla pagina precedente)

```

anim_back->start_x = lv_obj_get_x(active_arc) - SLICE_OFFSET;
anim_back->start_y = lv_obj_get_y(active_arc) - SLICE_OFFSET;
anim_back->end_x = active_info->home.x;
anim_back->end_y = active_info->home.y;

active_info->out = false;

lv_anim_t a;
lv_anim_init(&a);
lv_anim_set_var(&a, anim_back);
lv_anim_set_exec_cb(&a, anim_move_cb);
lv_anim_set_duration(&a, 200);
lv_anim_set_values(&a, 0, 100);
lv_anim_set_deleted_cb(&a, anim_cleanup_cb);
lv_anim_start(&a);

}

int target_x, target_y;
if(info->out) {
    target_x = info->home.x;
    target_y = info->home.y;
    info->out = false;
    active_info = NULL;
    active_arc = NULL;
}
else {
    target_x = info->home.x + x_off;
    target_y = info->home.y + y_off;
    info->out = true;
    active_info = info;
    active_arc = arc;
}

slice_anim_data_t * anim_data = (slice_anim_data_t *) lv_malloc(sizeof(slice_anim_data_t));
anim_data->obj = arc;
anim_data->start_x = lv_obj_get_x(arc) - SLICE_OFFSET;
anim_data->start_y = lv_obj_get_y(arc) - SLICE_OFFSET;
anim_data->end_x = target_x;
anim_data->end_y = target_y;

lv_anim_t a;
lv_anim_init(&a);
lv_anim_set_var(&a, anim_data);
lv_anim_set_exec_cb(&a, anim_move_cb);
lv_anim_set_duration(&a, 200);
lv_anim_set_values(&a, 0, 100);
lv_anim_set_deleted_cb(&a, anim_cleanup_cb);
lv_anim_start(&a);
}

static void create_slice(lv_obj_t * parent, int percentage, lv_color_t color)
{
    if(percentage <= 0) return;

    float slice_angle = (percentage * 360.0f) / 100.0f;
    int start = (int)(angle_accum + 0.5f);
    angle_accum += slice_angle;
    int end = (int)(angle_accum + 0.5f);
    if(end > 360) end = 360;

    lv_obj_t * arc = lv_arc_create(parent);
    lv_obj_set_size(arc, CHART_SIZE, CHART_SIZE);
    lv_obj_center(arc);

    lv_arc_set_mode(arc, LV_ARC_MODE_NORMAL);
    lv_arc_set_bg_start_angle(arc, start);
    lv_arc_set_bg_end_angle(arc, end);

    lv_obj_set_style_arc_width(arc, CHART_SIZE / 2, LV_PART_MAIN);
    lv_obj_set_style_arc_width(arc, 0, LV_PART_INDICATOR);

    lv_obj_set_style_arc_color(arc, color, LV_PART_MAIN);
    lv_obj_set_style_arc_rounded(arc, false, LV_PART_MAIN);
    lv_obj_remove_style(arc, NULL, LV_PART_KNOB);
    lv_obj_add_flag(arc, LV_OBJ_FLAG_ADV_HITTEST);

    lv_obj_t * label = lv_label_create(arc);
    lv_label_set_text_fmt(label, "%'", percentage);
    int mid_angle = start + ((end - start) / 2);
    int radius = CHART_SIZE / 4;
    int x_offset = (radius * lv_trigo_cos(mid_angle)) >> LV_TRIGO_SHIFT;
    int y_offset = (radius * lv_trigo_sin(mid_angle)) >> LV_TRIGO_SHIFT;

    lv_obj_align(label, LV_ALIGN_CENTER, x_offset, y_offset);

    slice_info_t * info = (slice_info_t *) lv_malloc(sizeof(slice_info_t));
    info->start_angle = start;
    info->end_angle = end;
    info->mid_angle = mid_angle;
    info->out = false;
    info->home.x = lv_obj_get_x(arc);
    info->home.y = lv_obj_get_y(arc);
    lv_obj_add_event_cb(arc, arc_click_cb, LV_EVENT_CLICKED, info);
}

void lv_example_arc_3(void)
{
    /* Root container: flex row */
    lv_obj_t * root = lv_obj_create(lv_screen_active());
    lv_obj_set_size(root, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_center(root);
    lv_obj_set_flex_flow(root, LV_FLEX_FLOW_ROW);
    lv_obj_set_flex_align(root, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    lv_obj_set_style_pad_all(root, 0, LV_PART_MAIN);
    lv_obj_set_style_border_width(root, 0, LV_PART_MAIN);
    lv_obj_set_style_border_color(root, lv_palette_main(LV_PALETTE_RED), LV_PART_MAIN);
    lv_obj_set_style_bg_opa(root, LV_OPA_TRANSN, LV_PART_MAIN);
    lv_obj_remove_flag(root, LV_OBJ_FLAG_SCROLLABLE);

    /* Slices container */
    lv_obj_t * slices_container = lv_obj_create(root);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_size(slices_container, CHART_SIZE + 2 * SLICE_OFFSET, CHART_SIZE + 2 * SLICE_OFFSET);
lv_obj_set_style_pad_all(slices_container, 0, LV_PART_MAIN);
lv_obj_set_style_margin_all(slices_container, 0, LV_PART_MAIN);
lv_obj_set_style_border_width(slices_container, 0, LV_PART_MAIN);
lv_obj_set_style_border_color(slices_container, lv_palette_main(LV_PALETTE_BLUE), LV_PART_MAIN);
lv_obj_set_style_bg_opa(slices_container, LV_OPA_TRANSP, LV_PART_MAIN);
lv_obj_remove_flag(slices_container, LV_OBJ_FLAG_SCROLLABLE);

/* Create slices */
angle_accum = 0.0f;
create_slice(slices_container, 12, lv_palette_main(LV_PALETTE_RED));
create_slice(slices_container, 18, lv_palette_main(LV_PALETTE_BLUE));
create_slice(slices_container, 26, lv_palette_main(LV_PALETTE_GREEN));
create_slice(slices_container, 24, lv_palette_main(LV_PALETTE_ORANGE));
create_slice(slices_container, 20, lv_palette_main(LV_PALETTE_BLUE_GREY));
}

#endif

```

3.7.4 Barra

Barra semplice

```

#include "../../lv_examples.h"
#ifndef LV_USE_BAR && LV_BUILD_EXAMPLES

void lv_example_bar_1(void)
{
    lv_obj_t * bar1 = lv_bar_create(lv_screen_active());
    lv_obj_set_size(bar1, 200, 20);
    lv_obj_center(bar1);
    lv_bar_set_value(bar1, 70, LV_ANIM_OFF);
}

#endif

```

Stile di una barra

```

#include "../../lv_examples.h"
#ifndef LV_USE_BAR && LV_BUILD_EXAMPLES

/***
 * Example of styling the bar
 */
void lv_example_bar_2(void)
{
    static lv_style_t style_bg;
    static lv_style_t style_indic;

    lv_style_init(&style_bg);
    lv_style_set_border_color(&style_bg, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_border_width(&style_bg, 2);
    lv_style_set_pad_all(&style_bg, 6); /*To make the indicator smaller*/
    lv_style_set_radius(&style_bg, 6);
    lv_style_set_anim_duration(&style_bg, 1000);

    lv_style_init(&style_indic);
    lv_style_set_bg_opa(&style_indic, LV_OPA_COVER);
    lv_style_set_bg_color(&style_indic, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_radius(&style_indic, 3);

    lv_obj_t * bar = lv_bar_create(lv_screen_active());
    lv_obj_remove_style_all(bar); /*To have a clean start*/
    lv_obj_add_style(bar, &style_bg, 0);
    lv_obj_add_style(bar, &style_indic, LV_PART_INDICATOR);

    lv_obj_set_size(bar, 200, 20);
    lv_obj_center(bar);
    lv_bar_set_value(bar, 100, LV_ANIM_ON);
}

#endif

```

Indicatore di temperatura

```

#include "../../lv_examples.h"
#ifndef LV_USE_BAR && LV_BUILD_EXAMPLES

static void set_temp(void * bar, int32_t temp)
{
    lv_bar_set_value((lv_obj_t *)bar, temp, LV_ANIM_ON);
}

/***
 * A temperature meter example
 */
void lv_example_bar_3(void)
{
    static lv_style_t style_indic;

    lv_style_init(&style_indic);
    lv_style_set_bg_opa(&style_indic, LV_OPA_COVER);
    lv_style_set_bg_color(&style_indic, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_bg_grad_color(&style_indic, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_bg_grad_dir(&style_indic, LV_GRAD_DIR_VER);

    lv_obj_t * bar = lv_bar_create(lv_screen_active());
    lv_obj_add_style(bar, &style_indic, LV_PART_INDICATOR);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_size(bar, 20, 200);
lv_obj_center(bar);
lv_bar_set_range(bar, -20, 40);

lv_anim_t a;
lv_anim_init(&a);
lv_anim_set_exec_cb(&a, set_temp);
lv_anim_set_duration(&a, 3000);
lv_anim_set_reverse_duration(&a, 3000);
lv_anim_set_var(&a, bar);
lv_anim_set_values(&a, -20, 40);
lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
lv_anim_start(&a);
}

#endif

```

Motivo a strisce e valore dell'intervallo

```

#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/** 
 * Bar with stripe pattern and ranged value
 */
void lv_example_bar_4(void)
{
    LV_IMAGE_DECLARE(img_skew_strip);
    static lv_style_t style_indic;

    lv_style_init(&style_indic);
    lv_style_set_bg_image_src(&style_indic, &img_skew_strip);
    lv_style_set_bg_image_tiled(&style_indic, true);
    lv_style_set_bg_image_opa(&style_indic, LV_OPA_30);

    lv_obj_t * bar = lv_bar_create(lv_screen_active());
    lv_obj_add_style(bar, &style_indic, LV_PART_INDICATOR);

    lv_obj_set_size(bar, 260, 20);
    lv_obj_center(bar);
    lv_bar_set_mode(bar, LV_BAR_MODE_RANGE);
    lv_bar_set_value(bar, 90, LV_ANIM_OFF);
    lv_bar_set_start_value(bar, 20, LV_ANIM_OFF);
}
#endif

```

Barra con direzione di base LTR e RTL

```

#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/** 
 * Bar with LTR and RTL base direction
 */
void lv_example_bar_5(void)
{
    lv_obj_t * label;

    lv_obj_t * bar_ltr = lv_bar_create(lv_screen_active());
    lv_obj_set_size(bar_ltr, 200, 20);
    lv_bar_set_value(bar_ltr, 70, LV_ANIM_OFF);
    lv_obj_align(bar_ltr, LV_ALIGN_CENTER, 0, -30);

    label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Left to Right base direction");
    lv_obj_align_to(label, bar_ltr, LV_ALIGN_OUT_TOP_MID, 0, -5);

    lv_obj_t * bar rtl = lv_bar_create(lv_screen_active());
    lv_obj_set_style_base_dir(bar_rtl, LV_BASE_DIR_RTL, 0);
    lv_obj_set_size(bar_rtl, 200, 20);
    lv_bar_set_value(bar_rtl, 70, LV_ANIM_OFF);
    lv_obj_align(bar_rtl, LV_ALIGN_CENTER, 0, 30);

    label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Right to Left base direction");
    lv_obj_align_to(label, bar_rtl, LV_ALIGN_OUT_TOP_MID, 0, -5);
}

#endif

```

"Drawer" personalizzato per visualizzare il valore corrente

```

#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

#define MAX_VALUE 100
#define MIN_VALUE 0

static void set_value(void * bar, int32_t v)
{
    lv_bar_set_value((lv_obj_t *)bar, v, LV_ANIM_OFF);
}

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);

    lv_draw_label_desc_t label_desc;
    lv_draw_label_desc_init(&label_desc);

```

(continues on next page)

(continua dalla pagina precedente)

```

label_dsc.font = LV_FONT_DEFAULT;

char buf[8];
lv_snprintf(buf, sizeof(buf), "%d", (int)lv_bar_get_value(obj));

lv_point_t txt_size;
lv_text_get_size(&txt_size, buf, label_dsc.font, label_dsc.letter_space, label_dsc.line_space, LV_COORD_MAX,
                 label_dsc.flag);

lv_area_t txt_area;
txt_area.x1 = 0;
txt_area.x2 = txt_size.x - 1;
txt_area.y1 = 0;
txt_area.y2 = txt_size.y - 1;

lv_area_t indic_area;
lv_obj_get_coords(obj, &indic_area);
lv_area_set_width(&indic_area, lv_area_get_width(&indic_area) * lv_bar_get_value(obj) / MAX_VALUE);

/*If the indicator is long enough put the text inside on the right*/
if(lv_area_get_width(&indic_area) > txt_size.x + 20) {
    lv_area_align(&indic_area, &txt_area, LV_ALIGN_RIGHT_MID, -10, 0);
    label_dsc.color = lv_color_white();
}
/*If the indicator is still short put the text out of it on the right*/
else {
    lv_area_align(&indic_area, &txt_area, LV_ALIGN_OUT_RIGHT_MID, 10, 0);
    label_dsc.color = lv_color_black();
}

label_dsc.text = buf;
label_dsc.text_local = true;
lv_layer_t * layer = lv_event_get_layer(e);
lv_draw_label(layer, &label_dsc, &txt_area);
}

/***
 * Custom drawer on the bar to display the current value
 */
void lv_example_bar_6(void)
{
    lv_obj_t * bar = lv_bar_create(lv_screen_active());
    lv_bar_set_range(bar, MIN_VALUE, MAX_VALUE);
    lv_obj_set_size(bar, 200, 20);
    lv_obj_center(bar);
    lv_obj_add_event_cb(bar, event_cb, LV_EVENT_DRAW_MAIN_END, NULL);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, bar);
    lv_anim_set_values(&a, 0, 100);
    lv_anim_set_exec_cb(&a, set_value);
    lv_anim_set_duration(&a, 4000);
    lv_anim_set_reverse_duration(&a, 4000);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);
}

#endif

```

Barra con direzione opposta

```

#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/***
 * Bar with opposite direction
 */
void lv_example_bar_7(void)
{
    lv_obj_t * bar_top = lv_bar_create(lv_screen_active());
    lv_obj_set_size(bar_top, 20, 180);
    lv_bar_set_range(bar_top, 100, 0);
    lv_bar_set_value(bar_top, 70, LV_ANIM_OFF);
    lv_obj_align(bar_top, LV_ALIGN_CENTER, 0, 10);

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "From top to bottom");
    lv_obj_align_to(label, bar_top, LV_ALIGN_OUT_TOP_MID, 0, -5);
}

#endif

```

3.7.5 Pulsante

Pulsanti semplici

```

#include "../../lv_examples.h"
#if LV_USE_BUTTON && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);

    if(code == LV_EVENT_CLICKED) {
        LV_LOG_USER("Clicked");
    }
    else if(code == LV_EVENT_VALUE_CHANGED) {
        LV_LOG_USER("Toggled");
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

void lv_example_button_1(void)
{
    lv_obj_t * label;

    lv_obj_t * btn1 = lv_button_create(lv_screen_active());
    lv_obj_add_event_cb(btn1, event_handler, LV_EVENT_ALL, NULL);
    lv_obj_align(btn1, LV_ALIGN_CENTER, 0, -40);
    lv_obj_remove_flag(btn1, LV_OBJ_FLAG_PRESS_LOCK);

    label = lv_label_create(btn1);
    lv_label_set_text(label, "Button");
    lv_obj_center(label);

    lv_obj_t * btn2 = lv_button_create(lv_screen_active());
    lv_obj_add_event_cb(btn2, event_handler, LV_EVENT_ALL, NULL);
    lv_obj_align(btn2, LV_ALIGN_CENTER, 0, 40);
    lv_obj_add_flag(btn2, LV_OBJ_FLAG_CHECKABLE);
    lv_obj_set_height(btn2, LV_SIZE_CONTENT);

    label = lv_label_create(btn2);
    lv_label_set_text(label, "Toggle");
    lv_obj_center(label);

}

#endif

```

Pulsanti con stili

```

#include "../../lv_examples.h"
#if LV_USE_BUTTON && LV_BUILD_EXAMPLES

/***
 * Style a button from scratch
 */
void lv_example_button_2(void)
{
    /*Init the style for the default state*/
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_radius(&style, 3);

    lv_style_set_bg_opa(&style, LV_OPA_100);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_bg_grad_color(&style, lv_palette_darken(LV_PALETTE_BLUE, 2));
    lv_style_set_bg_grad_dir(&style, LV_GRAD_DIR_VER);

    lv_style_set_border_opa(&style, LV_OPA_40);
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_GREY));

    lv_style_set_shadow_width(&style, 8);
    lv_style_set_shadow_color(&style, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_shadow_offset_y(&style, 8);

    lv_style_set_outline_opa(&style, LV_OPA_COVER);
    lv_style_set_outline_color(&style, lv_palette_main(LV_PALETTE_BLUE));

    lv_style_set_text_color(&style, lv_color_white());
    lv_style_set_pad_all(&style, 10);

    /*Init the pressed style*/
    static lv_style_t style_pr;
    lv_style_init(&style_pr);

    /*Add a large outline when pressed*/
    lv_style_set_outline_width(&style_pr, 30);
    lv_style_set_outline_opa(&style_pr, LV_OPA_TRANSP);

    lv_style_set_translate_y(&style_pr, 5);
    lv_style_set_shadow_offset_y(&style_pr, 3);
    lv_style_set_bg_color(&style_pr, lv_palette_darken(LV_PALETTE_BLUE, 2));
    lv_style_set_bg_grad_color(&style_pr, lv_palette_darken(LV_PALETTE_BLUE, 4));

    /*Add a transition to the outline*/
    static lv_style_transition_dsc_t trans;
    static lv_style_prop_t props[] = {LV_STYLE_OUTLINE_WIDTH, LV_STYLE_OUTLINE_OPA, 0};
    lv_style_set_transition_dsc_init(&trans, props, lv_anim_path_linear, 300, 0, NULL);

    lv_style_set_transition(&style_pr, &trans);

    lv_obj_t * btn1 = lv_button_create(lv_screen_active());
    lv_obj_remove_style_all(btn1); /*Remove the style coming from the theme*/
    lv_obj_add_style(btn1, &style, 0);
    lv_obj_set_size(btn1, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_center(btn1);

    lv_obj_t * label = lv_label_create(btn1);
    lv_label_set_text(label, "Button");
    lv_obj_center(label);
}
#endif

```

Pulsante gommoso

```

#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_BUTTON

/***
 * Create a style transition on a button to act like a gum when clicked
 */
void lv_example_button_3(void)
{

```

(continues on next page)

(continua dalla pagina precedente)

```

/*Properties to transition*/
static lv_style_prop_t props[] = {
    LV_STYLE_TRANSFORM_WIDTH, LV_STYLE_TRANSFORM_HEIGHT, LV_STYLE_TEXT_LETTER_SPACE, 0
};

/*Transition descriptor when going back to the default state.
 *Add some delay to be sure the press transition is visible even if the press was very short*/
static lv_style_transition_dsc_t transition_dsc_def;
lv_style_transition_dsc_init(&transition_dsc_def, props, lv_anim_path_overshoot, 250, 100, NULL);

/*Transition descriptor when going to pressed state.
 *No delay, go to presses state immediately*/
static lv_style_transition_dsc_t transition_dsc_pr;
lv_style_transition_dsc_init(&transition_dsc_pr, props, lv_anim_path_ease_in_out, 250, 0, NULL);

/*Add only the new transition to be default state*/
static lv_style_t style_def;
lv_style_init(&style_def);
lv_style_set_transition(&style_def, &transition_dsc_def);

/*Add the transition and some transformation to the presses state.*/
static lv_style_t style_pr;
lv_style_init(&style_pr);
lv_style_set_transform_width(&style_pr, 10);
lv_style_set_transform_height(&style_pr, -10);
lv_style_set_text_letter_space(&style_pr, 10);
lv_style_set_transition(&style_pr, &transition_dsc_pr);

lv_obj_t * btn1 = lv_button_create(lv_screen_active());
lv_obj_align(btn1, LV_ALIGN_CENTER, 0, -80);
lv_obj_add_style(btn1, &style_pr, LV_STATE_PRESSED);
lv_obj_add_style(btn1, &style_def, 0);

lv_obj_t * label = lv_label_create(btn1);
lv_label_set_text(label, "Gum");
}
#endif

```

3.7.6 Matrice di Pulsanti

Matrice di pulsanti semplice

```

#include "../../lv_examples.h"
#if LV_USE_BUTTONMATRIX && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        uint32_t id = lv_buttonmatrix_get_selected_button(obj);
        const char * txt = lv_buttonmatrix_get_button_text(obj, id);
        LV_UNUSED(txt);
        LV_LOG_USER("%s was pressed\n", txt);
    }
}

static const char * bmm_map[] = {"1", "2", "3", "4", "5",
                                "6", "7", "8", "9", "0", "\n",
                                "Action1", "Action2", ""};

void lv_example_buttonmatrix_1(void)
{
    lv_obj_t * bmm1 = lv_buttonmatrix_create(lv_screen_active());
    lv_buttonmatrix_set_map(bmm1, bmm_map);
    lv_buttonmatrix_set_button_width(bmm1, 10, 2);           /*Make "Action1" twice as wide as "Action2"*/
    lv_buttonmatrix_set_button_ctrl(bmm1, 10, LV_BUTTONMATRIX_CTRL_CHECKABLE);
    lv_buttonmatrix_set_button_ctrl(bmm1, 11, LV_BUTTONMATRIX_CTRL_CHECKED);
    lv_obj_align(bmm1, LV_ALIGN_CENTER, 0, 0);
    lv_obj_add_event_cb(bmm1, event_handler, LV_EVENT_ALL, NULL);
}
#endif

```

Pulsanti personalizzati

```

#include "../../lv_examples.h"
#if LV_USE_BUTTONMATRIX && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_event_get_draw_dsc(draw_task);
    /*When the button matrix draws the buttons...*/
    if(base_dsc->part == LV_PART_ITEMS) {
        bool pressed = false;
        if(lv_buttonmatrix_get_selected_button(obj) == base_dsc->id1 && lv_obj_has_state(obj, LV_STATE_PRESSED)) {
            pressed = true;
        }

        /*Change the draw descriptor of the 2nd button*/
        if(base_dsc->id1 == 1) {
            lv_draw_fill_dsc_t * fill_draw_dsc = lv_draw_task_get_fill_dsc(draw_task);
            if(fill_draw_dsc) {
                fill_draw_dsc->radius = 0;
                if(pressed) fill_draw_dsc->color = lv_palette_darken(LV_PALETTE_BLUE, 3);
                else fill_draw_dsc->color = lv_palette_main(LV_PALETTE_BLUE);
            }
            lv_draw_box_shadow_dsc_t * box_shadow_draw_dsc = lv_draw_task_get_box_shadow_dsc(draw_task);
            if(box_shadow_draw_dsc) {

```

(continues on next page)

(continua dalla pagina precedente)

```

        box_shadow_draw_dsc->width = 6;
        box_shadow_draw_dsc->ofs_x = 3;
        box_shadow_draw_dsc->ofs_y = 3;
    }
    lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
    if(label_draw_dsc) {
        label_draw_dsc->color = lv_color_white();
    }

}

/*Change the draw descriptor of the 3rd button*/
else if(base_dsc->id1 == 2) {
    lv_draw_fill_dsc_t * fill_draw_dsc = lv_draw_task_get_fill_dsc(draw_task);
    if(fill_draw_dsc) {
        fill_draw_dsc->radius = LV_RADIUS_CIRCLE;
        if(pressed) fill_draw_dsc->color = lv_palette_darken(LV_PALETTE_RED, 3);
        else fill_draw_dsc->color = lv_palette_main(LV_PALETTE_RED);
    }

    lv_draw_box_shadow_dsc_t * box_shadow_draw_dsc = lv_draw_task_get_box_shadow_dsc(draw_task);
    if(box_shadow_draw_dsc) {
        box_shadow_draw_dsc->radius = LV_RADIUS_CIRCLE;
    }
}
else if(base_dsc->id1 == 3) {
    lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
    if(label_draw_dsc) {
        label_draw_dsc->opa = 0;
    }
    if(lv_draw_task_get_type(draw_task) == LV_DRAW_TASK_TYPE_FILL) {
        LV_IMAGE_DECLARE(img_star);
        lv_image_header_t header;
        lv_result_t res = lv_image_decoder_get_info(&img_star, &header);
        if(res != LV_RESULT_OK) return;

        lv_area_t a;
        a.x1 = 0;
        a.x2 = header.w - 1;
        a.y1 = 0;
        a.y2 = header.h - 1;
        lv_area_t draw_task_area;
        lv_draw_task_get_area(draw_task, &draw_task_area);
        lv_area_align(&draw_task_area, &a, LV_ALIGN_CENTER, 0, 0);

        lv_draw_image_dsc_t img_draw_dsc;
        lv_draw_image_dsc_init(&img_draw_dsc);
        img_draw_dsc.sr = &img_star;
        img_draw_dsc.recolor = lv_color_black();
        if(pressed) img_draw_dsc.recolor_opa = LV_OPA_30;

        lv_draw_image(base_dsc->layer, &img_draw_dsc, &a);
    }
}
}

/***
 * Add custom drawer to the button matrix to customize buttons one by one
 */
void lv_example_buttonmatrix_2(void)
{
    lv_obj_t * btnm = lv_buttonmatrix_create(lv_screen_active());
    lv_obj_add_event_cb(btnm, event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(btnm, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
    lv_obj_center(btnm);
}

#endif

```

Paginazione

```

#include "../../lv_examples.h"
#if LV_USE_BUTTONMATRIX && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    uint32_t id = lv_buttonmatrix_get_selected_button(obj);
    bool prev = id == 0;
    bool next = id == 6;
    if(prev || next) {
        /*Find the checked button*/
        uint32_t i;
        for(i = 1; i < 7; i++) {
            if(lv_buttonmatrix_has_button_ctrl(obj, i, LV_BUTTONMATRIX_CTRL_CHECKED)) break;
        }

        if(prev && i > 1) i--;
        else if(next && i < 5) i++;

        lv_buttonmatrix_set_button_ctrl(obj, i, LV_BUTTONMATRIX_CTRL_CHECKED);
    }
}

/***
 * Make a button group (pagination)
 */
void lv_example_buttonmatrix_3(void)
{
    static lv_style_t style_bg;
    lv_style_init(&style_bg);
    lv_style_set_pad_all(&style_bg, 0);
    lv_style_set_pad_gap(&style_bg, 0);
    lv_style_set_clip_corner(&style_bg, true);
    lv_style_set_radius(&style_bg, LV_RADIUS_CIRCLE);
    lv_style_set_border_width(&style_bg, 0);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

static lv_style_t style_btn;
lv_style_init(&style_btn);
lv_style_set_radius(&style_btn, 0);
lv_style_set_border_width(&style_btn, 1);
lv_style_set_border_opa(&style_btn, LV_OPA_50);
lv_style_set_border_color(&style_btn, lv_palette_main(LV_PALETTE_GREY));
lv_style_set_border_side(&style_btn, LV_BORDER_SIDE_INTERNAL);
lv_style_set_radius(&style_btn, 0);

static const char * map[] = {LV_SYMBOL_LEFT, "1", "2", "3", "4", "5", LV_SYMBOL_RIGHT, ""};

lv_obj_t * btm = lv_buttonmatrix_create(lv_screen_active());
lv_buttonmatrix_set_map(btm, map);
lv_obj_add_style(btm, &style_bg, 0);
lv_obj_add_style(btm, &style_btn, LV_PART_ITEMS);
lv_obj_add_event_cb(btm, event_cb, LV_EVENT_VALUE_CHANGED, NULL);
lv_obj_set_size(btm, 225, 35);

/*Allow selecting on one number at time*/
lv_buttonmatrix_set_button_ctrl_all(btm, LV_BUTTONMATRIX_CTRL_CHECKABLE);
lv_buttonmatrix_clear_button_ctrl(btm, 0, LV_BUTTONMATRIX_CTRL_CHECKABLE);
lv_buttonmatrix_clear_button_ctrl(btm, 6, LV_BUTTONMATRIX_CTRL_CHECKABLE);

lv_buttonmatrix_set_one_checked(btm, true);
lv_buttonmatrix_set_button_ctrl(btm, 1, LV_BUTTONMATRIX_CTRL_CHECKED);

lv_obj_center(btm);

}

#endif

```

3.7.7 Calendario

Calendario con intestazione

```

#include "../../lv_examples.h"
#if LV_USE_CALENDAR && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = (lv_obj_t *)lv_event_get_current_target(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        lv_calendar_date_t date;
        if(lv_calendar_get_pressed_date(obj, &date)) {
            LV_LOG_USER("Clicked date: %02d.%02d.%d", date.day, date.month, date.year);
        }
    }
}

void lv_example_calendar_1(void)
{
    lv_obj_t * calendar = lv_calendar_create(lv_screen_active());
    lv_obj_set_size(calendar, 185, 230);
    lv_obj_align(calendar, LV_ALIGN_CENTER, 0, 27);
    lv_obj_add_event_cb(calendar, event_handler, LV_EVENT_ALL, NULL);

    lv_calendar_set_today_date(calendar, 2021, 02, 23);
    lv_calendar_set_month_shown(calendar, 2021, 02);

    /*Highlight a few days*/
    static lv_calendar_date_t highlighted_days[3];      /*Only its pointer will be saved so should be static*/
    highlighted_days[0].year = 2021;
    highlighted_days[0].month = 02;
    highlighted_days[0].day = 6;

    highlighted_days[1].year = 2021;
    highlighted_days[1].month = 02;
    highlighted_days[1].day = 11;

    highlighted_days[2].year = 2022;
    highlighted_days[2].month = 02;
    highlighted_days[2].day = 22;

    lv_calendar_set_highlighted_dates(calendar, highlighted_days, 3);

#if LV_USE_CALENDAR_HEADER_DROPDOWN
    lv_calendar_add_header_dropdown(calendar);
#elif LV_USE_CALENDAR_HEADER_ARROW
    lv_calendar_add_header_arrow(calendar);
#endif
}
#endif

```

Calendario cinese

```

#include "../../lv_examples.h"
#if LV_USE_CALENDAR && LV_USE_CALENDAR_CHINESE && LV_BUILD_EXAMPLES

void lv_example_calendar_2(void)
{
    lv_obj_t * calendar = lv_calendar_create(lv_screen_active());
    lv_obj_set_size(calendar, 300, 300);
    lv_obj_align(calendar, LV_ALIGN_TOP_MID, 0, 0);

    lv_calendar_set_today_date(calendar, 2024, 03, 22);
    lv_calendar_set_month_shown(calendar, 2024, 03);

#if LV_USE_CALENDAR_HEADER_DROPDOWN

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_calendar_add_header_dropdown(calendar);
#endif
#if LV_USE_CALENDAR_HEADER_ARROW
lv_calendar_add_header_arrow(calendar);
#endif

lv_calendar_set_chinese_mode(calendar, true);
lv_obj_set_style_text_font(calendar, &lv_font_source_han_sans_sc_14_cjk, LV_PART_MAIN);
}

#else

void lv_example_calendar_2(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "chinese calendar is not enabled");
    lv_obj_center(label);
}
#endif

```

3.7.8 Canvas

Disegno sul Canvas e rotazione

```

#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 200
#define CANVAS_HEIGHT 150

void lv_example_canvas_1(void)
{
    lv_draw_rect_dsc_t rect_dsc;
    lv_draw_rect_dsc_init(&rect_dsc);
    rect_dsc.radius = 10;
    rect_dsc.bg_opa = LV_OPA_COVER;
    rect_dsc.bg_grad.dir = LV_GRAD_DIR_VER;
    rect_dsc.bg_grad.stops[0].color = lv_palette_main(LV_PALETTE_RED);
    rect_dsc.bg_grad.stops[0].opa = LV_OPA_100;
    rect_dsc.bg_grad.stops[1].color = lv_palette_main(LV_PALETTE_BLUE);
    rect_dsc.bg_grad.stops[1].opa = LV_OPA_50;
    rect_dsc.border_width = 2;
    rect_dsc.border_opa = LV_OPA_90;
    rect_dsc.border_color = lv_color_white();
    rect_dsc.shadow_width = 5;
    rect_dsc.shadow_offset_x = 5;
    rect_dsc.shadow_offset_y = 5;

    lv_draw_label_dsc_t label_dsc;
    lv_draw_label_dsc_init(&label_dsc);
    label_dsc.color = lv_palette_main(LV_PALETTE_ORANGE);
    label_dsc.text = "Some text on text canvas";
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf_16bpp, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_RGB565);
    LV_DRAW_BUF_INIT_STATIC(draw_buf_16bpp);

    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf_16bpp);
    lv_obj_center(canvas);
    lv_canvas_fill_bg(canvas, lv_palette_lighten(LV_PALETTE_GREY, 3), LV_OPA_COVER);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_area_t coords_rect = {30, 20, 100, 70};
    lv_draw_rect(&layer, &rect_dsc, &coords_rect);

    lv_area_t coords_text = {40, 80, 100, 120};
    lv_draw_label(&layer, &label_dsc, &coords_text);

    lv_canvas_finish_layer(canvas, &layer);

    /*Test the rotation. It requires another buffer where the original image is stored.
     *So use previous canvas as image and rotate it to the new canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf_32bpp, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf_32bpp);

    /*Create a canvas and initialize its palette*/
    canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf_32bpp);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_canvas_fill_bg(canvas, lv_palette_lighten(LV_PALETTE_GREY, 1), LV_OPA_COVER);

    lv_canvas_init_layer(canvas, &layer);
    lv_image_dsc_t img;
    lv_draw_buf_to_image(&draw_buf_16bpp, &img);
    lv_draw_image_dsc_t img_dsc;
    lv_draw_image_dsc_init(&img_dsc);
    img_dsc.rotation = 120;
    img_dsc.src = &img;
    img_dsc.pivot.x = CANVAS_WIDTH / 2;
    img_dsc.pivot.y = CANVAS_HEIGHT / 2;

    lv_area_t coords_img = {0, 0, CANVAS_WIDTH - 1, CANVAS_HEIGHT - 1};
    lv_draw_image(&layer, &img_dsc, &coords_img);

    lv_canvas_finish_layer(canvas, &layer);
}
#endif

```

Canvas trasparente con chroma key

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 80
#define CANVAS_HEIGHT 40

/***
 * Create a transparent canvas with transparency
 */
void lv_example_canvas_2(void)
{
    lv_obj_set_style_bg_color(lv_screen_active(), lv_palette_lighten(LV_PALETTE_RED, 5), 0);

    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_obj_center(canvas);

    /*Red background (There is no dedicated alpha channel in indexed images so LV_OPA_COVER is ignored)*/
    lv_canvas_fill_bg(canvas, lv_palette_main(LV_PALETTE_BLUE), LV_OPA_COVER);

    /*Create hole on the canvas*/
    int32_t x;
    int32_t y;
    for(y = 10; y < 20; y++) {
        for(x = 5; x < 75; x++) {
            lv_canvas_set_px(canvas, x, y, lv_palette_main(LV_PALETTE_BLUE), LV_OPA_50);
        }
    }

    for(y = 20; y < 30; y++) {
        for(x = 5; x < 75; x++) {
            lv_canvas_set_px(canvas, x, y, lv_palette_main(LV_PALETTE_BLUE), LV_OPA_20);
        }
    }

    for(y = 30; y < 40; y++) {
        for(x = 5; x < 75; x++) {
            lv_canvas_set_px(canvas, x, y, lv_palette_main(LV_PALETTE_BLUE), LV_OPA_0);
        }
    }
}
#endif
```

Disegnare un rettangolo sul canvas

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw a rectangle to the canvas
 */
void lv_example_canvas_3(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);

    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_rect_dsc_t dsc;
    lv_draw_rect_dsc_init(&dsc);
    dsc.bg_color = lv_palette_main(LV_PALETTE_RED);
    dsc.border_color = lv_palette_main(LV_PALETTE_BLUE);
    dsc.border_width = 3;
    dsc.outline_color = lv_palette_main(LV_PALETTE_GREEN);
    dsc.outline_width = 2;
    dsc.outline_pad = 2;
    dsc.outline_opa = LV_OPA_50;
    dsc.radius = 5;
    dsc.border_width = 3;

    lv_area_t coords = {10, 10, 40, 30};

    lv_draw_rect(&layer, &dsc, &coords);
    lv_canvas_finish_layer(canvas, &layer);
}
#endif
```

Disegnare un'etichetta sul canvas

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_FONT_MONTSERAT_18 && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50
```

(continues on next page)

(continua dalla pagina precedente)

```
/**
 * Draw a text to the canvas
 */
void lv_example_canvas_4(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_label_dsc_t dsc;
    lv_draw_label_dsc_init(&dsc);
    dsc.color = lv_palette_main(LV_PALETTE_RED);
    dsc.font = &lv_font_montserrat_18;
    dsc.decor = LV_TEXT_DECOR_UNDERLINE;
    dsc.text = "Hello";

    lv_area_t coords = {10, 10, 30, 60};

    lv_draw_label(&layer, &dsc, &coords);

    lv_canvas_finish_layer(canvas, &layer);
}

#endif
```

Disegnare un arco sul canvas

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/**
 * Draw an arc to the canvas
 */
void lv_example_canvas_5(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_arc_dsc_t dsc;
    lv_draw_arc_dsc_init(&dsc);
    dsc.color = lv_palette_main(LV_PALETTE_RED);
    dsc.center.x = 25;
    dsc.center.y = 25;
    dsc.width = 10;
    dsc.radius = 15;
    dsc.start_angle = 0;
    dsc.end_angle = 220;

    lv_draw_arc(&layer, &dsc);

    lv_canvas_finish_layer(canvas, &layer);
}

#endif
```

Disegnare un'immagine sul canvas

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/**
 * Draw an image to the canvas
 */
void lv_example_canvas_6(void)
{
    /*Create a buffer for the canvas*/
    static uint8_t cbuf[LV_CANVAS_BUF_SIZE(CANVAS_WIDTH, CANVAS_HEIGHT, 32, LV_DRAW_BUF_STRIDE_ALIGN)];

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_buffer(canvas, cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    LV_IMAGE_DECLARE(img_star);
    lv_draw_image_dsc_t dsc;
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_draw_image_dsc_init(&dsc);
dsc.src = &img_star;

lv_area_t coords = {10, 10, 10 + img_star.header.w - 1, 10 + img_star.header.h - 1};

lv_draw_image(&layer, &dsc, &coords);

lv_canvas_finish_layer(canvas, &layer);
}
#endif

```

Disegnare una linea sul canvas

```

#include "../../lv_examples.h"
#if LV_USE_CANVAS&& LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw a line to the canvas
 */
void lv_example_canvas_7(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_line_dsc_t dsc;
    lv_draw_line_dsc_init(&dsc);
    dsc.color = lv_palette_main(LV_PALETTE_RED);
    dsc.width = 4;
    dsc.round_end = 1;
    dsc.round_start = 1;
    dsc.p1.x = 15;
    dsc.p1.y = 15;
    dsc.p2.x = 35;
    dsc.p2.y = 10;
    lv_draw_line(&layer, &dsc);

    lv_canvas_finish_layer(canvas, &layer);
}
#endif

```

Disegnare un'immagine vettoriale sul canvas

```

#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#if LV_USE_VECTOR_GRAPHIC

#define CANVAS_WIDTH 150
#define CANVAS_HEIGHT 150

/***
 * Draw a path to the canvas
 */
void lv_example_canvas_8(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_vector_dsc_t * dsc = lv_draw_vector_dsc_create(&layer);
    lv_vector_path_t * path = lv_vector_path_create(LV_VECTOR_PATH_QUALITY_MEDIUM);

    lv_fpint_t pts[] = {{10, 10}, {130, 130}, {10, 130}};
    lv_vector_path_move_to(path, &pts[0]);
    lv_vector_path_line_to(path, &pts[1]);
    lv_vector_path_line_to(path, &pts[2]);
    lv_vector_path_close(path);

    lv_draw_vector_dsc_set_fill_color(dsc, lv_color_make(0x00, 0x80, 0xff));
    lv_draw_vector_dsc_add_path(dsc, path);

    lv_draw_vector(dsc);
    lv_vector_path_delete(path);
    lv_draw_vector_dsc_delete(dsc);

    lv_canvas_finish_layer(canvas, &layer);
}
#endif

void lv_example_canvas_8(void)
{

```

(continues on next page)

(continua dalla pagina precedente)

```

/*fallback for online examples*/
lv_obj_t * label = lv_label_create(lv_screen_active());
lv_label_set_text(label, "Vector graphics is not enabled");
lv_obj_center(label);
}

#endif /*LV_USE_VECTOR_GRAPHIC*/
#endif

```

Disegnare un triangolo sul canvas

```

#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 150
#define CANVAS_HEIGHT 150

/***
 * Draw a triangle to the canvas
 */
void lv_example_canvas_9(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_triangle_dsc_t tri_dsc;
    lv_draw_triangle_dsc_init(&tri_dsc);
    tri_dsc.p[0].x = 10;
    tri_dsc.p[0].y = 10;
    tri_dsc.p[1].x = 100;
    tri_dsc.p[1].y = 30;
    tri_dsc.p[2].x = 50;
    tri_dsc.p[2].y = 100;

    tri_dsc.grad.stops_count = 2;
    tri_dsc.grad.dir = LV_GRAD_DIR_VER;
    tri_dsc.grad.stops[0].color = lv_color_hex(0xffff00);
    tri_dsc.grad.stops[0].frac = 64; /*Start at 25%*/
    tri_dsc.grad.stops[0].opa = LV_OPA_COVER;
    tri_dsc.grad.stops[1].color = lv_color_hex(0x0000ff);
    tri_dsc.grad.stops[1].opa = LV_OPA_TRANSP;
    tri_dsc.grad.stops[1].frac = 3 * 64; /*End at 75%*/

    tri_dsc.opa = 128; /*Set the overall opacity to 50%*/

    lv_draw_triangle(&layer, &tri_dsc);

    lv_canvas_finish_layer(canvas, &layer);
}
#endif

```

Disegnare Effetti di Lettere Fantasiose

```

#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 300
#define CANVAS_HEIGHT 200

static void timer_cb(lv_timer_t * timer)
{
    static int16_t counter = 0;
    const char * string = "lol- I'm wawwwwving>>>";
    const int16_t string_len = lv_strlen(string);

    lv_obj_t * canvas = (lv_obj_t *)lv_timer_get_user_data(timer);
    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_canvas_fill_bg(canvas, lv_color_white(), LV_OPA_COVER);

    lv_draw_letter_dsc_t letter_dsc;
    lv_draw_letter_dsc_init(&letter_dsc);
    letter_dsc.color = lv_color_hex(0xff0000);
    letter_dsc.font = lv_font_get_default();

    {
#define CURVE2_X(t) (t * 2 + 10)
#define CURVE2_Y(t) (lv_trigo_sin((t) * 5) * 40 / 32767 + CANVAS_HEIGHT / 2)

        int32_t pre_x = CURVE2_X(-1);
        int32_t pre_y = CURVE2_Y(-1);
        for(int16_t i = 0; i < string_len; i++) {
            const int16_t angle = (int16_t)(i * 5);
            const int32_t x = CURVE2_X(angle);
            const int32_t y = CURVE2_Y(angle + counter / 2);
            const lv_point_t point = { .x = x, .y = y };

            letter_dsc_unicode = (uint32_t)string[i % string_len];
            letter_dsc.rotation = lv_atan2(y - pre_y, x - pre_x) * 10;
            letter_dsc.color = lv_color_hsv_to_rgb(i * 10, 100, 100);
        }
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

    lv_draw_letter(&layer, &letter_dsc, &point);

    pre_x = x;
    pre_y = y;
}
}

lv_canvas_finish_layer(canvas, &layer);

counter++;
}

void lv_example_canvas_10(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_obj_set_size(canvas, CANVAS_WIDTH, CANVAS_HEIGHT);

    lv_obj_center(canvas);

    lv_canvas_set_draw_buf(canvas, &draw_buf);

    lv_timer_create(timer_cb, 16, canvas);
}
#endif

```

Disegnare Effetti di Lettere Fantasiose 2

```

#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 300
#define CANVAS_HEIGHT 200

static void timer_cb(lv_timer_t * timer)
{
    static int32_t counter = 0;
    const char * string = "windstorrccccccccccc>>>";
    const int16_t string_len = lv_strlen(string);

    lv_obj_t * canvas = (lv_obj_t *) lv_timer_get_user_data(timer);
    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_canvas_fill_bg(canvas, lv_color_white(), LV_OPA_COVER);

    lv_draw_letter_dsc_t letter_dsc;
    lv_draw_letter_dsc_init(&letter_dsc);
    letter_dsc.color = lv_color_hex(0xffff00);
    letter_dsc.font = lv_font_get_default();

    {
#define CURVE2_X(t) ((t) * 2 + lv_trigo_cos((t) * 5) * 40 / 32767 - 10)
#define CURVE2_Y(t, T) ((t) * lv_trigo_sin(((t) + (T)) * 5) * 40 / 32767 / 80 + CANVAS_HEIGHT / 2)

        int32_t pre_x = CURVE2_X(-1);
        int32_t pre_y = CURVE2_Y(-1, 0);
        for(int16_t i = 0; i < string_len; i++) {
            const int32_t angle = i * 5;
            const int32_t x = CURVE2_X(angle);
            const int32_t y = CURVE2_Y(angle + 30, counter / 2);

            letter_dsc_unicode = (uint32_t)string[i % string_len];
            letter_dsc.rotation = lv_atan2(y - pre_y, x - pre_x) * 10;
            letter_dsc.color = lv_color_hsv_to_rgb(i * 10, 100, 100);
            lv_point_t p = (lv_point_t) {
                .x = x, .y = y
            };
            lv_draw_letter(&layer, &letter_dsc, &p);

            pre_x = x;
            pre_y = y;
        }
    }

    lv_canvas_finish_layer(canvas, &layer);

    counter++;
}

void lv_example_canvas_11(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_obj_set_size(canvas, CANVAS_WIDTH, CANVAS_HEIGHT);

    lv_obj_center(canvas);

    lv_canvas_set_draw_buf(canvas, &draw_buf);

    lv_timer_create(timer_cb, 16, canvas);
}
#endif

```

3.7.9 Grafico

Grafico a linee

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

/***
 * A very basic line chart
 */
void lv_example_chart_1(void)
{
    /*Create a chart*/
    lv_obj_t * chart;
    chart = lv_chart_create(lv_screen_active());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_center(chart);
    lv_chart_set_type(chart, LV_CHART_TYPE_LINE); /*Show lines and points too*/

    /*Add two data series*/
    lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_GREEN), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_SECONDARY_Y);
    int32_t * ser2_y_points = lv_chart_get_series_y_array(chart, ser2);

    uint32_t i;
    for(i = 0; i < 10; i++) {
        /*Set the next points on 'ser1'*/
        lv_chart_set_next_value(chart, ser1, (int32_t)lv_rand(10, 50));

        /*Directly set points on 'ser2'*/
        ser2_y_points[i] = (int32_t)lv_rand(50, 90);
    }

    lv_chart_refresh(chart); /*Required after direct set*/
}

#endif
```

Segni di spunta e etichette degli assi con scorrimento su un grafico

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

/***
 * Use lv_scale to add ticks to a scrollable chart
 */
void lv_example_chart_2(void)
{
    /*Create a container*/
    lv_obj_t * main_cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(main_cont, 200, 150);
    lv_obj_center(main_cont);

    /*Create a transparent wrapper for the chart and the scale.
     *Set a large width, to make it scrollable on the main container*/
    lv_obj_t * wrapper = lv_obj_create(main_cont);
    lv_obj_remove_style_all(wrapper);
    lv_obj_set_size(wrapper, lv_pct(300), lv_pct(100));
    lv_obj_set_flex_flow(wrapper, LV_FLEX_FLOW_COLUMN);

    /*Create a chart on the wrapper
     *Set its width to 100% to fill the large wrapper*/
    lv_obj_t * chart = lv_chart_create(wrapper);
    lv_obj_set_width(chart, lv_pct(100));
    lv_obj_set_flex_grow(chart, 1);
    lv_chart_set_type(chart, LV_CHART_TYPE_BAR);
    lv_chart_set_axis_range(chart, LV_CHART_AXIS_PRIMARY_Y, 0, 100);
    lv_chart_set_axis_range(chart, LV_CHART_AXIS_SECONDARY_Y, 0, 400);
    lv_chart_set_point_count(chart, 12);
    lv_obj_set_style_radius(chart, 0, 0);

    /*Create a scale also with 100% width*/
    lv_obj_t * scale_bottom = lv_scale_create(wrapper);
    lv_scale_set_mode(scale_bottom, LV_SCALE_MODE_HORIZONTAL_BOTTOM);
    lv_obj_set_size(scale_bottom, lv_pct(100), 25);
    lv_scale_set_total_tick_count(scale_bottom, 12);
    lv_scale_set_major_tick_every(scale_bottom, 1);
    lv_obj_set_style_pad_hor(scale_bottom, lv_chart_get_first_point_center_offset(chart), 0);

    static const char * month[] = {"Jan", "Febr", "March", "Apr", "May", "Jun", "July", "Aug", "Sept", "Oct", "Nov", "Dec", NULL};
    lv_scale_set_text_src(scale_bottom, month);

    /*Add two data series*/
    lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_lighten(LV_PALETTE_GREEN, 2), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_darken(LV_PALETTE_GREEN, 2), LV_CHART_AXIS_PRIMARY_Y);

    /*Set the next points on 'ser1'*/
    uint32_t i;
    for(i = 0; i < 12; i++) {
        lv_chart_set_next_value(chart, ser1, (int32_t)lv_rand(10, 60));
        lv_chart_set_next_value(chart, ser2, (int32_t)lv_rand(50, 90));
    }

    lv_chart_refresh(chart); /*Required after direct set*/
}

#endif
```

Mostrare il valore dei punti premuti

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * chart = lv_event_get_target_obj(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        lv_obj_invalidate(chart);
    }
    if(code == LV_EVENT_REFR_EXT_DRAW_SIZE) {
        int32_t * s = (int32_t *)lv_event_get_param(e);
        *s = LV_MAX(*s, 20);
    }
    else if(code == LV_EVENT_DRAW_POST_END) {
        uint32_t id = lv_chart_get_pressed_point(chart);
        if(id == LV_CHART_POINT_NONE) return;

        LV_LOG_USER("Selected point %d", (int)id);

        lv_chart_series_t * ser = lv_chart_get_series_next(chart, NULL);
        int32_t value = 0;
        while(ser) {
            lv_point_t p;
            lv_chart_get_point_pos_by_id(chart, ser, id, &p);

            int32_t * y_array = lv_chart_get_series_y_array(chart, ser);
            if(y_array[id] != LV_CHART_POINT_NONE && y_array[id] >= 0) {
                /*Accumulate the values to show the rectangles at the top of each segment*/
                value += y_array[id];
            }

            /*Draw a rectangle above the clicked point*/
            lv_layer_t * layer = lv_event_get_layer(e);
            lv_draw_rect_dsc_t draw_rect_dsc;
            lv_draw_rect_dsc_init(&draw_rect_dsc);
            draw_rect_dsc.bg_color = lv_color_black();
            draw_rect_dsc.bg_opa = LV_OPA_50;
            draw_rect_dsc.radius = 3;

            lv_area_t chart_obj_coords;
            lv_obj_get_coords(chart, &chart_obj_coords);
            lv_area_t rect_area;
            rect_area.x1 = chart_obj_coords.x1 + p.x - 20;
            rect_area.x2 = chart_obj_coords.x1 + p.x + 20;
            rect_area.y1 = chart_obj_coords.y1 + p.y - 10;
            rect_area.y2 = chart_obj_coords.y1 + p.y + 10;
            lv_draw_rect(layer, &draw_rect_dsc, &rect_area);

            /*Draw the value as label to the center of the rectangle*/
            char buf[16];
            lv_snprintf(buf, sizeof(buf), LV_SYMBOL_DUMMY"%d", value);

            lv_draw_label_dsc_t draw_label_dsc;
            lv_draw_label_dsc_init(&draw_label_dsc);
            draw_label_dsc.color = lv_color_white();
            draw_label_dsc.text = buf;
            draw_label_dsc.text_local = 1;
            draw_label_dsc.align = LV_TEXT_ALIGN_CENTER;
            lv_area_t label_area = rect_area;
            lv_area_set_height(&label_area, lv_font_get_line_height(draw_label_dsc.font));
            lv_area_align(&rect_area, &label_area, LV_ALIGN_CENTER, 0, 0);
            lv_draw_label(layer, &draw_label_dsc, &label_area);
        }

        ser = lv_chart_get_series_next(chart, ser);
    }
    else if(code == LV_EVENT_RELEASED) {
        lv_obj_invalidate(chart);
    }
}

/**
 * Show the value of the pressed points
 */
void lv_example_chart_3(void)
{
    /*Create a chart*/
    lv_obj_t * chart;
    chart = lv_chart_create(lv_screen_active());
    lv_obj_set_size(chart, 280, 180);
    lv_obj_set_style(chart, 4, 0);
    lv_obj_center(chart);
    lv_chart_set_type(chart, LV_CHART_TYPE_STACKED);

    lv_obj_add_event_cb(chart, event_cb, LV_EVENT_ALL, NULL);
    lv_obj_refresh_ext_draw_size(chart);

    /*Add two data series*/
    lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_GREEN), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser3 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_BLUE), LV_CHART_AXIS_PRIMARY_Y);
    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_chart_set_next_value(chart, ser1, lv_rand(15, 30));
        lv_chart_set_next_value(chart, ser2, lv_rand(15, 30));
        lv_chart_set_next_value(chart, ser3, lv_rand(15, 30));
    }
}

#endif
```

Ricolorare le barre in base al loro valore

```
#include "../../lv_examples.h"

#if LV_USE_CHART && LV_DRAW_SW_COMPLEX && LV_BUILD_EXAMPLES

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);

    if(base_dsc->part != LV_PART_ITEMS) {
        return;
    }

    lv_draw_fill_dsc_t * fill_dsc = lv_draw_task_get_fill_dsc(draw_task);
    if(fill_dsc) {
        lv_obj_t * chart = lv_event_get_target_obj(e);
        int32_t * y_array = lv_chart_get_series_y_array(chart, lv_chart_get_series_next(chart, NULL));
        int32_t v = y_array[base_dsc->i02];

        uint8_t ratio = (uint8_t)(v * 255 / 100);
        fill_dsc->color = lv_color_mix(lv_palette_main(LV_PALETTE_GREEN), lv_palette_main(LV_PALETTE_RED), ratio);
    }
}

/**
 * Recolor the bars of a chart based on their value
 */
void lv_example_chart_4(void)
{
    /*Create a chart*/
    lv_obj_t * chart = lv_chart_create(lv_screen_active());
    lv_chart_set_type(chart, LV_CHART_TYPE_BAR);
    lv_chart_set_point_count(chart, 24);
    lv_obj_set_style_pad_column(chart, 2, 0);
    lv_obj_set_size(chart, 260, 160);
    lv_obj_center(chart);

    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_color_hex(0xffff00), LV_CHART_AXIS_PRIMARY_Y);
    lv_obj_add_event_cb(chart, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(chart, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);

    uint32_t i;
    for(i = 0; i < 24; i++) {
        lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 90));
    }
}
#endif
```

Grafico a linee sfumate con linee di divisione personalizzate

```
#include "../../lv_examples.h"

#if LV_USE_CHART && LV_DRAW_SW_COMPLEX && LV_BUILD_EXAMPLES
#include "../../../../lvgl_private.h"

static void hook_division_lines(lv_event_t * e);
static void add_faded_area(lv_event_t * e);
static void draw_event_cb(lv_event_t * e);

/**
 * Add a faded area effect to the line chart and make some division lines ticker
 */
void lv_example_chart_5(void)
{
    /*Create a chart*/
    lv_obj_t * chart = lv_chart_create(lv_screen_active());
    lv_chart_set_type(chart, LV_CHART_TYPE_LINE); /*Show lines and points too*/
    lv_obj_set_size(chart, 200, 150);
    lv_obj_set_style_pad_all(chart, 0, 0);
    lv_obj_set_style_radius(chart, 0, 0);
    lv_obj_center(chart);

    lv_chart_set_div_line_count(chart, 5, 7);

    lv_obj_add_event_cb(chart, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(chart, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);

    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);

    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 80));
    }
}

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);

    if(base_dsc->part == LV_PART_ITEMS && lv_draw_task_get_type(draw_task) == LV_DRAW_TASK_TYPE_LINE) {
        add_faded_area(e);
    }

    /*Hook the division lines too*/
    if(base_dsc->part == LV_PART_MAIN && lv_draw_task_get_type(draw_task) == LV_DRAW_TASK_TYPE_LINE) {
        hook_division_lines(e);
    }
}

static void add_faded_area(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_area_t coords;
    lv_obj_get_coords(obj, &coords);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);

const lv_chart_series_t * ser = lv_chart_get_series_next(obj, NULL);
lv_color_t ser_color = lv_chart_get_series_color(obj, ser);

/*Draw a triangle below the line with some opacity gradient*/
lv_draw_line_dsc_t * draw_line_dsc = (lv_draw_line_dsc_t *)lv_draw_task_get_draw_dsc(draw_task);
lv_draw_triangle_dsc_t tri_dsc;

lv_draw_triangle_dsc_init(&tri_dsc);
tri_dsc.p[0].x = draw_line_dsc->p1.x;
tri_dsc.p[0].y = draw_line_dsc->p1.y;
tri_dsc.p[1].x = draw_line_dsc->p2.x;
tri_dsc.p[1].y = draw_line_dsc->p2.y;
tri_dsc.p[2].x = draw_line_dsc->p1.y < draw_line_dsc->p2.y ? draw_line_dsc->p1.x : draw_line_dsc->p2.x;
tri_dsc.p[2].y = LV_MAX(draw_line_dsc->p1.y, draw_line_dsc->p2.y);
tri_dsc.grad.dir = LV_GRAD_DIR_VER;

int32_t full_h = lv_obj_get_height(obj);
int32_t fract_uppter = (int32_t)(LV_MIN(draw_line_dsc->p1.y, draw_line_dsc->p2.y) - coords.y1) * 255 / full_h;
int32_t fract_lower = (int32_t)(LV_MAX(draw_line_dsc->p1.y, draw_line_dsc->p2.y) - coords.y1) * 255 / full_h;
tri_dsc.grad.stops[0].color = ser_color;
tri_dsc.grad.stops[0].opa = (lv_opa_t)(255 - fract_uppter);
tri_dsc.grad.stops[0].frac = 0;
tri_dsc.grad.stops[1].color = ser_color;
tri_dsc.grad.stops[1].opa = (lv_opa_t)(255 - fract_lower);
tri_dsc.grad.stops[1].frac = 255;

lv_draw_triangle(base_dsc->layer, &tri_dsc);

/*Draw rectangle below the triangle*/
lv_draw_rect_dsc_t rect_dsc;
lv_draw_rect_dsc_init(&rect_dsc);
rect_dsc.bg.grad.dir = LV_GRAD_DIR_VER;
rect_dsc.bg.grad.stops[0].color = ser_color;
rect_dsc.bg.grad.stops[0].frac = 0;
rect_dsc.bg.grad.stops[0].opa = (lv_opa_t)(255 - fract_lower);
rect_dsc.bg.grad.stops[1].color = ser_color;
rect_dsc.bg.grad.stops[1].frac = 255;
rect_dsc.bg.grad.stops[1].opa = 0;

lv_area_t rect_area;
rect_area.x1 = (int32_t)draw_line_dsc->p1.x;
rect_area.x2 = (int32_t)draw_line_dsc->p2.x - 1;
rect_area.y1 = (int32_t)LV_MAX(draw_line_dsc->p1.y, draw_line_dsc->p2.y);
rect_area.y2 = (int32_t)coords.y2;
lv_draw_rect(base_dsc->layer, &rect_dsc, &rect_area);
}

static void hook_division_lines(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    lv_draw_line_dsc_t * line_dsc = (lv_draw_line_dsc_t *)lv_draw_task_get_draw_dsc(draw_task);

    /*Vertical line*/
    if(line_dsc->p1.x == line_dsc->p2.x) {
        line_dsc->color = lv_palette_lighten(LV_PALETTE_GREY, 1);
        if(base_dsc->id1 == 3) {
            line_dsc->width = 2;
            line_dsc->dash_gap = 0;
            line_dsc->dash_width = 0;
        }
        else {
            line_dsc->width = 1;
            line_dsc->dash_gap = 6;
            line_dsc->dash_width = 6;
        }
    }
    /*Horizontal line*/
    else {
        if(base_dsc->id1 == 2) {
            line_dsc->width = 2;
            line_dsc->dash_gap = 0;
            line_dsc->dash_width = 0;
        }
        else {
            line_dsc->width = 2;
            line_dsc->dash_gap = 6;
            line_dsc->dash_width = 6;
        }

        if(base_dsc->id1 == 1 || base_dsc->id1 == 3) {
            line_dsc->color = lv_palette_main(LV_PALETTE_GREEN);
        }
        else {
            line_dsc->color = lv_palette_lighten(LV_PALETTE_GREY, 1);
        }
    }
}
#endif

```

Mostrare il cursore sul punto cliccato

```

#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

static lv_obj_t * chart;
static lv_chart_series_t * ser;
static lv_chart_cursor_t * cursor;

static void value_changed_event_cb(lv_event_t * e)
{
    uint32_t last_id;
    lv_obj_t * obj = lv_event_get_target_obj(e);

```

(continues on next page)

(continua dalla pagina precedente)

```

last_id = lv_chart_get_pressed_point(obj);
if(last_id != LV_CHART_POINT_NONE) {
    lv_chart_set_cursor_point(obj, cursor, NULL, last_id);
}
}

/***
 * Show cursor on the clicked point
 */
void lv_example_chart_6(void)
{
    chart = lv_chart_create(lv_screen_active());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_align(chart, LV_ALIGN_CENTER, 0, -10);

    // lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_Y, 10, 5, 6, 5, true, 40);
    // lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_X, 10, 5, 10, 1, true, 30);

    lv_obj_add_event_cb(chart, value_changed_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_refresh_ext_draw_size(chart);

    cursor = lv_chart_add_cursor(chart, lv_palette_main(LV_PALETTE_BLUE), (lv_dir_t)(LV_DIR_LEFT | LV_DIR_BOTTOM));

    ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 90));
    }

    // lv_chart_set_scale_x(chart, 500);

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Click on a point");
    lv_obj_align_to(label, chart, LV_ALIGN_OUT_TOP_MID, 0, -5);
}

#endif

```

Grafico a dispersione

```

#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    if(base_dsc->part == LV_PART_INDICATOR) {
        lv_obj_t * obj = lv_event_get_target_obj(e);
        lv_chart_series_t * ser = lv_chart_get_series_next(obj, NULL);
        lv_draw_rect_dsc_t * rect_dsc = (lv_draw_rect_dsc_t *)lv_draw_task_get_draw_dsc(draw_task);
        uint32_t cnt = lv_chart_get_point_count(obj);

        /*Make older value more transparent*/
        rect_dsc->bg_opa = (lv_opa_t)((LV_OPA_COVER * base_dsc->id2) / (cnt - 1));

        /*Make smaller values blue, higher values red*/
        int32_t * x_array = lv_chart_get_series_x_array(obj, ser);
        int32_t * y_array = lv_chart_get_series_y_array(obj, ser);
        /*dsc->id is the tells drawing order, but we need the ID of the point being drawn.*/
        uint32_t start_point = lv_chart_get_x_start_point(obj, ser);
        uint32_t p_act = (start_point + base_dsc->id2) % cnt; /*Consider start point to get the index of the array*/
        lv_opa_t x_opa = ((x_array[p_act] * LV_OPA_50) / 200);
        lv_opa_t y_opa = ((y_array[p_act] * LV_OPA_50) / 1000);

        rect_dsc->bg_color = lv_color_mix(lv_palette_main(LV_PALETTE_RED),
                                           lv_palette_main(LV_PALETTE_BLUE),
                                           x_opa + y_opa);
    }
}

static void add_data(lv_timer_t * timer)
{
    lv_obj_t * chart = (lv_obj_t *)lv_timer_get_user_data(timer);
    lv_chart_set_next_value2(chart, lv_chart_get_series_next(chart, NULL), (int32_t)lv_rand(0, 200),
                           (int32_t)lv_rand(0, 1000));
}

/***
 * A scatter chart
 */
void lv_example_chart_7(void)
{
    lv_obj_t * chart = lv_chart_create(lv_screen_active());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_align(chart, LV_ALIGN_CENTER, 0, 0);
    lv_obj_add_event_cb(chart, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(chart, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
    lv_obj_set_style_line_width(chart, 0, LV_PART_ITEMS); /*Remove the lines*/

    lv_chart_set_type(chart, LV_CHART_TYPE_SCATTER);

    lv_chart_set_axis_range(chart, LV_CHART_AXIS_PRIMARY_X, 0, 200);
    lv_chart_set_axis_range(chart, LV_CHART_AXIS_PRIMARY_Y, 0, 1000);

    lv_chart_set_point_count(chart, 50);

    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    uint32_t i;
    for(i = 0; i < 50; i++) {
        lv_chart_set_next_value2(chart, ser, (int32_t)lv_rand(0, 200), (int32_t)lv_rand(0, 1000));
    }

    lv_timer_create(add_data, 100, chart);
}

#endif

```

Grafico a linee circolari con spazio

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_DRAW_SW_COMPLEX && LV_BUILD_EXAMPLES

static void add_data(lv_timer_t * t)
{
    lv_obj_t * chart = (lv_obj_t *)lv_timer_get_user_data(t);
    lv_chart_series_t * ser = lv_chart_get_series_next(chart, NULL);

    lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 90));

    uint32_t p = lv_chart_get_point_count(chart);
    uint32_t s = lv_chart_get_x_start_point(chart, ser);
    int32_t * a = lv_chart_get_series_y_array(chart, ser);

    a[(s + 1) % p] = LV_CHART_POINT_NONE;
    a[(s + 2) % p] = LV_CHART_POINT_NONE;
    a[(s + 2) % p] = LV_CHART_POINT_NONE;

    lv_chart_refresh(chart);
}

/**
 * Circular line chart with gap
 */
void lv_example_chart_8(void)
{
    /*Create a stacked_area_chart.obj*/
    lv_obj_t * chart = lv_chart_create(lv_screen_active());
    lv_chart_set_update_mode(chart, LV_CHART_UPDATE_MODE_CIRCULAR);
    lv_obj_set_style_size(chart, 0, 0, LV_PART_INDICATOR);
    lv_obj_set_size(chart, 280, 150);
    lv_obj_center(chart);

    lv_chart_set_point_count(chart, 80);
    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    /*Prefill with data*/
    uint32_t i;
    for(i = 0; i < 80; i++) {
        lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 90));
    }

    lv_timer_create(add_data, 300, chart);
}
#endif
```

3.7.10 Checkbox

Checkbox Semplici

```
#include "../../lv_examples.h"
#if LV_USE_CHECKBOX && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        LV_UNUSED(obj);
        const char * txt = lv_checkbox_get_text(obj);
        const char * state = lv_obj_get_state(obj) & LV_STATE_CHECKED ? "Checked" : "Unchecked";
        LV_UNUSED(txt);
        LV_UNUSED(state);
        LV_LOG_USER("%s: %s", txt, state);
    }
}

void lv_example_checkbox_1(void)
{
    lv_obj_set_flex_flow(lv_screen_active(), LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(lv_screen_active(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER);

    lv_obj_t * cb;
    cb = lv_checkbox_create(lv_screen_active());
    lv_checkbox_set_text(cb, "Apple");
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_screen_active());
    lv_checkbox_set_text(cb, "Banana");
    lv_obj_add_state(cb, LV_STATE_CHECKED);
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_screen_active());
    lv_checkbox_set_text(cb, "Lemon");
    lv_obj_add_state(cb, LV_STATE_DISABLED);
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_screen_active());
    lv_obj_add_state(cb, LV_STATE_CHECKED);
    lv_obj_add_state(cb, LV_STATE_DISABLED);
    lv_checkbox_set_text(cb, "Melon\nand a new line");
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    lv_obj_update_layout(cb);
}
```

Checkbox come radio button

```
#include "../../lv_examples.h"
#if LV_USE_CHECKBOX && LV_BUILD_EXAMPLES

static lv_style_t style_radio;
static lv_style_t style_radio_chk;
static int32_t active_index_1 = 0;
static int32_t active_index_2 = 0;

static void radio_event_handler(lv_event_t * e)
{
    int32_t * active_id = (int32_t *)lv_event_get_user_data(e);
    lv_obj_t * cont = (lv_obj_t *)lv_event_get_current_target(e);
    lv_obj_t * act_cb = lv_event_get_target_obj(e);
    lv_obj_t * old_cb = lv_obj_get_child(cont, *active_id);

    /*Do nothing if the container was clicked*/
    if(act_cb == cont) return;

    lv_obj_remove_state(old_cb, LV_STATE_CHECKED); /*Uncheck the previous radio button*/
    lv_obj_add_state(act_cb, LV_STATE_CHECKED); /*Check the current radio button*/

    *active_id = lv_obj_get_index(act_cb);

    LV_LOG_USER("Selected radio buttons: %d, %d, (%int)active_index_1, (%int)active_index_2");
}

static void radiobutton_create(lv_obj_t * parent, const char * txt)
{
    lv_obj_t * obj = lv_checkbox_create(parent);
    lv_checkbox_set_text(obj, txt);
    lv_obj_add_flag(obj, LV_OBJ_FLAG_EVENT_BUBBLE);
    lv_obj_add_style(obj, &style_radio, LV_PART_INDICATOR);
    lv_obj_add_style(obj, &style_radio_chk, LV_PART_INDICATOR | LV_STATE_CHECKED);
}

/**
 * Checkboxes as radio buttons
 */
void lv_example_checkbox_2(void)
{
    /* The idea is to enable `LV_OBJ_FLAG_EVENT_BUBBLE` on checkboxes and process the
     * `LV_EVENT_CLICKED` on the container.
     * A variable is passed as event user data where the index of the active
     * radiobutton is saved */

    lv_style_init(&style_radio);
    lv_style_set_radius(&style_radio, LV_RADIUS_CIRCLE);

    lv_style_init(&style_radio_chk);
    lv_style_set_bg_image_src(&style_radio_chk, NULL);

    uint32_t i;
    char buf[32];

    lv_obj_t * cont1 = lv_obj_create(lv_screen_active());
    lv_obj_set.flex_flow(cont1, LV_FLEX_FLOW_COLUMN);
    lv_obj.set_size(cont1, lv_pct(40), lv_pct(60));
    lv_obj.add_event_cb(cont1, radio_event_handler, LV_EVENT_CLICKED, &active_index_1);

    for(i = 0; i < 5; i++) {
        lv_snprintf(buf, sizeof(buf), "A %d", (int)i + 1);
        radiobutton_create(cont1, buf);
    }

    /*Make the first checkbox checked*/
    lv_obj_add_state(lv_obj_get_child(cont1, 0), LV_STATE_CHECKED);

    lv_obj_t * cont2 = lv_obj_create(lv_screen_active());
    lv_obj.set.flex_flow(cont2, LV_FLEX_FLOW_COLUMN);
    lv_obj.set.size(cont2, lv_pct(40), lv_pct(80));
    lv_obj.set.x(cont2, lv_pct(50));
    lv_obj.add_event_cb(cont2, radio_event_handler, LV_EVENT_CLICKED, &active_index_2);

    for(i = 0; i < 3; i++) {
        lv_snprintf(buf, sizeof(buf), "B %d", (int)i + 1);
        radiobutton_create(cont2, buf);
    }

    /*Make the first checkbox checked*/
    lv_obj_add_state(lv_obj_get_child(cont2, 0), LV_STATE_CHECKED);
}

#endif
```

3.7.11 Menù a discesa

Menù a discesa semplice

```
#include "../../lv_examples.h"
#if LV_USE_DROPDOWN && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_dropdown_get_selected_str(obj, buf, sizeof(buf));
        LV_LOG_USER("Option: %s", buf);
    }
}

void lv_example_dropdown_1(void)
{
```

(continues on next page)

(continua dalla pagina precedente)

```

/*Create a normal drop down list*/
lv_obj_t * dd = lv_dropdown_create(lv_screen_active());
lv_dropdown_set_options(dd, "Apple\n"
                        "Banana\n"
                        "Orange\n"
                        "Cherry\n"
                        "Grape\n"
                        "Raspberry\n"
                        "Melon\n"
                        "Orange\n"
                        "Lemon\n"
                        "Nuts");

lv_obj_align(dd, LV_ALIGN_TOP_MID, 0, 20);
lv_obj_add_event_cb(dd, event_handler, LV_EVENT_ALL, NULL);
}

#endif

```

Menù a discesa nelle quattro direzioni

```

#include "../../lv_examples.h"
#if LV_USE_DROPDOWN && LV_BUILD_EXAMPLES

/***
 * Create a drop down, up, left and right menus
 */
void lv_example_dropdown_2(void)
{
    static const char * opts = "Apple\n"
                                "Banana\n"
                                "Orange\n"
                                "Melon";

    lv_obj_t * dd;
    dd = lv_dropdown_create(lv_screen_active());
    lv_dropdown_set_options_static(dd, opts);
    lv_obj_align(dd, LV_ALIGN_TOP_MID, 0, 10);

    dd = lv_dropdown_create(lv_screen_active());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_BOTTOM);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_UP);
    lv_obj_align(dd, LV_ALIGN_BOTTOM_MID, 0, -10);

    dd = lv_dropdown_create(lv_screen_active());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_RIGHT);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_RIGHT);
    lv_obj_align(dd, LV_ALIGN_LEFT_MID, 10, 0);

    dd = lv_dropdown_create(lv_screen_active());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_LEFT);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_LEFT);
    lv_obj_align(dd, LV_ALIGN_RIGHT_MID, -10, 0);
}

#endif

```

Menù

```

#include "../../lv_examples.h"
#if LV_USE_DROPDOWN && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * dropdown = lv_event_get_target_obj(e);
    char buf[64];
    lv_dropdown_get_selected_str(dropdown, buf, sizeof(buf));
    LV_LOG_USER("%s is selected", buf);
}

/***
 * Create a menu from a drop-down list and show some drop-down list features and styling
 */
void lv_example_dropdown_3(void)
{
    /*Create a drop down list*/
    lv_obj_t * dropdown = lv_dropdown_create(lv_screen_active());
    lv_obj_align(dropdown, LV_ALIGN_TOP_LEFT, 10, 10);
    lv_dropdown_set_options(dropdown, "New project\n"
                            "New file\n"
                            "Save\n"
                            "Save as ...\n"
                            "Open project\n"
                            "Recent projects\n"
                            "Preferences\n"
                            "Exit");

    /*Set a fixed text to display on the button of the drop-down list*/
    lv_dropdown_set_text(dropdown, "Menu");

    /*Use a custom image as down icon and flip it when the list is opened*/
    LV_IMAGE_DECLARE(img_caret_down);
    lv_dropdown_set_symbol(dropdown, &img_caret_down);
    lv_obj_set_style_transform_rotation(dropdown, 1800, LV_PART_INDICATOR | LV_STATE_CHECKED);

    /*In a menu we don't need to show the last clicked item*/
    lv_dropdown_set_selected_highlight(dropdown, false);

    lv_obj_add_event_cb(dropdown, event_cb, LV_EVENT_VALUE_CHANGED, NULL);
}

```

(continues on next page)

(continua dalla pagina precedente)

#endif

3.7.12 Immagine

Immagine da variabile e simbolo

```
#include "../../lv_examples.h"
#if LV_USE_IMAGE && LV_BUILD_EXAMPLES

void lv_example_image_1(void)
{
    LV_IMAGE_DECLARE(img_cogwheel_argb);
    lv_obj_t * img1 = lv_image_create(lv_screen_active());
    lv_image_set_src(img1, &img_cogwheel_argb);
    lv_obj_align(img1, LV_ALIGN_CENTER, 0, 0);

    lv_obj_t * img2 = lv_image_create(lv_screen_active());
    lv_image_set_src(img2, LV_SYMBOL_OK "Accept");
    lv_obj_align_to(img2, img1, LV_ALIGN_OUT_BOTTOM_MID, 0, 20);
}

#endif
```

Ricolorazione dell'immagine

```
#include "../../lv_examples.h"
#if LV_USE_IMAGE && LV_USE_SLIDER && LV_BUILD_EXAMPLES

static lv_obj_t * create_slider(lv_color_t color);
static void slider_event_cb(lv_event_t * e);

static lv_obj_t * red_slider, * green_slider, * blue_slider, * intense_slider;
static lv_obj_t * img1;

/***
 * Demonstrate runtime image re-coloring
 */
void lv_example_image_2(void)
{
    /*Create 4 sliders to adjust RGB color and re-color intensity*/
    red_slider = create_slider(lv_palette_main(LV_PALETTE_RED));
    green_slider = create_slider(lv_palette_main(LV_PALETTE_GREEN));
    blue_slider = create_slider(lv_palette_main(LV_PALETTE_BLUE));
    intense_slider = create_slider(lv_palette_main(LV_PALETTE_GREY));

    lv_slider_set_value(red_slider, LV_OPA_20, LV_ANIM_OFF);
    lv_slider_set_value(green_slider, LV_OPA_90, LV_ANIM_OFF);
    lv_slider_set_value(blue_slider, LV_OPA_60, LV_ANIM_OFF);
    lv_slider_set_value(intense_slider, LV_OPA_50, LV_ANIM_OFF);

    lv_obj_align(red_slider, LV_ALIGN_LEFT_MID, 25, 0);
    lv_obj_align_to(green_slider, red_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);
    lv_obj_align_to(blue_slider, green_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);
    lv_obj_align_to(intense_slider, blue_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);

    /*Now create the actual image*/
    LV_IMAGE_DECLARE(img_cogwheel_argb);
    img1 = lv_image_create(lv_screen_active());
    lv_image_set_src(img1, &img_cogwheel_argb);
    lv_obj_align(img1, LV_ALIGN_RIGHT_MID, -20, 0);

    lv_obj_send_event(intense_slider, LV_EVENT_VALUE_CHANGED, NULL);
}

static void slider_event_cb(lv_event_t * e)
{
    LV_UNUSED(e);

    /*Recolor the image based on the sliders' values*/
    lv_color_t color = lv_color_make((uint8_t)lv_slider_get_value(red_slider),
                                    (uint8_t)lv_slider_get_value(green_slider),
                                    (uint8_t)lv_slider_get_value(blue_slider));
    lv_opa_t intense = (lv_opa_t)lv_slider_get_value(intense_slider);
    lv_obj_set_style_image_recolor_opa(img1, intense, 0);
    lv_obj_set_style_image_recolor(img1, color, 0);
}

static lv_obj_t * create_slider(lv_color_t color)
{
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_slider_set_range(slider, 0, 255);
    lv_obj_set_size(slider, 10, 200);
    lv_obj_set_style_bg_color(slider, color, LV_PART_KNOB);
    lv_obj_set_style_bg_color(slider, lv_color_darken(color, LV_OPA_40), LV_PART_INDICATOR);
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    return slider;
}

#endif
```

Rotazione e zoom

```
#include "../../lv_examples.h"
#if LV_USE_IMAGE && LV_BUILD_EXAMPLES

static void set_angle(void * img, int32_t v)
{
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_image_set_rotation((lv_obj_t *img, v);
}

static void set_scale(void * img, int32_t v)
{
    lv_image_set_scale((lv_obj_t *)img, v);
}

/**
 * Show transformations (zoom and rotation) using a pivot point.
 */
void lv_example_image_3(void)
{
    LV_IMAGE_DECLARE(img_cogwheel_argb);

    /*Now create the actual image*/
    lv_obj_t * img = lv_image_create(lv_screen_active());
    lv_image_set_src(img, &img_cogwheel_argb);
    lv_obj_align(img, LV_ALIGN_CENTER, 50, 50);
    lv_image_set_pivot(img, 0, 0); /*Rotate around the top left corner*/

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, img);
    lv_anim_set_exec_cb(&a, set_angle);
    lv_anim_set_values(&a, 0, 3600);
    lv_anim_set_duration(&a, 5000);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);

    lv_anim_set_exec_cb(&a, set_scale);
    lv_anim_set_values(&a, 128, 256);
    lv_anim_set_reverse_duration(&a, 3000);
    lv_anim_start(&a);
}
#endif

```

Offset e stile dell'immagine

```

#include "../../lv_examples.h"
#if LV_USE_IMAGE && LV_BUILD_EXAMPLES

static void ofs_y_anim(void * img, int32_t v)
{
    lv_image_set_offset_y((lv_obj_t *)img, v);
}

/**
 * Image styling and offset
 */
void lv_example_image_4(void)
{
    LV_IMAGE_DECLARE(img_skew_strip);

    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_image_recolor_opa(&style, LV_OPA_COVER);
    lv_style_set_image_recolor(&style, lv_color_black());

    lv_obj_t * img = lv_image_create(lv_screen_active());
    lv_obj_add_style(img, &style, 0);
    lv_image_set_src(img, &img_skew_strip);
    lv_obj_set_size(img, 150, 100);
    lv_obj_center(img);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, img);
    lv_anim_set_exec_cb(&a, ofs_y_anim);
    lv_anim_set_values(&a, 0, 100);
    lv_anim_set_duration(&a, 3000);
    lv_anim_set_reverse_duration(&a, 500);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);
}

#endif

```

3.7.13 Pulsante Immagine

Pulsante immagine semplice

```

#include "../../lv_examples.h"
#if LV_USE_IMAGEBUTTON && LV_BUILD_EXAMPLES

void lv_example_imagebutton_1(void)
{
    LV_IMAGE_DECLARE(imagebutton_left);
    LV_IMAGE_DECLARE(imagebutton_right);
    LV_IMAGE_DECLARE(imagebutton_mid);

    /*Create a transition animation on width transformation and recolor.*/
    static lv_style_prop_t tr_prop[] = {LV_STYLE_TRANSFORM_WIDTH, LV_STYLE_IMAGE_RECOLOR_OPA, 0};
    static lv_style_transition_dsc_t tr;
    lv_style_transition_dsc_init(&tr, tr_prop, lv_anim_path_linear, 200, 0, NULL);

    static lv_style_t style_def;
    lv_style_init(&style_def);

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_set_text_color(&style_def, lv_color_white());
lv_style_set_transition(&style_def, &tr);

/*Darken the button when pressed and make it wider*/
static lv_style_t style_pr;
lv_style_init(&style_pr);
lv_style_set_image_recolor_opa(&style_pr, LV_OPA_30);
lv_style_set_image_recolor(&style_pr, lv_color_black());
lv_style_set_transform_width(&style_pr, 20);

/*Create an image button*/
lv_obj_t * imagebutton1 = lv_imagebutton_create(lv_screen_active());
lv_imagebutton_set_src(imagebutton1, LV_IMAGEBUTTON_STATE_RELEASED, &imagebutton_left, &imagebutton_mid,
                      &imagebutton_right);
lv_obj_add_style(imagebutton1, &style_def, 0);
lv_obj_add_style(imagebutton1, &style_pr, LV_STATE_PRESSED);

lv_obj_set_width(imagebutton1, 100);
lv_obj_align(imagebutton1, LV_ALIGN_CENTER, 0, 0);

/*Create a label on the image button*/
lv_obj_t * label = lv_label_create(imagebutton1);
lv_label_set_text(label, "Button");
lv_obj_align(label, LV_ALIGN_CENTER, 0, -4);
}

#endif

```

3.7.14 Tastiera

Tastiera con area di testo

```
#include "../../lv_examples.h"
#ifndef LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * ta = lv_event_get_target_obj(e);
    lv_obj_t * kb = (lv_obj_t *)lv_event_get_user_data(e);
    if(code == LV_EVENT_FOCUSED) {
        lv_keyboard_set_textarea(kb, ta);
        lv_obj_add_flag(kb, LV_OBJ_FLAG_HIDDEN);
    }

    if(code == LV_EVENT_DEFOCUSSED) {
        lv_keyboard_set_textarea(kb, NULL);
        lv_obj_add_flag(kb, LV_OBJ_FLAG_HIDDEN);
    }
}

void lv_example_keyboard_1(void)
{
    /*Create a keyboard to use it with an of the text areas*/
    lv_obj_t * kb = lv_keyboard_create(lv_screen_active());

    /*Create a text area. The keyboard will write here*/
    lv_obj_t * ta1;
    ta1 = lv_textarea_create(lv_screen_active());
    lv_obj_align(ta1, LV_ALIGN_TOP_LEFT, 10, 10);
    lv_obj_add_event_cb(ta1, ta_event_cb, LV_EVENT_ALL, kb);
    lv_textarea_set_placeholder_text(ta1, "Hello");
    lv_obj_set_size(ta1, 140, 80);

    lv_obj_t * ta2;
    ta2 = lv_textarea_create(lv_screen_active());
    lv_obj_align(ta2, LV_ALIGN_TOP_RIGHT, -10, 10);
    lv_obj_add_event_cb(ta2, ta_event_cb, LV_EVENT_ALL, kb);
    lv_obj_set_size(ta2, 140, 80);

    lv_keyboard_set_textarea(kb, ta1);

    /*The keyboard will show Arabic characters if they are enabled*/
    #if LV_USE_ARABIC_PERSIAN_CHARS && LV_FONT_DEJAVU_16_PERSIAN_H
    lv_obj_set_style_text_font(kb, &lv_font_dejavu_16_persian_h);
    lv_obj_set_style_text_font(ta1, &lv_font_dejavu_16_persian_h);
    lv_obj_set_style_text_font(ta2, &lv_font_dejavu_16_persian_h);
    #endif
}
#endif
```

Tastiera con mappa personalizzata

(continues on next page)

(continua dalla pagina precedente)

```

→ BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_
→ BUTTONMATRIX_CTRL_WIDTH_4,
    LV_BUTTONMATRIX_CTRL_WIDTH_2, (lv_buttonmatrix_ctrl_t)(LV_BUTTONMATRIX_CTRL_HIDDEN | LV_BUTTONMATRIX_CTRL_WIDTH_2), LV_BUTTONMATRIX_
→ CTRL_WIDTH_6, (lv_buttonmatrix_ctrl_t)(LV_BUTTONMATRIX_CTRL_HIDDEN | LV_BUTTONMATRIX_CTRL_WIDTH_2), LV_BUTTONMATRIX_CTRL_WIDTH_2
};

/*Create a keyboard and add the new map as USER_1 mode*/
lv_obj_t * kb = lv_keyboard_create(lv_screen_active());

lv_keyboard_set_map(kb, LV_KEYBOARD_MODE_USER_1, kb_map, kb_ctrl);

lv_keyboard_set_mode(kb, LV_KEYBOARD_MODE_USER_1);

/*Create a text area. The keyboard will write here*/
lv_obj_t * ta;
ta = lv_textarea_create(lv_screen_active());
lv_obj_align(ta, LV_ALIGN_TOP_MID, 0, 10);
lv_obj_set_size(ta, lv_pct(90), 80);
lv_obj_add_state(ta, LV_STATE_FOCUSED);

lv_keyboard_set_textarea(kb, ta);
}

#endif

```

Tastiera con disegno

```

#include "../../lv_examples.h"
#if LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    bool pressed = false;
    if(lv_keyboard_get_selected_button(obj) == base_dsc->idl && lv_obj_has_state(obj, LV_STATE_PRESSED)) {
        pressed = true;
    }

    /*When the keyboard draws the buttons...*/
    if(base_dsc->part == LV_PART_ITEMS) {
        /*Get a color based on the button's index*/
        lv_palette_t palette = (lv_palette_t)base_dsc->id1 % LV_PALETTE_LAST;
        lv_draw_fill_dsc_t * fill_draw_dsc = lv_draw_task_get_fill_dsc(draw_task);
        if(fill_draw_dsc) {
            fill_draw_dsc->color = pressed ? lv_palette_darken(palette, 3) : lv_palette_main(palette);
        }
        lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
        if(label_draw_dsc) {
            /*For the OK symbol, draw a star instead*/
            if(lv_strocpy(label_draw_dsc->text, LV_SYMBOL_OK) == 0) {
                label_draw_dsc->opa = 0; /*Hide the label*/
                LV_IMAGE_DECLARE(img_star);
                lv_image_header_t header;
                lv_result_t res = lv_image_decoder_get_info(&img_star, &header);
                if(res != LV_RESULT_OK) return;

                lv_area_t a;
                lv_area_set(&a, 0, 0, header.w - 1, header.h - 1);
                lv_area_t draw_task_area;
                lv_draw_task_get_area(draw_task, &draw_task_area);
                lv_area_align(&draw_task_area, &a, LV_ALIGN_CENTER, 0, 0);

                lv_draw_image_dsc_t img_draw_dsc;
                lv_draw_image_dsc_init(&img_draw_dsc);
                img_draw_dsc.src = &img_star;
                lv_draw_image(base_dsc->layer, &img_draw_dsc, &a);
            }
            else {
                /*For the other labels just pick an lighter color*/
                label_draw_dsc->color = lv_palette_lighten(palette, 4);
            }
        }
    }
}

/***
 * Add custom drawer to the keyboard to customize the buttons one by one
 */
void lv_example_keyboard_3(void)
{
    lv_obj_t * kb = lv_keyboard_create(lv_screen_active());
    lv_obj_add_event_cb(kb, event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(kb, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
    lv_obj_center(kb);
}
#endif

```

3.7.15 Etichetta

A capo automatico, ricolorazione e scorrimento

```

#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES

/***
 * Show line wrap, re-color, line align and text scrolling.
 */
void lv_example_label_1(void)
{

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * labell = lv_label_create(lv_screen_active());
lv_label_set_long_mode(labell, LV_LABEL_LONG_MODE_WRAP);      /*Break the long lines*/
lv_label_set_recolor(labell, true);                            /*Enable re-coloring by commands in the text*/
lv_label_set_text(labell, "#0000ff Re-color #fff0ff words #ff0000 of aa label, align the lines to the center */
                "and wrap long text automatically.");
lv_obj_set_width(labell, 150); /*Set smaller width to make the lines wrap*/
lv_obj_set_style_text_align(labell, LV_TEXT_ALIGN_CENTER, 0);
lv_obj_align(labell, LV_ALIGN_CENTER, 0, -40);

lv_obj_t * label2 = lv_label_create(lv_screen_active());
lv_label_set_long_mode(label2, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);    /*Circular scroll*/
lv_obj_set_width(label2, 150);
lv_label_set_text(label2, "It is a circularly scrolling text. ");
lv_obj_align(label2, LV_ALIGN_CENTER, 0, 40);

}

#endif

```

Ombreggiatura del testo

```
#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES

/***
 * Create a fake text shadow
 */
void lv_example_label_2(void)
{
    /*Create a style for the shadow*/
    static lv_style_t style_shadow;
    lv_style_init(&style_shadow);
    lv_style_set_text_opa(&style_shadow, LV_OPA_30);
    lv_style_set_text_color(&style_shadow, lv_color_black());

    /*Create a label for the shadow first (it's in the background)*/
    lv_obj_t * shadow_label = lv_label_create(lv_screen_active());
    lv_obj_add_style(shadow_label, &style_shadow, 0);

    /*Create the main label*/
    lv_obj_t * main_label = lv_label_create(lv_screen_active());
    lv_label_set_text(main_label, "A simple method to create\n"
                        "shadows on a text.\n"
                        "It even works with\n"
                        "newlines and spaces.");
    lv_label_set_text(shadow_label, lv_label_get_text(main_label));

    /*Position the main label*/
    lv_obj_align(main_label, LV_ALIGN_CENTER, 0, 0);

    /*Shift the second label down and to the right by 2 pixel*/
    lv_obj_align_to(shadow_label, main_label, LV_ALIGN_TOP_LEFT, 2, 2
}

#endif
```

Visualizzazione di testi LTR, RTL e in cinese

Disegno di etichette con gradiente di colore

```
#include "../../lv_examples.h"

#if LV_USE_LABEL && LV_FONT_MONTserrat_24 && LV_USE_CANVAS && LV_BUILD_EXAMPLES && LV_DRAW_SW_COMPLEX

#define MASK_WIDTH 150
#define MASK_HEIGHT 60
```

(continues on next page)

(continua dalla pagina precedente)

```

static void generate_mask(lv_draw_buf_t * mask, int32_t w, int32_t h, const char * txt)
{
    /*Create a "8 bit alpha" canvas and clear it*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, mask);
    lv_canvas_fill_bg(canvas, lv_color_black(), LV_OPA_TRANSP);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    /*Draw a label to the canvas. The result "image" will be used as mask*/
    lv_draw_label_dsc_t label_dsc;
    lv_draw_label_dsc_init(&label_dsc);
    label_dsc.color = lv_color_white();
    label_dsc.align = LV_TEXT_ALIGN_CENTER;
    label_dsc.text = txt;
    label_dsc.font = &lv_font_montserrat_24;
    lv_area_t a = {0, 0, w - 1, h - 1};
    lv_draw_label(&layer, &label_dsc, &a);

    lv_canvas_finish_layer(canvas, &layer);

    lv_obj_delete(canvas);
}

/***
 * Draw label with gradient color
 */
void lv_example_label_4(void)
{
    /* Create the mask of a text by drawing it to a canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(mask, MASK_WIDTH, MASK_HEIGHT, LV_COLOR_FORMAT_L8);
    LV_DRAW_BUF_INIT_STATIC(mask);

    generate_mask(&mask, MASK_WIDTH, MASK_HEIGHT, "Text with gradient");

    /* Create an object from where the text will be masked out.
     * Now it's a rectangle with a gradient but it could be an image too*/
    lv_obj_t * grad = lv_obj_create(lv_screen_active());
    lv_obj_set_size(grad, MASK_WIDTH, MASK_HEIGHT);
    lv_obj_center(grad);
    lv_obj_set_style_bg_color(grad, lv_color_hex(0xffff0000), 0);
    lv_obj_set_style_bg_grad_color(grad, lv_color_hex(0x0000ff), 0);
    lv_obj_set_style_bg_grad_dir(grad, LV_GRAD_DIR_HOR, 0);
    lv_obj_set_style_bitmap_src(grad, &mask, 0);
}
#endif

```

Personalizzazione dell'animazione a scorrimento circolare

```

#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES

/***
 * Show customizing the circular scrolling animation of a label with 'LV_LABEL_LONG_MODE_SCROLL_CIRCULAR'
 * long mode.
 */
void lv_example_label_5(void)
{
    static lv_anim_t animation_template;
    static lv_style_t label_style;

    lv_anim_init(&animation_template);
    lv_anim_set_delay(&animation_template, 1000);           /*Wait 1 second to start the first scroll*/
    lv_anim_set_repeat_delay(&animation_template,
                           3000);           /*Repeat the scroll 3 seconds after the label scrolls back to the initial position*/
    lv_anim_set_repeat_count(&animation_template, LV_ANIM_REPEAT_INFINITE);

    /*Initialize the label style with the animation template*/
    lv_style_init(&label_style);
    lv_style_set_anim(&label_style, &animation_template);

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);           /*Circular scroll*/
    lv_obj_set_width(label, 150);
    lv_label_set_text(label, "It is a circularly scrolling text. ");
    lv_obj_align(label, LV_ALIGN_CENTER, 0, 40);
    lv_obj_add_style(label, &label_style, LV_STATE_DEFAULT);           /*Add the style to the label*/
}
#endif

```

Carattere a spaziatura fissa

```

#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES && LV_FONT_MONTSERAT_20

static bool fix_w_get_glyph_dsc(const lv_font_t * font, lv_font_glyph_dsc_t * dsc, uint32_t letter,
                               uint32_t letter_next)
{
    bool ret = lv_font_get_glyph_dsc_fmt_txt(font, dsc, letter, letter_next);
    if(!ret) return false;

    /* Set a fixed width */
    dsc->adv_w = 20;
    dsc->ofs_x = (dsc->adv_w - dsc->box_w) / 2;
    return true;
}

void lv_example_label_6(void)
{
    /* Clone the original font and override its behavior */
    static lv_font_t mono_font;

```

(continues on next page)

(continua dalla pagina precedente)

```

mono_font = lv_font_montserrat_20;
mono_font.get_glyph_dsc = fix_w_get_glyph_dsc;

/* Create a label with normal font */
lv_obj_t * label1 = lv_label_create(lv_screen_active());
lv_obj_set_style_text_font(label1, &lv_font_montserrat_20, 0);
lv_label_set_text(label1, "0123.Wabc");

/* Create a label with fixed-width glyph descriptor override */
lv_obj_t * label2 = lv_label_create(lv_screen_active());
lv_obj_set_y(label2, 30);
lv_obj_set_style_text_font(label2, &mono_font, 0);
lv_label_set_text(label2, "0123.Wabc");
}

#endif

```

Assegnare un tag di traduzione a un'etichetta

```

#include "../../lv_examples.h"

#if LV_USE_TRANSLATION && LV_USE_DROPDOWN && LV_USE_LABEL && LV_BUILD_EXAMPLES

/* Arrays are defined `const` to place them in program space instead of RAM. */
static const char * const tags[] = {"tiger", "lion", "rabbit", "elephant", NULL};
static const char * const languages[] = {"English", "Deutsch", "Espanol", NULL};

static void add_static_translations(void)
{
    static const char * const translations[] = {
        "The Tiger", "Der Tiger", "El Tigre",
        "The Lion", "Der Löwe", "El León",
        "The Rabbit", "Das Kaninchen", "El Conejo",
        "The Elephant", "Der Elefant", "El Elefante",
    };
    lv_translation_add_static(languages, tags, translations);
}

static void language_change_cb(lv_event_t * e)
{
    static char selected_lang[20];

    lv_obj_t * dropdown = lv_event_get_target_obj(e);
    lv_dropdown_get_selected_str(dropdown, selected_lang, sizeof(selected_lang));
    lv_translation_set_language(selected_lang);
}

/**
 * Use a translation tag in labels
 */
void lv_example_label_7(void)
{
    lv_obj_set_flex_flow(lv_screen_active(), LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(lv_screen_active(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    add_static_translations();
    const size_t tag_count = sizeof(tags) / sizeof(tags[0]) - 1;
    const size_t lang_count = sizeof(languages) / sizeof(languages[0]) - 1;

    /* Create a dropdown to be able to select the language */
    lv_obj_t * language_dropdown = lv_dropdown_create(lv_screen_active());
    lv_dropdown_clear_options(language_dropdown);

    for(size_t i = 0; i < lang_count; ++i) {
        lv_dropdown_add_option(language_dropdown, languages[i], i);
    }

    lv_obj_add_event_cb(language_dropdown, language_change_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_translation_set_language("English");

    /* Create a label for each tag */
    for(size_t i = 0; i < tag_count; ++i) {
        lv_obj_t * label = lv_label_create(lv_screen_active());
        lv_label_set_translation_tag(label, tags[i]);
    }
}

#endif /*LV_USE_TRANSLATION && LV_USE_DROPDOWN && LV_USE_LABEL && LV_BUILD_EXAMPLES*/

```

3.7.16 LED

LED con stile personalizzato

```

#include "../../lv_examples.h"
#ifndef LV_USE_LED && LV_BUILD_EXAMPLES

/***
 * Create LED's with different brightness and color
 */
void lv_example_led_1(void)
{
    /*Create a LED and switch it OFF*/
    lv_obj_t * led1 = lv_led_create(lv_screen_active());
    lv_obj_align(led1, LV_ALIGN_CENTER, -80, 0);
    lv_led_off(led1);

    /*Copy the previous LED and set a brightness*/
    lv_obj_t * led2 = lv_led_create(lv_screen_active());
    lv_obj_align(led2, LV_ALIGN_CENTER, 0, 0);
    lv_led_set_brightness(led2, 150);
    lv_led_set_color(led2, lv_palette_main(LV_PALETTE_RED));
}

```

(continues on next page)

(continua dalla pagina precedente)

```

/*Copy the previous LED and switch it ON*/
lv_obj_t * led3 = lv_led_create(lv_screen_active());
lv_obj_align(led3, LV_ALIGN_CENTER, 80, 0);
lv_led_on(led3);
}

#endif

```

3.7.17 Line

Linea semplice

```

#include "../../lv_examples.h"
#if LV_USE_LINE && LV_BUILD_EXAMPLES

void lv_example_line_1(void)
{
    /*Create an array for the points of the line*/
    static lv_point_precise_t line_points[] = { {5, 5}, {70, 70}, {120, 10}, {180, 60}, {240, 10} };

    /*Create style*/
    static lv_style_t style_line;
    lv_style_init(&style_line);
    lv_style_set_line_width(&style_line, 8);
    lv_style_set_line_color(&style_line, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_line_rounded(&style_line, true);

    /*Create a line and apply the new style*/
    lv_obj_t * line1;
    line1 = lv_line_create(lv_screen_active());
    lv_line_set_points(line1, line_points, 5);      /*Set the points*/
    lv_obj_add_style(line1, &style_line, 0);
    lv_obj_center(line1);
}

#endif

```

3.7.18 Lista

Lista semplice

```

#include "../../lv_examples.h"
#if LV_USE_LIST && LV_BUILD_EXAMPLES
static lv_obj_t * list1;

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_CLICKED) {
        LV_UNUSED(obj);
        LV_LOG_USER("Clicked: %s", lv_list_get_button_text(list1, obj));
    }
}
void lv_example_list_1(void)
{
    /*Create a list*/
    list1 = lv_list_create(lv_screen_active());
    lv_obj_set_size(list1, 180, 220);
    lv_obj_center(list1);

    /*Add buttons to the list*/
    lv_obj_t * btn;
    lv_list_add_text(list1, "File");
    btn = lv_list_add_button(list1, LV_SYMBOL_FILE, "New");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_DIRECTORY, "Open");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_SAVE, "Save");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_CLOSE, "Delete");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_EDIT, "Edit");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_list_add_text(list1, "Connectivity");
    btn = lv_list_add_button(list1, LV_SYMBOL_BLUETOOTH, "Bluetooth");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_GPS, "Navigation");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_USB, "USB");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_BATTERY_FULL, "Battery");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_list_add_text(list1, "Exit");
    btn = lv_list_add_button(list1, LV_SYMBOL_OK, "Apply");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_CLOSE, "Close");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
}

#endif

```

Ordinamento di un elenco utilizzando i pulsanti Su e Giù

```
#include "../../lv_examples.h"
#if LV_USE_LIST && LV_BUILD_EXAMPLES

static lv_obj_t * list1;
static lv_obj_t * list2;

static lv_obj_t * currentButton = NULL;

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_CLICKED) {
        LV_LOG_USER("Clicked: %s", lv_list_get_button_text(list1, obj));

        if(currentButton == obj) {
            currentButton = NULL;
        } else {
            currentButton = obj;
        }
        lv_obj_t * parent = lv_obj_get_parent(obj);
        int32_t i;
        for(i = 0; i < (int32_t)lv_obj_get_child_count(parent); i++) {
            lv_obj_t * child = lv_obj_get_child(parent, i);
            if(child == currentButton) {
                lv_obj_add_state(child, LV_STATE_CHECKED);
            } else {
                lv_obj_remove_state(child, LV_STATE_CHECKED);
            }
        }
    }
}

static void event_handler_top(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_CLICKED) {
        if(currentButton == NULL) return;
        lv_obj_move_to_index(currentButton, 0);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_up(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT)) {
        if(currentButton == NULL) return;
        int32_t index = lv_obj_get_index(currentButton);
        if(index < 0) return;
        lv_obj_move_to_index(currentButton, index - 1);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_center(lv_event_t * e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT)) {
        if(currentButton == NULL) return;

        lv_obj_t * parent = lv_obj_get_parent(currentButton);
        const int32_t pos = (int32_t)lv_obj_get_child_count(parent) / 2;

        lv_obj_move_to_index(currentButton, pos);

        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_dn(lv_event_t * e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT)) {
        if(currentButton == NULL) return;
        const int32_t index = lv_obj_get_index(currentButton);

        lv_obj_move_to_index(currentButton, index + 1);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_bottom(lv_event_t * e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_CLICKED) {
        if(currentButton == NULL) return;
        lv_obj_move_to_index(currentButton, -1);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_swap(lv_event_t * e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT)) {
        uint32_t cnt = lv_obj_get_child_count(list1);
        for(int i = 0; i < 100; i++) {
            lv_obj_t * obj = lv_obj_get_child(list1, (int32_t)lv_rand(0, cnt - 1));
            lv_obj_move_to_index(obj, (int32_t)lv_rand(0, cnt - 1));
            if(currentButton != NULL) {
                lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
            }
        }
    }
}

void lv_example_list_2(void)
{
```

(continues on next page)

(continua dalla pagina precedente)

```

/*Create a list*/
list1 = lv_list_create(lv_screen_active());
lv_obj_set_size(list1, lv_pct(60), lv_pct(100));
lv_obj_set_style_pad_row(list1, 5, 0);

/*Add buttons to the list*/
lv_obj_t * btn;
int i;
for(i = 0; i < 15; i++) {
    btn = lv_button_create(list1);
    lv_obj_set_width(btn, lv_pct(50));
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_obj_t * lab = lv_label_create(btn);
    lv_label_set_text_fmt(lab, "Item %d", i);
}

/*Select the first button by default*/
currentButton = lv_obj_get_child(list1, 0);
lv_obj_add_state(currentButton, LV_STATE_CHECKED);

/*Create a second list with up and down buttons*/
list2 = lv_list_create(lv_screen_active());
lv_obj_set_size(list2, lv_pct(40), lv_pct(100));
lv_obj_align(list2, LV_ALIGN_TOP_RIGHT, 0, 0);
lv_obj_set_flex_flow(list2, LV_FLEX_FLOW_COLUMN);

btn = lv_list_add_button(list2, NULL, "Top");
lv_obj_add_event_cb(btn, event_handler_top, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);

btn = lv_list_add_button(list2, LV_SYMBOL_UP, "Up");
lv_obj_add_event_cb(btn, event_handler_up, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);

btn = lv_list_add_button(list2, LV_SYMBOL_LEFT, "Center");
lv_obj_add_event_cb(btn, event_handler_center, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);

btn = lv_list_add_button(list2, LV_SYMBOL_DOWN, "Down");
lv_obj_add_event_cb(btn, event_handler_dn, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);

btn = lv_list_add_button(list2, NULL, "Bottom");
lv_obj_add_event_cb(btn, event_handler_bottom, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);

btn = lv_list_add_button(list2, LV_SYMBOL_SHUFFLE, "Shuffle");
lv_obj_add_event_cb(btn, event_handler_swap, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);
}

#endif

```

3.7.19 Lottie

Caricamento di un'animazione Lottie da un array

```

#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_LOTTIE

/**
 * Load an lottie animation from data
 */
void lv_example_lottie_1(void)
{
    extern const uint8_t lv_example_lottie_approve[];
    extern const size_t lv_example_lottie_approve_size;

    lv_obj_t * lottie = lv_lottie_create(lv_screen_active());
    lv_lottie_set_src_data(lottie, lv_example_lottie_approve, lv_example_lottie_approve_size);

#if LV_DRAW_BUF_ALIGN == 4 && LV_DRAW_BUF_STRIDE_ALIGN == 1
    /*if there are no special requirements, just declare a buffer
     * x4 because the Lottie is rendered in ARGB8888_PREMULTIPLIED format/
    static uint8_t buf[64 * 64 * 4];
    lv_lottie_set_buffer(lottie, 64, 64, buf);
#else
    /*For GPUs and special alignment/stride setting use a draw_buf instead*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, 64, 64, LV_COLOR_FORMAT_ARGB8888_PREMULTIPLIED);
    lv_lottie_set_draw_buf(lottie, &draw_buf);
#endif

    lv_obj_center(lottie);
}

#else

void lv_example_lottie_1(void)
{
    /*fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Lottie cannot be previewed online");
    lv_obj_center(label);
}

#endif /*LV_USE_LOTTIE*/
#endif /*LV_BUILD_EXAMPLES*/

```

Caricamento di un'animazione Lottie da un file

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_LOTTIE

/** Load an lottie animation from file */
void lv_example_lottie_2(void)
{
    lv_obj_t * lottie = lv_lottie_create(lv_screen_active());
    lv_lottie_set_src_file(lottie, "lvg/examples/widgets/lottie/lv_example_lottie_approve.json");

#if LV_DRAW_BUF_ALIGN == 4 && LV_DRAW_BUF_STRIDE_ALIGN == 1
    /*If there are no special requirements, just declare a buffer
     *x4 because the Lottie is rendered in ARGB8888_PREMULTIPLIED format*/
    static uint8_t buf[64 * 64 * 4];
    lv_lottie_set_buffer(lottie, 64, 64, buf);
#else
    /*For GPUs and special alignment/stride setting use a draw_buf instead*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, 64, 64, LV_COLOR_FORMAT_ARGB8888_PREMULTIPLIED);
    lv_lottie_set_draw_buf(lottie, &draw_buf);
#endif

    lv_obj_center(lottie);
}

#else

void lv_example_lottie_2(void)
{
    /*fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Lottie cannot be previewed online");
    lv_obj_center(label);
}

#endif /*LV_USE_LOTTIE*/
#endif /*LV_BUILD_EXAMPLES/
```

3.7.20 Menù

Menù semplice

```
#include "../../lv_examples.h"
#if LV_USE_MENU && LV_BUILD_EXAMPLES

void lv_example_menu_1(void)
{
    /*Create a menu object*/
    lv_obj_t * menu = lv_menu_create(lv_screen_active());
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create a sub page*/
    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(sub_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    /*Create a main page*/
    lv_obj_t * main_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 1");

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 2");

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 3 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_page);

    lv_menu_set_page(menu, main_page);
}

#endif
```

Menù semplice con pulsante principale

```
#include "../../lv_examples.h"
#if LV_USE_MENU && LV_USE_MSGBOX && LV_BUILD_EXAMPLES

static void back_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_obj_t * menu = (lv_obj_t *)lv_event_get_user_data(e);

    if(lv_menu_back_button_is_root(menu, obj)) {
        lv_obj_t * mbox1 = lv_msgbox_create(NULL);
        lv_msgbox_add_title(mbox1, "Hello");
        lv_msgbox_add_text(mbox1, "Root back btn click.");
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

    lv_msgbox_add_close_button(mbox1);
}
}

void lv_example_menu_2(void)
{
    lv_obj_t * menu = lv_menu_create(lv_screen_active());
    lv_menu_set_mode_root_back_button(menu, LV_MENU_ROOT_BACK_BUTTON_ENABLED);
    lv_obj_add_event_cb(menu, back_event_handler, LV_EVENT_CLICKED, menu);
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create a sub page*/
    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(sub_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    /*Create a main page*/
    lv_obj_t * main_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 1");

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 2");

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 3 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_page);

    lv_menu_set_page(menu, main_page);
}

#endif

```

Menù semplice con intestazione personalizzata

```

#include "../lv_examples.h"
#if LV_USE_MENU && LV_BUILD_EXAMPLES

void lv_example_menu_3(void)
{
    /*Create a menu object*/
    lv_obj_t * menu = lv_menu_create(lv_screen_active());
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    /*Modify the header*/
    lv_obj_t * back_btn = lv_menu_get_main_header_back_button(menu);
    lv_obj_t * back_button_label = lv_label_create(back_btn);
    lv_label_set_text(back_button_label, "Back");

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create sub pages*/
    lv_obj_t * sub_1_page = lv_menu_page_create(menu, "Page 1");

    cont = lv_menu_cont_create(sub_1_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    lv_obj_t * sub_2_page = lv_menu_page_create(menu, "Page 2");

    cont = lv_menu_cont_create(sub_2_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    lv_obj_t * sub_3_page = lv_menu_page_create(menu, "Page 3");

    cont = lv_menu_cont_create(sub_3_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    /*Create a main page*/
    lv_obj_t * main_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 1 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_1_page);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 2 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_2_page);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 3 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_3_page);

    lv_menu_set_page(menu, main_page);
}

#endif

```

Menù semplice con pulsante mobile per aggiungere una nuova Menu Page

```
#include "../../lv_examples.h"
#if LV_USE_MENU && LV_BUILD_EXAMPLES

static uint32_t btn_cnt = 1;
static lv_obj_t * main_page;
static lv_obj_t * menu;

static void float_button_event_cb(lv_event_t * e)
{
    LV_UNUSED(e);

    btn_cnt++;

    lv_obj_t * cont;
    lv_obj_t * label;

    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(sub_page);
    label = lv_label_create(cont);
    lv_label_set_text_fmt(label, "Hello, I am hiding inside %" LV_PRIu32 "", btn_cnt);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text_fmt(label, "Item %" LV_PRIu32 "", btn_cnt);
    lv_menu_set_load_page_event(menu, cont, sub_page);

    lv_obj_scroll_to_view_recursive(cont, LV_ANIM_ON);
}

void lv_example_menu_4(void)
{
    /*Create a menu object*/
    menu = lv_menu_create(lv_screen_active());
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create a sub page*/
    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(sub_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding inside the first item");

    /*Create a main page*/
    main_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 1");
    lv_menu_set_load_page_event(menu, cont, sub_page);

    lv_menu_set_page(menu, main_page);

    /*Create floating btn*/
    lv_obj_t * float_btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(float_btn, 50, 50);
    lv_obj_add_flag(float_btn, LV_OBJ_FLAG_FLOATING);
    lv_obj_align(float_btn, LV_ALIGN_BOTTOM_RIGHT, -10, -10);
    lv_obj_add_event_cb(float_btn, float_button_event_cb, LV_EVENT_CLICKED, menu);
    lv_obj_set_style_radius(float_btn, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_bg_image_src(float_btn, LV_SYMBOL_PLUS, 0);
    lv_obj_set_style_text_font(float_btn, lv_theme_get_font_large(float_btn), 0);
}

#endif
```

Menù complesso

```
#include "../../lv_examples.h"
#if LV_USE_MENU && LV_USE_MSGBOX && LV_BUILD_EXAMPLES

typedef enum {
    LV_MENU_ITEM_BUILDER_VARIANT_1,
    LV_MENU_ITEM_BUILDER_VARIANT_2
} lv_menu_builder_variant_t;

static void back_event_handler(lv_event_t * e);
static void switch_handler(lv_event_t * e);
static lv_obj_t * root_page;
static lv_obj_t * create_text(lv_obj_t * parent, const char * icon, const char * txt,
                             lv_menu_builder_variant_t builder_variant);
static lv_obj_t * create_slider(lv_obj_t * parent,
                               const char * icon, const char * txt, int32_t min, int32_t max, int32_t val);
static lv_obj_t * create_switch(lv_obj_t * parent,
                               const char * icon, const char * txt, bool chk);

void lv_example_menu_5(void)
{
    lv_obj_t * menu = lv_menu_create(lv_screen_active());

    lv_color_t bg_color = lv_obj_get_style_bg_color(menu, LV_PART_MAIN);
    if(lv_color_brightness(bg_color) > 127) {
        lv_obj_set_style_bg_color(menu, lv_color_darken(lv_obj_get_style_bg_color(menu, LV_PART_MAIN), 10), 0);
    } else {
        lv_obj_set_style_bg_color(menu, lv_color_darken(lv_obj_get_style_bg_color(menu, LV_PART_MAIN), 50), 0);
    }
    lv_menu_set_mode_root_back_button(menu, LV_MENU_ROOT_BACK_BUTTON_ENABLED);
    lv_obj_add_event_cb(menu, back_event_handler, LV_EVENT_CLICKED, menu);
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * section;
/*Create sub pages*/
lv_obj_t * sub_mechanics_page = lv_menu_page_create(menu, NULL);
lv_obj_set_style_pad_hor(sub_mechanics_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
lv_menu_separator_create(sub_mechanics_page);
section = lv_menu_section_create(sub_mechanics_page);
create_slider(section, LV_SYMBOL_SETTINGS, "Velocity", 0, 150, 120);
create_slider(section, LV_SYMBOL_SETTINGS, "Acceleration", 0, 150, 50);
create_slider(section, LV_SYMBOL_SETTINGS, "Weight limit", 0, 150, 80);

lv_obj_t * sub_sound_page = lv_menu_page_create(menu, NULL);
lv_obj_set_style_pad_hor(sub_sound_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
lv_menu_separator_create(sub_sound_page);
section = lv_menu_section_create(sub_sound_page);
create_switch(section, LV_SYMBOL_AUDIO, "Sound", false);

lv_obj_t * sub_display_page = lv_menu_page_create(menu, NULL);
lv_obj_set_style_pad_hor(sub_display_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
lv_menu_separator_create(sub_display_page);
section = lv_menu_section_create(sub_display_page);
create_slider(section, LV_SYMBOL_SETTINGS, "Brightness", 0, 150, 100);

lv_obj_t * sub_software_info_page = lv_menu_page_create(menu, NULL);
lv_obj_set_style_pad_hor(sub_software_info_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
section = lv_menu_section_create(sub_software_info_page);
create_text(section, NULL, "Version 1.0", LV_MENU_ITEM_BUILDER VARIANT_1);

lv_obj_t * sub_legal_info_page = lv_menu_page_create(menu, NULL);
lv_obj_set_style_pad_hor(sub_legal_info_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
section = lv_menu_section_create(sub_legal_info_page);
for(uint32_t i = 0; i < 15; i++) {
    create_text(section, NULL,
        "This is a long long long long long long long long text, if it is long enough it may scroll.",
        LV_MENU_ITEM_BUILDER VARIANT_1);
}

lv_obj_t * sub_about_page = lv_menu_page_create(menu, NULL);
lv_obj_set_style_pad_hor(sub_about_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
lv_menu_separator_create(sub_about_page);
section = lv_menu_section_create(sub_about_page);
cont = create_text(section, NULL, "Software information", LV_MENU_ITEM_BUILDER VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_software_info_page);
cont = create_text(section, NULL, "Legal information", LV_MENU_ITEM_BUILDER VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_legal_info_page);

lv_obj_t * sub_menu_mode_page = lv_menu_page_create(menu, NULL);
lv_obj_set_style_pad_hor(sub_menu_mode_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
lv_menu_separator_create(sub_menu_mode_page);
section = lv_menu_section_create(sub_menu_mode_page);
cont = create_switch(section, LV_SYMBOL_AUDIO, "Sidebar enable", true);
lv_obj_add_event_cb(lv_obj_get_child(cont, 2), switch_handler, LV_EVENT_VALUE_CHANGED, menu);

/*Create a root page*/
root_page = lv_menu_page_create(menu, "Settings");
lv_obj_set_style_pad_hor(root_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
section = lv_menu_section_create(root_page);
cont = create_text(section, LV_SYMBOL_SETTINGS, "Mechanics", LV_MENU_ITEM_BUILDER VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_mechanics_page);
cont = create_text(section, LV_SYMBOL_AUDIO, "Sound", LV_MENU_ITEM_BUILDER VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_sound_page);
cont = create_text(section, LV_SYMBOL_SETTINGS, "Display", LV_MENU_ITEM_BUILDER VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_display_page);

create_text(root_page, NULL, "Others", LV_MENU_ITEM_BUILDER VARIANT_1);
section = lv_menu_section_create(root_page);
cont = create_text(section, NULL, "About", LV_MENU_ITEM_BUILDER VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_about_page);
cont = create_text(section, LV_SYMBOL_SETTINGS, "Menu mode", LV_MENU_ITEM_BUILDER VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_menu_mode_page);

lv_menu_set_sidebar_page(menu, root_page);

lv_obj_send_event(lv_obj_get_child(lv_obj_get_child(lv_menu_get_cur_sidebar_page(menu), 0), 0), LV_EVENT_CLICKED, NULL);
}

static void back_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_obj_t * menu = (lv_obj_t *)lv_event_get_user_data(e);

    if(lv_menu_back_button_is_root(menu, obj)) {
        lv_obj_t * mbox1 = lv_msgbox_create(NULL);
        lv_msgbox_add_title(mbox1, "Hello");
        lv_msgbox_add_text(mbox1, "Root back btn click.");
        lv_msgbox_add_close_button(mbox1);
    }
}

static void switch_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * menu = (lv_obj_t *)lv_event_get_user_data(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        if(lv_obj_has_state(obj, LV_STATE_CHECKED)) {
            lv_menu_set_page(menu, NULL);
            lv_menu_set_sidebar_page(menu, root_page);
            lv_obj_send_event(lv_obj_get_child(lv_obj_get_child(lv_menu_get_cur_sidebar_page(menu), 0), 0), LV_EVENT_CLICKED, NULL);
        }
        else {
            lv_menu_set_sidebar_page(menu, NULL);
            lv_menu_clear_history(menu); /* Clear history because we will be showing the root page later */
            lv_menu_set_page(menu, root_page);
        }
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

static lv_obj_t * create_text(lv_obj_t * parent, const char * icon, const char * txt,
                             lv_menu_builder_variant_t builder_variant)
{
    lv_obj_t * obj = lv_menu_cont_create(parent);

    lv_obj_t * img = NULL;
    lv_obj_t * label = NULL;

    if(icon) {
        img = lv_image_create(obj);
        lv_image_set_src(img, icon);
    }

    if(txt) {
        label = lv_label_create(obj);
        lv_label_set_text(label, txt);
        lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
        lv_obj_set_flex_grow(label, 1);
    }

    if(builder_variant == LV_MENU_ITEM_BUILDER VARIANT_2 && icon && txt) {
        lv_obj_add_flag(img, LV_OBJ_FLAG_FLEX_IN_NEW_TRACK);
        lv_obj_swap(img, label);
    }

    return obj;
}

static lv_obj_t * create_slider(lv_obj_t * parent, const char * icon, const char * txt, int32_t min, int32_t max,
                               int32_t val)
{
    lv_obj_t * obj = create_text(parent, icon, txt, LV_MENU_ITEM_BUILDER VARIANT_2);

    lv_obj_t * slider = lv_slider_create(obj);
    lv_obj_set_flex_grow(slider, 1);
    lv_slider_set_range(slider, min, max);
    lv_slider_set_value(slider, val, LV_ANIM_OFF);

    if(icon == NULL) {
        lv_obj_add_flag(slider, LV_OBJ_FLAG_FLEX_IN_NEW_TRACK);
    }

    return obj;
}

static lv_obj_t * create_switch(lv_obj_t * parent, const char * icon, const char * txt, bool chk)
{
    lv_obj_t * obj = create_text(parent, icon, txt, LV_MENU_ITEM_BUILDER VARIANT_1);

    lv_obj_t * sw = lv_switch_create(obj);
    lv_obj_add_state(sw, chk ? LV_STATE_CHECKED : LV_STATE_DEFAULT);

    return obj;
}

#endif

```

3.7.21 Casella Messaggi

Message box semplice

```

#include "../../lv_examples.h"
#if LV_USE_MSGBOX && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * btn = lv_event_get_target_obj(e);
    lv_obj_t * label = lv_obj_get_child(btn, 0);
    LV_UNUSED(label);
    LV_LOG_USER("Button %s clicked", lv_label_get_text(label));
}

void lv_example_msgbox_1(void)
{
    lv_obj_t * mbox1 = lv_msgbox_create(NULL);

    lv_msgbox_add_title(mbox1, "Hello");

    lv_msgbox_add_text(mbox1, "This is a message box with two buttons.");
    lv_msgbox_add_close_button(mbox1);

    lv_obj_t * btn;
    btn = lv_msgbox_add_footer_button(mbox1, "Apply");
    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_CLICKED, NULL);
    btn = lv_msgbox_add_footer_button(mbox1, "Cancel");
    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_CLICKED, NULL);
    return;
}

#endif

```

Message box con scrolling e stile

```

#include "../../lv_examples.h"
#if LV_USE_MSGBOX && LV_BUILD_EXAMPLES

static void minimize_button_event_cb(lv_event_t * e)
{
    lv_obj_t * mbox = (lv_obj_t *) lv_event_get_user_data(e);
    lv_obj_add_flag(mbox, LV_OBJ_FLAG_HIDDEN);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

void lv_example_msgbox_2(void)
{
    lv_obj_t * setting = lv_msgbox_create(lv_screen_active());
    lv_obj_set_style_clip_corner(setting, true, 0);

    /* setting fixed size */
    lv_obj_set_size(setting, 300, 200);

    /* setting's titlebar/header */
    lv_msgbox_add_title(setting, "Setting");
    lv_obj_t * minimize_button = lv_msgbox_add_header_button(setting, LV_SYMBOL_MINUS);
    lv_obj_add_event_cb(minimize_button, minimize_button_event_cb, LV_EVENT_CLICKED, setting);
    lv_msgbox_add_close_button(setting);

    /* setting's content */
    lv_obj_t * content = lv_msgbox_get_content(setting);
    lv_obj_set_flex_flow(content, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(content, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);
    lv_obj_set_style_pad_right(content, -1, LV_PART_SCROLLBAR);

    lv_obj_t * cont_brightness = lv_obj_create(content);
    lv_label_set_text(cont_brightness, "Brightness : ");
    lv_obj_t * slider_brightness = lv_slider_create(cont_brightness);
    lv_obj_set_width(slider_brightness, lv_pct(100));
    lv_slider_set_value(slider_brightness, 50, LV_ANIM_OFF);

    lv_obj_t * cont_speed = lv_obj_create(content);
    lv_obj_set_size(cont_speed, lv_pct(100), LV_SIZE_CONTENT);
    lv_obj_set_flex_flow(cont_speed, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(cont_speed, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER);

    lv_obj_t * lb_speed = lv_label_create(cont_speed);
    lv_label_set_text(lb_speed, "Speed : ");
    lv_obj_t * slider_speed = lv_slider_create(cont_speed);
    lv_obj_set_width(slider_speed, lv_pct(100));
    lv_slider_set_value(slider_speed, 80, LV_ANIM_OFF);

    /* footer */
    lv_obj_t * apply_button = lv_msgbox_add_footer_button(setting, "Apply");
    lv_obj_set_flex_grow(apply_button, 1);

    lv_obj_t * cancel_button = lv_msgbox_add_footer_button(setting, "Cancel");
    lv_obj_set_flex_grow(cancel_button, 1);

    lv_obj_t * footer = lv_msgbox_get_footer(setting);
    lv_obj_set_style_bg_color(footer, lv_palette_main(LV_PALETTE_INDIGO), 0);
    lv_obj_set_style_bg_opa(footer, LV_OPA_100, 0);
}
#endif

```

3.7.22 Roller

Roller semplice

```

#include "../../lv_examples.h"
#if LV_USE_ROLLER && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_roller_get_selected_str(obj, buf, sizeof(buf));
        LV_LOG_USER("Selected month: %s\n", buf);
    }
}

/**
 * An infinite roller with the name of the months
 */
void lv_example_roller_1(void)
{
    lv_obj_t * roller1 = lv_roller_create(lv_screen_active());
    lv_roller_set_options(roller1,
        "January\n"
        "February\n"
        "March\n"
        "April\n"
        "May\n"
        "June\n"
        "July\n"
        "August\n"
        "September\n"
        "October\n"
        "November\n"
        "December",
        LV_ROLLER_MODE_INFINITE);

    lv_roller_set_visible_row_count(roller1, 4);
    lv_obj_center(roller1);
    lv_obj_add_event_cb(roller1, event_handler, LV_EVENT_ALL, NULL);
}
#endif

```

Roller con stili

```
#include "../../lv_examples.h"
#if LV_USE_ROLLER && LV_FONT_MONTserrat_22 && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_roller_get_selected_str(obj, buf, sizeof(buf));
        LV_LOG_USER("Selected value: %s", buf);
    }
}

/**
 * Roller with various alignments and larger text in the selected area
 */
void lv_example_roller_2(void)
{
    /*A style to make the selected option larger*/
    static lv_style_t style_sel;
    lv_style_init(&style_sel);
    lv_style_set_text_font(&style_sel, &lv_font_montserrat_22);
    lv_style_set_bg_color(&style_sel, lv_color_hex3(0xf88));
    lv_style_set_border_width(&style_sel, 2);
    lv_style_set_border_color(&style_sel, lv_color_hex3(0xf00));

    const char * opts = "1\n2\n3\n4\n5\n6\n7\n8\n9\n10";
    lv_obj_t * roller;

    /*A roller on the left with left aligned text, and custom width*/
    roller = lv_roller_create(lv_screen_active());
    lv_roller_set_options(roller, opts, LV_ROLLER_MODE_NORMAL);
    lv_roller_set_visible_row_count(roller, 2);
    lv_obj_set_width(roller, 100);
    lv_obj_add_style(roller, &style_sel, LV_PART_SELECTED);
    lv_obj_set_style_text_align(roller, LV_TEXT_ALIGN_LEFT, 0);
    lv_obj_set_style_bg_color(roller, lv_color_hex3(0x0f0), 0);
    lv_obj_set_style_bg_grad_color(roller, lv_color_hex3(0xfafa), 0);
    lv_obj_set_style_bg_grad_dir(roller, LV_GRAD_DIR_VER, 0);
    lv_obj_align(roller, LV_ALIGN_LEFT_MID, 10, 0);
    lv_obj_add_event_cb(roller, event_handler, LV_EVENT_ALL, NULL);
    lv_roller_set_selected(roller, 2, LV_ANIM_OFF);

    /*A roller on the middle with center aligned text, and auto (default) width*/
    roller = lv_roller_create(lv_screen_active());
    lv_roller_set_options(roller, opts, LV_ROLLER_MODE_NORMAL);
    lv_roller_set_visible_row_count(roller, 3);
    lv_obj_add_style(roller, &style_sel, LV_PART_SELECTED);
    lv_obj_align(roller, LV_ALIGN_CENTER, 0, 0);
    lv_obj_add_event_cb(roller, event_handler, LV_EVENT_ALL, NULL);
    lv_roller_set_selected(roller, 5, LV_ANIM_OFF);

    /*A roller on the right with right aligned text, and custom width*/
    roller = lv_roller_create(lv_screen_active());
    lv_roller_set_options(roller, opts, LV_ROLLER_MODE_NORMAL);
    lv_roller_set_visible_row_count(roller, 4);
    lv_obj_set_width(roller, 80);
    lv_obj_add_style(roller, &style_sel, LV_PART_SELECTED);
    lv_obj_set_style_text_align(roller, LV_TEXT_ALIGN_RIGHT, 0);
    lv_obj_align(roller, LV_ALIGN_RIGHT_MID, -10, 0);
    lv_obj_add_event_cb(roller, event_handler, LV_EVENT_ALL, NULL);
    lv_roller_set_selected(roller, 8, LV_ANIM_OFF);
}

#endif
```

Aggiungere una maschera di dissolvenza al roller

```
#include "../../lv_examples.h"
#if LV_USE_ROLLER && LV_DRAW_SW_COMPLEX && LV_BUILD_EXAMPLES && 1

static void generate_mask(lv_draw_buf_t * mask)
{
    /*Create a "8 bit alpha" canvas and clear it*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, mask);
    lv_canvas_fill_bg(canvas, lv_color_white(), LV_OPA_TRANSP);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    /*Draw a label to the canvas. The result "image" will be used as mask*/
    lv_draw_rect_dsc_t rect_dsc;
    lv_draw_rect_dsc_init(&rect_dsc);
    rect_dsc.bg_grad.dir = LV_GRAD_DIR_VER;
    rect_dsc.bg_grad.stops[0].color = lv_color_black();
    rect_dsc.bg_grad.stops[1].color = lv_color_white();
    rect_dsc.bg_grad.stops[0].opa = LV_OPA_COVER;
    rect_dsc.bg_grad.stops[1].opa = LV_OPA_COVER;
    lv_area_t a = {0, 0, mask->header.w - 1, mask->header.h / 2 - 10};
    lv_draw_rect(&layer, &rect_dsc, &a);

    a.y1 = mask->header.h / 2 + 10;
    a.y2 = mask->header.h - 1;
    rect_dsc.bg_grad.stops[0].color = lv_color_white();
    rect_dsc.bg_grad.stops[1].color = lv_color_black();
    lv_draw_rect(&layer, &rect_dsc, &a);

    lv_canvas_finish_layer(canvas, &layer);

    /*Comment it to make the mask visible*/
    lv_obj_delete(canvas);
}

/**
```

(continues on next page)

(continua dalla pagina precedente)

```

/* Add a fade mask to roller.
*/
void lv_example_roller_3(void)
{
    lv_obj_set_style_bg_color(lv_screen_active(), lv_palette_main(LV_PALETTE_BLUE_GREY), 0);

    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_bg_color(&style, lv_color_black());
    lv_style_set_text_color(&style, lv_color_white());
    lv_style_set_border_width(&style, 0);
    lv_style_set_radius(&style, 0);

    lv_obj_t * roller1 = lv_roller_create(lv_screen_active());
    lv_obj_add_style(roller1, &style, 0);
    lv_obj_set_style_bg_opa(roller1, LV_OPA_50, LV_PART_SELECTED);

    lv_roller_set_options(roller1,
        "January\n"
        "February\n"
        "March\n"
        "April\n"
        "May\n"
        "June\n"
        "July\n"
        "August\n"
        "September\n"
        "October\n"
        "November\n"
        "December",
        LV_ROLLER_MODE_NORMAL);

    lv_obj_center(roller1);
    lv_roller_set_visible_row_count(roller1, 4);

    /* Create the mask to make the top and bottom part of roller faded.
     * The width and height are empirical values for simplicity/
    LV_DRAW_BUF_DEFINE_STATIC(mask, 130, 150, LV_COLOR_FORMAT_L8);
    LV_DRAW_BUF_INIT_STATIC(mask);

    generate_mask(&mask);
    lv_obj_set_style_bitmap_mask_src(roller1, &mask, 0);
}
#endif

```

3.7.23 Scale

Una semplice scala orizzontale

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/***
 * A simple horizontal scale
 */
void lv_example_scale_1(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, lv_pct(80), 100);
    lv_scale_set_mode(scale, LV_SCALE_MODE_HORIZONTAL_BOTTOM);
    lv_obj_center(scale);

    lv_scale_set_label_show(scale, true);

    lv_scale_set_total_tick_count(scale, 31);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 10, 40);
}

#endif

```

Una scala verticale con sezioni e stile personalizzato

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/***
 * An vertical scale with section and custom styling
 */
void lv_example_scale_2(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, 60, 200);
    lv_scale_set_label_show(scale, true);
    lv_scale_set_mode(scale, LV_SCALE_MODE_VERTICAL_RIGHT);
    lv_obj_center(scale);

    lv_scale_set_total_tick_count(scale, 21);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_scale_set_range(scale, 0, 100);

    static const char * custom_labels[] = {"0 °C", "25 °C", "50 °C", "75 °C", "100 °C", NULL};

    static lv_style_t indicator_style;
    lv_style_init(&indicator_style);

```

(continues on next page)

(continua dalla pagina precedente)

```

/* Label style properties */
lv_style_set_text_font(&indicator_style, LV_FONT_DEFAULT);
lv_style_set_text_color(&indicator_style, lv_palette_darken(LV_PALETTE_BLUE, 3));

/* Major tick properties */
lv_style_set_line_color(&indicator_style, lv_palette_darken(LV_PALETTE_BLUE, 3));
lv_style_set_width(&indicator_style, 10U); /*Tick length*/
lv_style_set_line_width(&indicator_style, 2U); /*Tick width*/
lv_obj_add_style(scale, &indicator_style, LV_PART_INDICATOR);

static lv_style_t minor_ticks_style;
lv_style_init(&minor_ticks_style);
lv_style_set_line_color(&minor_ticks_style, lv_palette_lighten(LV_PALETTE_BLUE, 2));
lv_style_set_width(&minor_ticks_style, 5U); /*Tick length*/
lv_style_set_line_width(&minor_ticks_style, 2U); /*Tick width*/
lv_obj_add_style(scale, &minor_ticks_style, LV_PART_ITEMS);

static lv_style_t main_line_style;
lv_style_init(&main_line_style);
/* Main line properties */
lv_style_set_line_color(&main_line_style, lv_palette_darken(LV_PALETTE_BLUE, 3));
lv_style_set_line_width(&main_line_style, 2U); // Tick width
lv_obj_add_style(scale, &main_line_style, LV_PART_MAIN);

/* Add a section */
static lv_style_t section_minor_tick_style;
static lv_style_t section_label_style;
static lv_style_t section_main_line_style;

lv_style_init(&section_label_style);
lv_style_init(&section_minor_tick_style);
lv_style_init(&section_main_line_style);

/* Label style properties */
lv_style_set_text_font(&section_label_style, LV_FONT_DEFAULT);
lv_style_set_text_color(&section_label_style, lv_palette_darken(LV_PALETTE_RED, 3));

lv_style_set_line_color(&section_label_style, lv_palette_darken(LV_PALETTE_RED, 3));
lv_style_set_line_width(&section_label_style, 5U); /*Tick width*/

lv_style_set_line_color(&section_minor_tick_style, lv_palette_lighten(LV_PALETTE_RED, 2));
lv_style_set_line_width(&section_minor_tick_style, 4U); /*Tick width*/

/* Main line properties */
lv_style_set_line_color(&section_main_line_style, lv_palette_darken(LV_PALETTE_RED, 3));
lv_style_set_line_width(&section_main_line_style, 4U); /*Tick width*/

/* Configure section styles */
lv_scale_section_t * section = lv_scale_add_section(scale);
lv_scale_set_section_range(scale, section, 75, 100);
lv_scale_set_section_style_indicator(scale, section, &section_label_style);
lv_scale_set_section_style_items(scale, section, &section_minor_tick_style);
lv_scale_set_section_style_main(scale, section, &section_main_line_style);

lv_obj_set_style_bg_color(scale, lv_palette_main(LV_PALETTE_BLUE_GREY), 0);
lv_obj_set_style_bg_opa(scale, LV_OPA_50, 0);
lv_obj_set_style_pad_left(scale, 8, 0);
lv_obj_set_style_radius(scale, 8, 0);
lv_obj_set_style_pad_ver(scale, 20, 0);
}

#endif

```

Una semplice scala circolare

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

LV_IMAGE_DECLARE(img_hand);

static lv_obj_t * needle_line;
static lv_obj_t * needle_img;

static void set_needle_line_value(void * obj, int32_t v)
{
    lv_scale_set_line_needle_value((lv_obj_t *)obj, needle_line, 60, v);
}

static void set_needle_img_value(void * obj, int32_t v)
{
    lv_scale_set_image_needle_value((lv_obj_t *)obj, needle_img, v);
}

/**
 * A simple round scale
 */
void lv_example_scale_3(void)
{
    lv_obj_t * scale_line = lv_scale_create(lv_screen_active());

    lv_obj_set_size(scale_line, 150, 150);
    lv_scale_set_mode(scale_line, LV_SCALE_MODE_ROUND_INNER);
    lv_obj_set_style_bg_opa(scale_line, LV_OPA_COVER, 0);
    lv_obj_set_style_bg_color(scale_line, lv_palette_lighten(LV_PALETTE_RED, 5), 0);
    lv_obj_set_style_radius(scale_line, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_clip_corner(scale_line, true, 0);
    lv_obj_align(scale_line, LV_ALIGN_LEFT_MID, LV_PCT(2), 0);

    lv_scale_set_label_show(scale_line, true);

    lv_scale_set_total_tick_count(scale_line, 31);
    lv_scale_set_major_tick_every(scale_line, 5);

    lv_obj_set_style_length(scale_line, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale_line, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale_line, 10, 40);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_scale_set_angle_range(scale_line, 270);
lv_scale_set_rotation(scale_line, 135);

needle_line = lv_line_create(scale_line);

lv_obj_set_style_line_width(needle_line, 6, LV_PART_MAIN);
lv_obj_set_style_line_rounded(needle_line, true, LV_PART_MAIN);

lv_anim_t anim_scale_line;
lv_anim_init(&anim_scale_line);
lv_anim_set_var(&anim_scale_line, scale_line);
lv_anim_set_exec_cb(&anim_scale_line, set_needle_line_value);
lv_anim_set_duration(&anim_scale_line, 1000);
lv_anim_set_repeat_count(&anim_scale_line, LV_ANIM_REPEAT_INFINITE);
lv_anim_set_reverse_duration(&anim_scale_line, 1000);
lv_anim_set_values(&anim_scale_line, 10, 40);
lv_anim_start(&anim_scale_line);

lv_obj_t * scale_img = lv_scale_create(lv_screen_active());

lv_obj_set_size(scale_img, 150, 150);
lv_scale_set_mode(scale_img, LV_SCALE_MODE_ROUND_INNER);
lv_obj_set_style_bg_opa(scale_img, LV_OPA_COVER, 0);
lv_obj_set_style_bg_color(scale_img, lv_palette_lighten(LV_PALETTE_RED, 5), 0);
lv_obj_set_style_radius(scale_img, LV_RADIUS_CIRCLE, 0);
lv_obj_set_style_clip_corner(scale_img, true, 0);
lv_obj_align(scale_img, LV_ALIGN_RIGHT_MID, LV_PCT(-2), 0);

lv_scale_set_label_show(scale_img, true);

lv_scale_set_total_tick_count(scale_img, 31);
lv_scale_set_major_tick_every(scale_img, 5);

lv_obj_set_style_length(scale_img, 5, LV_PART_ITEMS);
lv_obj_set_style_length(scale_img, 10, LV_PART_INDICATOR);
lv_scale_set_range(scale_img, 10, 40);

lv_scale_set_angle_range(scale_img, 270);
lv_scale_set_rotation(scale_img, 135);

/* image must point to the right. E.g. -0----->*
needle_img = lv_image_create(scale_img);
lv_image_set_src(needle_img, &img_hand);
lv_obj_align(needle_img, LV_ALIGN_CENTER, 47, -2);
lv_image_set_pivot(needle_img, 3, 4);

lv_anim_t anim_scale_img;
lv_anim_init(&anim_scale_img);
lv_anim_set_var(&anim_scale_img, scale_img);
lv_anim_set_exec_cb(&anim_scale_img, set_needle_img_value);
lv_anim_set_duration(&anim_scale_img, 1000);
lv_anim_set_repeat_count(&anim_scale_img, LV_ANIM_REPEAT_INFINITE);
lv_anim_set_reverse_duration(&anim_scale_img, 1000);
lv_anim_set_values(&anim_scale_img, 10, 40);
lv_anim_start(&anim_scale_img);
}

#endif

```

Una scala circolare con sezione e stile personalizzato

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/***
 * A round scale with section and custom styling
 */
void lv_example_scale_4(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, 150, 150);
    lv_scale_set_label_show(scale, true);
    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_OUTER);
    lv_obj_center(scale);

    lv_scale_set_total_tick_count(scale, 21);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 0, 100);

    static const char * custom_labels[] = {"0 °C", "25 °C", "50 °C", "75 °C", "100 °C", NULL};
    lv_scale_set_text_src(scale, custom_labels);

    static lv_style_t indicator_style;
    lv_style_init(&indicator_style);

    /* Label style properties */
    lv_style_set_text_font(&indicator_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&indicator_style, lv_palette_darker(LV_PALETTE_BLUE, 3));

    /* Major tick properties */
    lv_style_set_line_color(&indicator_style, lv_palette_darker(LV_PALETTE_BLUE, 3));
    lv_style_set_width(&indicator_style, 10U); /*Tick length*/
    lv_style_set_line_width(&indicator_style, 2U); /*Tick width*/
    lv_obj_add_style(scale, &indicator_style, LV_PART_INDICATOR);

    static lv_style_t minor_ticks_style;
    lv_style_init(&minor_ticks_style);
    lv_style_set_line_color(&minor_ticks_style, lv_palette_lighten(LV_PALETTE_BLUE, 2));
    lv_style_set_width(&minor_ticks_style, 5U); /*Tick length*/
    lv_style_set_line_width(&minor_ticks_style, 1U); /*Tick width*/
    lv_obj_add_style(scale, &minor_ticks_style, LV_PART_ITEMS);

    static lv_style_t main_line_style;
    lv_style_init(&main_line_style);

```

(continues on next page)

(continua dalla pagina precedente)

```

/* Main line properties */
lv_style_set_arc_color(&main_line_style, lv_palette_darken(LV_PALETTE_BLUE, 3));
lv_style_set_arc_width(&main_line_style, 20); /*Tick width*/
lv_obj_add_style(scale, &main_line_style, LV_PART_MAIN);

/* Add a section */
static lv_style_t section_minor_tick_style;
static lv_style_t section_label_style;
static lv_style_t section_main_line_style;

lv_style_init(&section_label_style);
lv_style_init(&section_minor_tick_style);
lv_style_init(&section_main_line_style);

/* Label style properties */
lv_style_set_text_font(&section_label_style, LV_FONT_DEFAULT);
lv_style_set_text_color(&section_label_style, lv_palette_darken(LV_PALETTE_RED, 3));

lv_style_set_line_color(&section_label_style, lv_palette_darken(LV_PALETTE_RED, 3));
lv_style_set_line_width(&section_label_style, 5U); /*Tick width*/

lv_style_set_line_color(&section_minor_tick_style, lv_palette_lighten(LV_PALETTE_RED, 2));
lv_style_set_line_width(&section_minor_tick_style, 4U); /*Tick width*/

/* Main line properties */
lv_style_set_arc_color(&section_main_line_style, lv_palette_darken(LV_PALETTE_RED, 3));
lv_style_set_arc_width(&section_main_line_style, 4U); /*Tick width*/

/* Configure section styles */
lv_scale_section_t * section = lv_scale_add_section(scale);
lv_scale_set_section_range(scale, section, 75, 100);
lv_scale_set_section_style_indicator(scale, section, &section_label_style);
lv_scale_set_section_style_items(scale, section, &section_minor_tick_style);
lv_scale_set_section_style_main(scale, section, &section_main_line_style);

}

#endif

```

Una scala con sezione e stile personalizzato

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/***
 * An scale with section and custom styling
 */
void lv_example_scale_5(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, lv_display_get_horizontal_resolution(NULL) / 2, lv_display_get_vertical_resolution(NULL) / 2);
    lv_scale_set_label_show(scale, true);

    lv_scale_set_total_tick_count(scale, 10);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 25, 35);

    static const char * custom_labels[3] = {"One", "Two", NULL};
    lv_scale_set_text_src(scale, custom_labels);

    static lv_style_t indicator_style;
    lv_style_init(&indicator_style);
    /* Label style properties */
    lv_style_set_text_font(&indicator_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&indicator_style, lv_color_hex(0xffff00));
    /* Major tick properties */
    lv_style_set_line_color(&indicator_style, lv_color_hex(0x00ff00));
    lv_style_set_width(&indicator_style, 10U); // Tick length
    lv_style_set_line_width(&indicator_style, 2U); // Tick width
    lv_obj_add_style(scale, &indicator_style, LV_PART_INDICATOR);

    static lv_style_t minor_ticks_style;
    lv_style_init(&minor_ticks_style);
    lv_style_set_line_color(&minor_ticks_style, lv_color_hex(0xff0000));
    lv_style_set_width(&minor_ticks_style, 5U); // Tick length
    lv_style_set_line_width(&minor_ticks_style, 2U); // Tick width
    lv_obj_add_style(scale, &minor_ticks_style, LV_PART_ITEMS);

    static lv_style_t main_line_style;
    lv_style_init(&main_line_style);
    /* Main line properties */
    lv_style_set_line_color(&main_line_style, lv_color_hex(0x0000ff));
    lv_style_set_line_width(&main_line_style, 20); // Tick width
    lv_obj_add_style(scale, &main_line_style, LV_PART_MAIN);

    lv_obj_center(scale);

    /* Add a section */
    static lv_style_t section_minor_tick_style;
    static lv_style_t section_label_style;

    lv_style_init(&section_label_style);
    lv_style_init(&section_minor_tick_style);

    /* Label style properties */
    lv_style_set_text_font(&section_label_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&section_label_style, lv_color_hex(0xffff00));
    lv_style_set_text_letter_space(&section_label_style, 10);
    lv_style_set_text_opa(&section_label_style, LV_OPA_50);

    lv_style_set_line_color(&section_label_style, lv_color_hex(0x0000ff));
    // lv_style_set_width(&section_label_style, 20U); // Tick length
    lv_style_set_line_width(&section_label_style, 5U); // Tick width

    lv_style_set_line_color(&section_minor_tick_style, lv_color_hex(0x0000ff));

```

(continues on next page)

(continua dalla pagina precedente)

```

// lv_style_set_width(&section_label_style, 20U); // Tick length
lv_style_set_line_width(&section_minor_tick_style, 4U); // Tick width

/* Configure section styles */
lv_scale_section_t * section = lv_scale_add_section(scale);
lv_scale_set_section_range(scale, section, 25, 30);
lv_scale_set_section_style_indicator(&scale, section, &section_label_style);
lv_scale_set_section_style_items(&scale, section, &section_minor_tick_style);
}

#endif

```

Una scala circolare con più lancette, simile a un orologio

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

#if LV_USE_FLOAT
#define my_PRIPrecise "f"
#else
#define my_PRIPrecise LV_PRId32
#endif

static lv_obj_t * scale;
static lv_obj_t * minute_hand;
static lv_obj_t * hour_hand;
static lv_point_precise_t minute_hand_points[2];
static int32_t hour;
static int32_t minute;

static void timer_cb(lv_timer_t * timer)
{
    LV_UNUSED(timer);

    minute++;
    if(minute > 59) {
        minute = 0;
        hour++;
        if(hour > 11) {
            hour = 0;
        }
    }

    /**
     * the scale will store the needle line points in the existing
     * point array if one was set with `lv_line_set_points Mutable`.
     * Otherwise, it will allocate the needle line points.
     */

    /* the scale will store the minute hand line points in 'minute_hand_points' */
    lv_scale_set_line_needle_value(scale, minute_hand, 60, minute);
    /* log the points that were stored in the array */
    LV_LOG_USER(
        "minute hand points - "
        "0: (%" my_PRIPrecise ", %" my_PRIPrecise "), "
        "1: (%" my_PRIPrecise ", %" my_PRIPrecise ")",
        minute_hand_points[0].x, minute_hand_points[0].y,
        minute_hand_points[1].x, minute_hand_points[1].y
    );
    /* the scale will allocate the hour hand line points */
    lv_scale_set_line_needle_value(scale, hour_hand, 40, hour * 5 + (minute / 12));
}

/**
 * A round scale with multiple needles, resembling a clock
 */
void lv_example_scale_6(void)
{
    scale = lv_scale_create(lv_screen_active());

    lv_obj_set_size(scale, 150, 150);
    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_INNER);
    lv_obj_set_style_bg_opa(scale, LV_OPA_60, 0);
    lv_obj_set_style_bg_color(scale, lv_color_black(), 0);
    lv_obj_set_style_radius(scale, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_clip_corner(scale, true, 0);
    lv_obj_center(scale);

    lv_scale_set_label_show(scale, true);

    lv_scale_set_total_tick_count(scale, 61);
    lv_scale_set_major_tick_every(scale, 5);

    static const char * hour_ticks[] = {"12", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", NULL};
    lv_scale_set_text_src(scale, hour_ticks);

    static lv_style_t indicator_style;
    lv_style_init(&indicator_style);

    /* Label style properties */
    lv_style_set_text_font(&indicator_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&indicator_style, lv_palette_main(LV_PALETTE_YELLOW));

    /* Major tick properties */
    lv_style_set_line_color(&indicator_style, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_length(&indicator_style, 8); /* tick length */
    lv_style_set_line_width(&indicator_style, 2); /* tick width */
    lv_obj_add_style(scale, &indicator_style, LV_PART_INDICATOR);

    /* Minor tick properties */
    static lv_style_t minor_ticks_style;
    lv_style_init(&minor_ticks_style);
    lv_style_set_line_color(&minor_ticks_style, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_length(&minor_ticks_style, 6); /* tick length */
    lv_style_set_line_width(&minor_ticks_style, 2); /* tick width */
    lv_obj_add_style(scale, &minor_ticks_style, LV_PART_ITEMS);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

/* Main line properties */
static lv_style_t main_line_style;
lv_style_init(&main_line_style);
lv_style_set_arc_color(&main_line_style, lv_color_black());
lv_style_set_arc_width(&main_line_style, 5);
lv_obj_add_style(scale, &main_line_style, LV_PART_MAIN);

lv_scale_set_range(scale, 0, 60);

lv_scale_set_angle_range(scale, 360);
lv_scale_set_rotation(scale, 270);

minute_hand = lv_line_create(scale);
lv_line_set_points_mutable(minute_hand, minute_hand_points, 2);

lv_obj_set_style_line_width(minute_hand, 3, 0);
lv_obj_set_style_line_rounded(minute_hand, true, 0);
lv_obj_set_style_line_color(minute_hand, lv_color_white(), 0);

hour_hand = lv_line_create(scale);

lv_obj_set_style_line_width(hour_hand, 5, 0);
lv_obj_set_style_line_rounded(hour_hand, true, 0);
lv_obj_set_style_line_color(hour_hand, lv_palette_main(LV_PALETTE_RED), 0);

hour = 11;
minute = 5;
lv_timer_t * timer = lv_timer_create(timer_cb, 250, NULL);
lv_timer_ready(timer);
}

#endif

```

Personalizzazione del colore dell'etichetta principale della scala con l'evento LV_EVENT_DRAW_TASK_ADDED

```

#include "../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

#include "../../../../lvgl_private.h" /*To expose the fields of lv_draw_task_t*/

static void draw_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
    if(base_dsc->part == LV_PART_INDICATOR) {
        if(label_draw_dsc) {
            const lv_color_t color_idx[7] = {
                lv_palette_main(LV_PALETTE_RED),
                lv_palette_main(LV_PALETTE_ORANGE),
                lv_palette_main(LV_PALETTE_YELLOW),
                lv_palette_main(LV_PALETTE_GREEN),
                lv_palette_main(LV_PALETTE_CYAN),
                lv_palette_main(LV_PALETTE_BLUE),
                lv_palette_main(LV_PALETTE_PURPLE),
            };
            uint32_t major_tick = lv_scale_get_major_tick_every(obj);
            label_draw_dsc->color = color_idx[base_dsc->id1 / major_tick];
        }
        /*Free the previously allocated text if needed*/
        if(label_draw_dsc->text_local) lv_free((void *)label_draw_dsc->text);

        /*Malloc the text and set text_local as 1 to make LVGL automatically free the text.
         * (Local texts are malloc'd internally by LVGL. Mimic this behavior here too)*/
        char tmp_buffer[20] = {0}; /* Big enough buffer */
        lv_snprintf(tmp_buffer, sizeof(tmp_buffer), "%.\lf", (double)base_dsc->id2);
        label_draw_dsc->text = lv_strdup(tmp_buffer);
        label_draw_dsc->text_local = 1;

        lv_point_t size;
        lv_text_get_size(&size, label_draw_dsc->text, label_draw_dsc->font, 0, 0, 1000, LV_TEXT_FLAG_NONE);
        int32_t new_w = size.x;
        int32_t old_w = lv_area_get_width(&draw_task->area);

        /* Distribute the new size equally on both sides */
        draw_task->area.x1 -= (new_w - old_w) / 2;
        draw_task->area.x2 += ((new_w - old_w) + 1) / 2; /* +1 for rounding */
    }
}

/**
 * Customizing scale major tick label color with `LV_EVENT_DRAW_TASK_ADDED` event
 */
void lv_example_scale_7(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, lv_pct(80), 100);
    lv_scale_set_mode(scale, LV_SCALE_MODE_HORIZONTAL_BOTTOM);
    lv_obj_center(scale);

    lv_scale_set_label_show(scale, true);

    lv_scale_set_total_tick_count(scale, 31);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 10, 40);

    lv_obj_add_event_cb(scale, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(scale, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
}
#endif

```

Una scala circolare con etichette ruotate e traslate

```
#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/***
 * A simple round scale with label/tick translation
 */
void lv_example_scale_8(void)
{
    lv_obj_t * scale_line = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale_line, 150, 150);
    lv_scale_set_mode(scale_line, LV_SCALE_MODE_ROUND_INNER);
    lv_obj_set_style_bg_opa(scale_line, LV_OPA_COVER, 0);
    lv_obj_set_style_bg_color(scale_line, Lv_palette_lighten(LV_PALETTE_RED, 5), 0);
    lv_obj_set_style_radius(scale_line, LV_RADIUS_CIRCLE, 0);
    lv_obj_align(scale_line, LV_ALIGN_LEFT_MID, LV_PCT(2), 0);

    /*Set the texts' and major ticks' style (make the texts rotated)*/
    lv_obj_set_style_transform_rotation(scale_line, LV_SCALE_LABEL_ROTATE_MATCH_TICKS | LV_SCALE_LABEL_ROTATE_KEEP_UPRIGHT,
                                      LV_PART_INDICATOR);
    lv_obj_set_style_translate_x(scale_line, 10, LV_PART_INDICATOR);
    lv_obj_set_style_length(scale_line, 15, LV_PART_INDICATOR);
    lv_obj_set_style_radial_offset(scale_line, 10, LV_PART_INDICATOR);

    /*Set the style of the minor ticks*/
    lv_obj_set_style_length(scale_line, 10, LV_PART_ITEMS);
    lv_obj_set_style_radial_offset(scale_line, 5, LV_PART_ITEMS);
    lv_obj_set_style_line_opa(scale_line, LV_OPA_50, LV_PART_ITEMS);

    lv_scale_set_label_show(scale_line, true);
    lv_scale_set_total_tick_count(scale_line, 31);
    lv_scale_set_major_tick_every(scale_line, 5);

    lv_scale_set_range(scale_line, 10, 40);
    lv_scale_set_angle_range(scale_line, 270);
    lv_scale_set_rotation(scale_line, 135);

    lv_obj_t * needle_line = lv_line_create(scale_line);

    lv_obj_set_style_line_width(needle_line, 3, LV_PART_MAIN);
    lv_obj_set_style_line_rounded(needle_line, true, LV_PART_MAIN);
    lv_scale_set_line_needle_value(scale_line, needle_line, 60, 33);
}

#endif
```

Una scala orizzontale con etichette ruotate e traslate

```
#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/***
 * A simple horizontal scale with transforms
 */
void lv_example_scale_9(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, 200, 100);
    lv_scale_set_mode(scale, LV_SCALE_MODE_HORIZONTAL_BOTTOM);
    lv_obj_center(scale);

    lv_scale_set_label_show(scale, true);
    lv_obj_set_style_transform_rotation(scale, 450, LV_PART_INDICATOR);
    lv_obj_set_style_length(scale, 30, LV_PART_INDICATOR);
    lv_obj_set_style_translate_x(scale, 5, LV_PART_INDICATOR);

    lv_scale_set_total_tick_count(scale, 31);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 10, 40);
}

#endif
```

Uno stile di scala circolare che simula un cardiofrequenzimetro

```
#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES && LV_FONT_MONTserrat_40 && LV_FONT_MONTserrat_18

static int32_t hr_value = 98;
static int8_t hr_step = 1;
static lv_obj_t * needle_line = NULL;
static lv_obj_t * hr_value_label = NULL;
static lv_obj_t * bpm_label = NULL;
static lv_obj_t * scale = NULL;

typedef struct {
    lv_style_t items;
    lv_style_t indicator;
    lv_style_t main;
} section_styles_t;

static section_styles_t zone1_styles;
static section_styles_t zone2_styles;
static section_styles_t zone3_styles;
static section_styles_t zone4_styles;
static section_styles_t zone5_styles;

static lv_color_t get_hr_zone_color(int32_t hr)
```

(continues on next page)

(continua dalla pagina precedente)

```

{
    if(hr < 117) return lv_palette_main(LV_PALETTE_GREY); /* Zone 1 */
    else if(hr < 135) return lv_palette_main(LV_PALETTE_BLUE); /* Zone 2 */
    else if(hr < 158) return lv_palette_main(LV_PALETTE_GREEN); /* Zone 3 */
    else if(hr < 176) return lv_palette_main(LV_PALETTE_ORANGE); /* Zone 4 */
    else return lv_palette_main(LV_PALETTE_RED); /* Zone 5 */
}

static void hr_anim_timer_cb(lv_timer_t * timer)
{
    LV_UNUSED(timer);

    hr_value += hr_step;

    if(hr_value >= 195) {
        hr_value = 195;
        hr_step = -1;
    }
    else if(hr_value <= 98) {
        hr_value = 98;
        hr_step = 1;
    }

    /* Update needle */
    lv_scale_set_line_needle_value(scale, needle_line, -8, hr_value);

    /* Update HR text */
    lv_label_set_text_fmt(hr_value_label, "%d", hr_value);

    /* Update text color based on zone */
    lv_color_t zone_color = get_hr_zone_color(hr_value);
    lv_obj_set_style_text_color(hr_value_label, zone_color, 0);
    lv_obj_set_style_text_color(bpm_label, zone_color, 0);
}

static void init_section_styles(section_styles_t * styles, lv_color_t color)
{
    lv_style_init(&styles->items);
    lv_style_set_line_color(&styles->items, color);
    lv_style_set_line_width(&styles->items, 0);

    lv_style_init(&styles->indicator);
    lv_style_set_line_color(&styles->indicator, color);
    lv_style_set_line_width(&styles->indicator, 0);

    lv_style_init(&styles->main);
    lv_style_set_arc_color(&styles->main, color);
    lv_style_set_arc_width(&styles->main, 20);
}

static void add_section(lv_obj_t * target_scale,
                      int32_t from,
                      int32_t to,
                      const section_styles_t * styles)
{
    lv_scale_section_t * sec = lv_scale_add_section(target_scale);
    lv_scale_set_section_range(target_scale, sec, from, to);
    lv_scale_set_section_style_items(target_scale, sec, &styles->items);
    lv_scale_set_section_style_indicator(target_scale, sec, &styles->indicator);
    lv_scale_set_section_style_main(target_scale, sec, &styles->main);
}

void lv_example_scale_10(void)
{
    scale = lv_scale_create(lv_screen_active());
    lv_obj_center(scale);
    lv_obj_set_size(scale, 200, 200);

    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_INNER);
    lv_scale_set_range(scale, 98, 195);
    lv_scale_set_total_tick_count(scale, 15);
    lv_scale_set_major_tick_every(scale, 3);
    lv_scale_set_angle_range(scale, 280);
    lv_scale_set_rotation(scale, 130);
    lv_scale_set_label_show(scale, false);

    lv_obj_set_style_length(scale, 6, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_obj_set_style_arc_width(scale, 0, LV_PART_MAIN);

    /* Zone 1: (Grey) */
    init_section_styles(&zone1_styles, lv_palette_main(LV_PALETTE_GREY));
    add_section(scale, 98, 117, &zone1_styles);

    /* Zone 2: (Blue) */
    init_section_styles(&zone2_styles, lv_palette_main(LV_PALETTE_BLUE));
    add_section(scale, 117, 135, &zone2_styles);

    /* Zone 3: (Green) */
    init_section_styles(&zone3_styles, lv_palette_main(LV_PALETTE_GREEN));
    add_section(scale, 135, 158, &zone3_styles);

    /* Zone 4: (Orange) */
    init_section_styles(&zone4_styles, lv_palette_main(LV_PALETTE_ORANGE));
    add_section(scale, 158, 176, &zone4_styles);

    /* Zone 5: (Red) */
    init_section_styles(&zone5_styles, lv_palette_main(LV_PALETTE_RED));
    add_section(scale, 176, 195, &zone5_styles);

    needle_line = lv_line_create(scale);

    /* Optional styling */
    lv_obj_set_style_line_color(needle_line, lv_color_black(), LV_PART_MAIN);
    lv_obj_set_style_line_width(needle_line, 12, LV_PART_MAIN);
    lv_obj_set_style_length(needle_line, 20, LV_PART_MAIN);
    lv_obj_set_style_line_rounded(needle_line, false, LV_PART_MAIN);
    lv_obj_set_style_pad_right(needle_line, 50, LV_PART_MAIN);

    int32_t current_hr = 145;
    lv_scale_set_line_needle_value(scale, needle_line, 0, current_hr);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * circle = lv_obj_create(lv_screen_active());
lv_obj_set_size(circle, 130, 130);
lv_obj_center(circle);

lv_obj_set_style_radius(circle, LV_RADIUS_CIRCLE, 0);

lv_obj_set_style_bg_color(circle, lv_obj_get_style_bg_color(lv_screen_active(), LV_PART_MAIN), 0);
lv_obj_set_style_bg_opa(circle, LV_OPA_COVER, 0);
lv_obj_set_style_border_width(circle, 0, LV_PART_MAIN);

lv_obj_t * hr_container = lv_obj_create(circle);
lv_obj_center(hr_container);
lv_obj_set_size(hr_container, lv_pct(100), LV_SIZE_CONTENT);
lv_obj_set_style_bg_opa(hr_container, LV_OPA_TRANS, 0);
lv_obj_set_style_border_width(hr_container, 0, 0);
lv_obj_set_layout(hr_container, LV_LAYOUT_FLEX);
lv_obj_set_flex_flow(hr_container, LV_FLEX_FLOW_COLUMN);
lv_obj_set_style_pad_all(hr_container, 0, LV_PART_MAIN);
lv_obj_set_style_pad_row(hr_container, 0, 0);
lv_obj_set_flex_align(hr_container, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

hr_value_label = lv_label_create(hr_container);
lv_label_set_text_fmt(hr_value_label, "%d", current_hr);
lv_obj_set_style_text_font(hr_value_label, &lv_font_montserrat_40, 0);
lv_obj_set_style_text_align(hr_value_label, LV_TEXT_ALIGN_CENTER, 0);

bpm_label = lv_label_create(hr_container);
lv_label_set_text(bpm_label, "bpm");
lv_obj_set_style_text_font(bpm_label, &lv_font_montserrat_18, 0);
lv_obj_set_style_text_align(bpm_label, LV_TEXT_ALIGN_CENTER, 0);

lv_color_t zone_color = get_hr_zone_color(current_hr);
lv_obj_set_style_text_color(hr_value_label, zone_color, 0);
lv_obj_set_style_text_color(bpm_label, zone_color, 0);

lv_timer_create(hr_anim_timer_cb, 80, NULL);
}

#endif

```

Uno stile di scala circolare che simula un widget tramonto/alba

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES && LV_FONT_MONTSERATT_12 && LV_FONT_MONTSERATT_14 && LV_FONT_MONTSERATT_16 && LV_FONT_MONTSERATT_20

static void label_color_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    if(!draw_task) return;

    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_event_get_draw_dsc(draw_task);
    if(!base_dsc || base_dsc->part != LV_PART_INDICATOR) return;

    lv_draw_label_dsc_t * label_dsc = lv_draw_task_get_label_dsc(draw_task);
    if(!label_dsc || !label_dsc->text) return;

    const char * txt = label_dsc->text;

    if(lv_strcmp(txt, "06") == 0 || lv_strcmp(txt, "12") == 0 ||
       lv_strcmp(txt, "18") == 0 || lv_strcmp(txt, "24") == 0) {
        label_dsc->color = lv_color_white();
    }
    else {
        label_dsc->color = lv_palette_darken(LV_PALETTE_GREY, 1);
    }
}

void lv_example_scale_11(void)
{
    lv_obj_t * bg = lv_obj_create(lv_screen_active());
    lv_obj_set_size(bg, 210, 210);
    lv_obj_center(bg);
    lv_obj_set_style_radius(bg, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_bg_color(bg, lv_palette_darken(LV_PALETTE_GREY, 4), 0);
    lv_obj_set_style_bg_opa(bg, LV_OPA_COVER, 0);
    lv_obj_remove_flag(bg, LV_OBJ_FLAG_SCROLLABLE);
    lv_obj_set_style_pad_all(bg, 0, LV_PART_MAIN);

    lv_obj_t * scale = lv_scale_create(bg);
    lv_obj_center(scale);
    lv_obj_set_size(scale, 150, 150);
    lv_obj_set_style_arc_width(scale, 5, LV_PART_MAIN);

    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_OUTER);
    lv_scale_set_range(scale, 0, 24);
    lv_scale_set_total_tick_count(scale, 25);
    lv_scale_set_major_tick_every(scale, 1);
    lv_scale_set_angle_range(scale, 360);
    lv_scale_set_rotation(scale, 105);
    lv_scale_set_label_show(scale, true);
    lv_obj_set_style_text_font(scale, &lv_font_montserrat_12, LV_PART_INDICATOR);
    lv_obj_set_style_pad_radial(scale, -6, LV_PART_INDICATOR);

    /*Rotate the labels of the ticks*/
    lv_obj_set_style_transform_rotation(scale, LV_SCALE_LABEL_ROTATE_MATCH_TICKS | LV_SCALE_LABEL_ROTATE_KEEP_UPRIGHT,
                                      LV_PART_INDICATOR);

    /* Style for major ticks */
    static lv_style_t style_ticks;
    lv_style_init(&style_ticks);
    lv_style_set_line_color(&style_ticks, lv_palette_darken(LV_PALETTE_GREY, 1));
    lv_style_set_line_width(&style_ticks, 2);
    lv_style_set_width(&style_ticks, 10);
    lv_obj_add_style(scale, &style_ticks, LV_PART_INDICATOR);

    /* Style for NIGHT - blue */
    static lv_style_t style_night;

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_init(&style_night);
lv_style_set_arc_color(&style_night, lv_palette_main(LV_PALETTE_BLUE));

/* Style for DAY - dark yellow */
static lv_style_t style_day;
lv_style_init(&style_day);
lv_style_set_arc_color(&style_day, lv_palette_darken(LV_PALETTE_YELLOW, 3));

/* NIGHT section */
lv_scale_section_t * section_night1 = lv_scale_add_section(scale);
lv_scale_set_section_range(scale, section_night1, 17, 5);
lv_scale_set_section_style_main(scale, section_night1, &style_night);

/* DAY section */
lv_scale_section_t * section_day = lv_scale_add_section(scale);
lv_scale_set_section_range(scale, section_day, 5, 17);
lv_scale_set_section_style_main(scale, section_day, &style_day);

static const char * hour_labels[] = {
    "01", "02", "03", "04", "05",
    "06", "07", "08", "09", "10",
    "11", "12", "13", "14", "15",
    "16", "17", "18", "19", "20",
    "21", "22", "23", "24",
    NULL
};
lv_scale_set_text_src(scale, hour_labels);

lv_obj_add_flag(scale, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
lv_obj_add_event_cb(scale, label_color_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);

lv_obj_t * today = lv_label_create(bg);
lv_label_set_text(today, "TODAY");
lv_obj_set_style_text_font(today, &lv_font_montserrat_16, 0);
lv_obj_set_style_text_color(today, lv_color_white(), 0);
lv_obj_align(today, LV_ALIGN_TOP_MID, 0, 60);

lv_obj_t * sunrise_lbl = lv_label_create(bg);
lv_label_set_text(sunrise_lbl, "SUNRISE");
lv_obj_set_style_text_font(sunrise_lbl, &lv_font_montserrat_14, 0);
lv_obj_set_style_text_color(sunrise_lbl, lv_palette_main(LV_PALETTE_GREY), 0);
lv_obj_align(sunrise_lbl, LV_ALIGN_LEFT_MID, 37, -10);

lv_obj_t * sunrise_time = lv_label_create(bg);
lv_label_set_text(sunrise_time, "6:43");
lv_obj_set_style_text_font(sunrise_time, &lv_font_montserrat_20, 0);
lv_obj_set_style_text_color(sunrise_time, lv_color_white(), 0);
lv_obj_align_to(sunrise_time, sunrise_lbl, LV_ALIGN_OUT_BOTTOM_MID, 0, 2);

lv_obj_t * sunset_lbl = lv_label_create(bg);
lv_label_set_text(sunset_lbl, "SUNSET");
lv_obj_set_style_text_font(sunset_lbl, &lv_font_montserrat_14, 0);
lv_obj_set_style_text_color(sunset_lbl, lv_palette_main(LV_PALETTE_GREY), 0);
lv_obj_align(sunset_lbl, LV_ALIGN_RIGHT_MID, -37, -10);

lv_obj_t * sunset_time = lv_label_create(bg);
lv_label_set_text(sunset_time, "17:37");
lv_obj_set_style_text_font(sunset_time, &lv_font_montserrat_20, 0);
lv_obj_set_style_text_color(sunset_time, lv_color_white(), 0);
lv_obj_align_to(sunset_time, sunset_lbl, LV_ALIGN_OUT_BOTTOM_MID, 0, 2);
}

#endif

```

Uno stile di scala circolare che simula una bussola

```

#include "../lv_examples.h"
#ifndef LV_USE_SCALE || !LV_BUILD_EXAMPLES

static lv_obj_t * scale;
static lv_obj_t * label;

static const char * heading_to_cardinal(int32_t heading)
{
    /* Normalize heading to range [0, 360) */
    while(heading < 0) heading += 360;
    while(heading >= 360) heading -= 360;

    if(heading < 23) return "N";
    else if(heading < 68) return "NE";
    else if(heading < 113) return "E";
    else if(heading < 158) return "SE";
    else if(heading < 203) return "S";
    else if(heading < 248) return "SW";
    else if(heading < 293) return "W";
    else if(heading < 338) return "NW";

    return "N";
}

static void set_heading_value(void * obj, int32_t v)
{
    LV_UNUSED(obj);
    lv_scale_set_rotation(scale, 270 - v);
    lv_label_set_text_fmt(label, "%d\n%s", (int)v, heading_to_cardinal(v));
}

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
    lv_draw_line_dsc_t * line_draw_dsc = lv_draw_task_get_line_dsc(draw_task);
    if(base_dsc->part == LV_PART_INDICATOR) {
        if(label_draw_dsc) {
            if(base_dsc->id1 == 0) {
                label_draw_dsc->color = lv_palette_main(LV_PALETTE_RED);
            }
        }
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

        }
    }

    if(line_draw_dsc) {
        if(base_dsc->id1 == 60) {
            line_draw_dsc->color = lv_palette_main(LV_PALETTE_RED);
        }
    }
}

/***
 * A round scale style simulating a compass
 */
void lv_example_scale_l2(void)
{
    scale = lv_scale_create(lv_screen_active());

    lv_obj_set_size(scale, 200, 200);
    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_INNER);
    lv_obj_set_align(scale, LV_ALIGN_CENTER);

    lv_scale_set_total_tick_count(scale, 61);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_obj_set_style_line_width(scale, 3, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 0, 360);

    static const char * custom_labels[] = {"N", "30", "60", "E", "120", "150", "S", "210", "240", "W", "300", "330", NULL};
    lv_scale_set_text_src(scale, custom_labels);

    lv_scale_set_angle_range(scale, 360);
    lv_scale_set_rotation(scale, 270);

    lv_obj_add_event_cb(scale, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(scale, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);

    label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, 100);
    lv_obj_set_align(label, LV_ALIGN_CENTER);
    lv_label_set_text(label, "0%\n");
    lv_obj_set_style_text_align(label, LV_TEXT_ALIGN_CENTER, 0);

    set_heading_value(NULL, 0);

    symbol = lv_label_create(scale);
    lv_obj_set_align(symbol, LV_ALIGN_TOP_MID);
    lv_obj_set_y(symbol, 5);
    lv_label_set_text(symbol, LV_SYMBOL_UP);
    lv_obj_set_style_text_align(symbol, LV_TEXT_ALIGN_CENTER, 0);
    lv_obj_set_style_text_color(symbol, lv_palette_main(LV_PALETTE_RED), 0);

    lv_anim_t anim_scale;
    lv_anim_init(&anim_scale);
    lv_anim_set_var(&anim_scale, scale);
    lv_anim_set_exec_cb(&anim_scale, set_heading_value);
    lv_anim_set_duration(&anim_scale, 5000);
    lv_anim_set_repeat_delay(&anim_scale, 500);
    lv_anim_set_repeat_count(&anim_scale, LV_ANIM_REPEAT_INFINITE);
    lv_anim_set_reverse_duration(&anim_scale, 5000);
    lv_anim_set_reverse_delay(&anim_scale, 500);
    lv_anim_set_values(&anim_scale, 0, 360);
    lv_anim_start(&anim_scale);
}

#endif

```

3.7.24 Slider

Slider semplice

```

#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

static void slider_event_cb(lv_event_t * e);
static lv_obj_t * slider_label;

/***
 * A default slider with a label displaying the current value
 */
void lv_example_slider_1(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_obj_center(slider);
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);

    lv_obj_set_style_anim_duration(slider, 2000, 0);
    /*Create a label below the slider*/
    slider_label = lv_label_create(lv_screen_active());
    lv_label_set_text(slider_label, "0%");

    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
}

static void slider_event_cb(lv_event_t * e)
{
    lv_obj_t * slider = lv_event_get_target_obj(e);
    char buf[8];
    lv_snprintf(buf, sizeof(buf), "%d%", (int)lv_slider_get_value(slider));
    lv_label_set_text(slider_label, buf);
    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
}

#endif

```

Slider con stile personalizzato

```
#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

/***
 * Show how to style a slider.
 */
void lv_example_slider_2(void)
{
    /*Create a transition*/
    static const lv_style_prop_t props[] = {LV_STYLE_BG_COLOR, 0};
    static lv_style_transition_dsc_t transition_dsc;
    lv_style_transition_dsc_init(&transition_dsc, props, lv_anim_path_linear, 300, 0, NULL);

    static lv_style_t style_main;
    static lv_style_t style_indicator;
    static lv_style_t style_knob;
    static lv_style_t style_pressed_color;
    lv_style_init(&style_main);
    lv_style_set_bg_opa(&style_main, LV_OPA_COVER);
    lv_style_set_bg_color(&style_main, lv_color_hex3(0xbbbb));
    lv_style_set_radius(&style_main, LV_RADIUS_CIRCLE);
    lv_style_set_pad(&style_main, -2); /*Makes the indicator larger*/

    lv_style_init(&style_indicator);
    lv_style_set_bg_opa(&style_indicator, LV_OPA_COVER);
    lv_style_set_bg_color(&style_indicator, lv_palette_main(LV_PALETTE_CYAN));
    lv_style_set_radius(&style_indicator, LV_RADIUS_CIRCLE);
    lv_style_set_transition(&style_indicator, &transition_dsc);

    lv_style_init(&style_knob);
    lv_style_set_bg_opa(&style_knob, LV_OPA_COVER);
    lv_style_set_bg_color(&style_knob, lv_palette_main(LV_PALETTE_CYAN));
    lv_style_set_border_color(&style_knob, lv_palette_darken(LV_PALETTE_CYAN, 3));
    lv_style_set_border_width(&style_knob, 2);
    lv_style_set_radius(&style_knob, LV_RADIUS_CIRCLE);
    lv_style_set_pad_all(&style_knob, 6); /*Makes the knob larger*/
    lv_style_set_transition(&style_knob, &transition_dsc);

    lv_style_init(&style_pressed_color);
    lv_style_set_bg_color(&style_pressed_color, lv_palette_darken(LV_PALETTE_CYAN, 2));

    /*Create a slider and add the style*/
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_obj_remove_style_all(slider); /*Remove the styles coming from the theme*/

    lv_obj_add_style(slider, &style_main, LV_PART_MAIN);
    lv_obj_add_style(slider, &style_indicator, LV_PART_INDICATOR);
    lv_obj_add_style(slider, &style_pressed_color, LV_PART_INDICATOR | LV_STATE_PRESSED);
    lv_obj_add_style(slider, &style_knob, LV_PART_KNOB);
    lv_obj_add_style(slider, &style_pressed_color, LV_PART_KNOB | LV_STATE_PRESSED);

    lv_obj_center(slider);
}

#endif
```

Slider estensibile

```
#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

#define MAX_VALUE 100
#define MIN_VALUE 0

static void slider_event_cb(lv_event_t * e);

/***
 * Show the current value when the slider is pressed by extending the drawer
 */
void lv_example_slider_3(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider;
    slider = lv_slider_create(lv_screen_active());
    lv_obj_center(slider);

    lv_slider_set_mode(slider, LV_SLIDER_MODE_RANGE);
    lv_slider_set_range(slider, MIN_VALUE, MAX_VALUE);
    lv_slider_set_value(slider, 70, LV_ANIM_OFF);
    lv_slider_set_start_value(slider, 20, LV_ANIM_OFF);

    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_ALL, NULL);
    lv_obj_refresh_ext_draw_size(slider);
}

static void slider_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);

    /*Provide some extra space for the value*/
    if(code == LV_EVENT_REFR_EXT_DRAW_SIZE) {
        lv_event_set_ext_draw_size(e, 50);
    }
    else if(code == LV_EVENT_DRAW_MAIN_END) {
        if(!lv_obj_has_state(obj, LV_STATE_PRESSED)) return;

        lv_area_t slider_area;
        lv_obj_get_coords(obj, &slider_area);
        lv_area_t indic_area = slider_area;
        lv_area_set_width(indic_area, lv_area_get_width(&slider_area) * lv_slider_get_value(obj) / MAX_VALUE);
        indic_area.x1 += lv_area_get_width(&slider_area) * lv_slider_get_left_value(obj) / MAX_VALUE;
        char buf[16];
        lv_snprintf(buf, sizeof(buf), "%d - %d", (int)lv_slider_get_left_value(obj), (int)lv_slider_get_value(obj));

        lv_point_t label_size;
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_text_get_size(&label_size, buf, LV_FONT_DEFAULT, 0, 0, LV_COORD_MAX, LV_TEXT_FLAG_NONE);
lv_area_t label_area;
label_area.x1 = 0;
label_area.x2 = label_size.x - 1;
label_area.y1 = 0;
label_area.y2 = label_size.y - 1;

lv_area_align(&indic_area, &label_area, LV_ALIGN_OUT_TOP_MID, 0, -10);

lv_draw_label_dsc_t label_draw_dsc;
lv_draw_label_dsc_init(&label_draw_dsc);
label_draw_dsc.color = lv_color_hex3(0x888);
label_draw_dsc.text = buf;
label_draw_dsc.text_local = true;
label_layer_t * layer = lv_event_get_layer(e);
lv_draw_label(layer, &label_draw_dsc, &label_area);
}

#endif
}

```

Slider con direzione opposta

```

#include "../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

static void slider_event_cb(lv_event_t * e);
static lv_obj_t * slider_label;

/***
 * Slider with opposite direction
 */
void lv_example_slider_4(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_obj_center(slider);
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    /*Reverse the direction of the slider*/
    lv_slider_set_range(slider, 100, 0);
    /*Create a label below the slider*/
    slider_label = lv_label_create(lv_screen_active());
    lv_label_set_text(slider_label, "0%");

    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
}

static void slider_event_cb(lv_event_t * e)
{
    lv_obj_t * slider = lv_event_get_target_obj(e);
    char buf[8];
    lv_snprintf(buf, sizeof(buf), "%d%%", (int)lv_slider_get_value(slider));
    lv_label_set_text(slider_label, buf);
    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
}

#endif
}

```

3.7.25 Span

Span con stili personalizzati

```

#include "../lv_examples.h"
#if LV_USE_SPAN && LV_BUILD_EXAMPLES

static void click_event_cb(lv_event_t * e)
{
    lv_indev_t * indev = lv_event_get_indev(e);
    lv_point_t point;
    lv_indev_get_point(indev, &point);
#ifndef LV_USE_LOG
    lv_obj_t * spans = lv_event_get_target_obj(e);
    lv_span_t * span = lv_spangroup_get_span_by_point(spans, &point);
    LV_LOG_USERC("%s", span ? lv_span_get_text(span) : "NULL");
#endif
}

/***
 * Create spans and get clicked one
 */
void lv_example_span_1(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_border_width(&style, 1);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_ORANGE));
    lv_style_set_pad_all(&style, 2);

    lv_obj_t * spans = lv_spangroup_create(lv_screen_active());
    /* Setting a fixed width and height to LV_SIZE_CONTENT will make the text wrap */
    lv_obj_set_width(spans, 300);
    lv_obj_set_height(spans, LV_SIZE_CONTENT);
    lv_obj_center(spans);
    lv_obj_add_style(spans, &style, 0);
    lv_obj_add_flag(spans, LV_OBJ_FLAG_CLICKABLE);

    lv_spangroup_set_align(spans, LV_TEXT_ALIGN_LEFT);
    lv_spangroup_set_overflow(spans, LV_SPAN_OVERFLOW_CLIP);
    lv_spangroup_set_indent(spans, 20);

    lv_span_t * span = lv_spangroup_add_span(spans);
    lv_span_set_text(span, "China is a beautiful country.");
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_set_text_color(lv_span_get_style(span), lv_palette_main(LV_PALETTE_RED));
lv_style_set_text_decor(lv_span_get_style(span), LV_TEXT_DECOR_UNDERLINE);
lv_style_set_text_opa(lv_span_get_style(span), LV_OPA_50);

span = lv_spangroup_add_span(spans);
lv_span_set_text_static(span, "good good study, day day up.");
#if LV_FONT_MONTSERRAT_24
lv_style_set_text_font(lv_span_get_style(span), &lv_font_montserrat_24);
#endif
lv_style_set_text_color(lv_span_get_style(span), lv_palette_main(LV_PALETTE_GREEN));

span = lv_spangroup_add_span(spans);
lv_span_set_text_static(span, "LVGL is an open-source graphics library.");
lv_style_set_text_color(lv_span_get_style(span), lv_palette_main(LV_PALETTE_BLUE));

span = lv_spangroup_add_span(spans);
lv_span_set_text_static(span, "the boy no name.");
lv_style_set_text_color(lv_span_get_style(span), lv_palette_main(LV_PALETTE_GREEN));
#if LV_FONT_MONTSERRAT_20
lv_style_set_text_font(lv_span_get_style(span), &lv_font_montserrat_20);
#endif
lv_style_set_text_decor(lv_span_get_style(span), LV_TEXT_DECOR_UNDERLINE);

span = lv_spangroup_add_span(spans);
lv_span_set_text(span, "I have a dream that hope to come true.");
lv_style_set_text_decor(lv_span_get_style(span), LV_TEXT_DECOR_STRIKETHROUGH);

lv_spangroup_refresh(spans);

lv_obj_add_event_cb(spans, click_event_cb, LV_EVENT_CLICKED, NULL);
}

#endif

```

3.7.26 Spinbox

Spinbox semplice

```

#include "../../lv_examples.h"
#if LV_USE_SPINBOX && LV_BUILD_EXAMPLES

static lv_obj_t * spinbox;

static void lv_spinbox_increment_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_SHORT_CLICKED || code == LV_EVENT_LONG_PRESSED_REPEAT) {
        lv_spinbox_increment(spinbox);
    }
}

static void lv_spinbox_decrement_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_SHORT_CLICKED || code == LV_EVENT_LONG_PRESSED_REPEAT) {
        lv_spinbox_decrement(spinbox);
    }
}

void lv_example_spinbox_1(void)
{
    spinbox = lv_spinbox_create(lv_screen_active());
    lv_spinbox_set_range(spinbox, -1000, 25000);
    lv_spinbox_set_digit_count(spinbox, 5);
    lv_spinbox_set_dec_point_pos(spinbox, 2);
    lv_spinbox_step_prev(spinbox);
    lv_obj_set_width(spinbox, 100);
    lv_obj_center(spinbox);

    int32_t h = lv_obj_get_height(spinbox);

    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, h, h);
    lv_obj_align_to(btn, spinbox, LV_ALIGN_OUT_RIGHT_MID, 5, 0);
    lv_obj_set_style_bg_image_src(btn, LV_SYMBOL_PLUS, 0);
    lv_obj_add_event_cb(btn, lv_spinbox_increment_event_cb, LV_EVENT_ALL, NULL);

    btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, h, h);
    lv_obj_align_to(btn, spinbox, LV_ALIGN_OUT_LEFT_MID, -5, 0);
    lv_obj_set_style_bg_image_src(btn, LV_SYMBOL_MINUS, 0);
    lv_obj_add_event_cb(btn, lv_spinbox_decrement_event_cb, LV_EVENT_ALL, NULL);
}
#endif

```

3.7.27 Spinner

Spinner semplice

```

#include "../../lv_examples.h"
#if LV_USE_SPINNER && LV_BUILD_EXAMPLES

void lv_example_spinner_1(void)
{
    /*Create a spinner*/
    lv_obj_t * spinner = lv_spinner_create(lv_screen_active());
    lv_obj_set_size(spinner, 100, 100);
    lv_obj_center(spinner);
    lv_spinner_set_anim_params(spinner, 10000, 200);
}

```

(continues on next page)

(continua dalla pagina precedente)

#endif

3.7.28 Switch

Switch semplice

```
#include "../../lv_examples.h"
#if LV_USE_SWITCH && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        LV_UNUSED(obj);
        LV_LOG_USER("State: %s\n", lv_obj_has_state(obj, LV_STATE_CHECKED) ? "On" : "Off");
    }
}

void lv_example_switch_1(void)
{
    lv_obj_set.flex_flow(lv_screen_active(), LV_FLEX_FLOW_COLUMN);
    lv_obj_set.flex_align(lv_screen_active(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    lv_obj_t * sw;

    sw = lv_switch_create(lv_screen_active());
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);
    lv_obj_add_flag(sw, LV_OBJ_FLAG_EVENT_BUBBLE);

    sw = lv_switch_create(lv_screen_active());
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);

    sw = lv_switch_create(lv_screen_active());
    lv_obj_add_state(sw, LV_STATE_DISABLED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);

    sw = lv_switch_create(lv_screen_active());
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_state(sw, LV_STATE_DISABLED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);
}

#endif
```

Orientamento dello Switch

```
#include "../../lv_examples.h"
#if LV_USE_SWITCH && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        LV_UNUSED(obj);
        LV_LOG_USER("State: %s\n", lv_obj_has_state(obj, LV_STATE_CHECKED) ? "On" : "Off");
    }
}

void lv_example_switch_2(void)
{
    lv_obj_set.flex_flow(lv_screen_active(), LV_FLEX_FLOW_ROW);
    lv_obj_set.flex_align(lv_screen_active(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    lv_obj_t * sw;

    sw = lv_switch_create(lv_screen_active());
    lv_obj_set_size(sw, 30, 60);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);

    sw = lv_switch_create(lv_screen_active());
    lv_obj_set_size(sw, 30, 60);
    lv_switch_set_orientation(sw, LV_SWITCH_ORIENTATION_VERTICAL);
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);
}

#endif
```

3.7.29 Tabella

Tabella semplice

```
#include "../../lv_examples.h"
#if LV_USE_TABLE && LV_BUILD_EXAMPLES

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    /*If the cells are drawn...*/
    if(base_dsc->part == LV_PART_ITEMS) {
        uint32_t row = base_dsc->id1;
        uint32_t col = base_dsc->id2;
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

/*Make the texts in the first cell center aligned*/
if(row == 0) {
    lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
    if(label_draw_dsc) {
        label_draw_dsc->align = LV_TEXT_ALIGN_CENTER;
    }
    lv_draw_fill_dsc_t * fill_draw_dsc = lv_draw_task_get_fill_dsc(draw_task);
    if(fill_draw_dsc) {
        fill_draw_dsc->color = lv_color_mix(lv_palette_main(LV_PALETTE_BLUE), fill_draw_dsc->color, LV_OPA_20);
        fill_draw_dsc->opa = LV_OPA_COVER;
    }
}
/*In the first column align the texts to the right*/
else if(col == 0) {
    lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
    if(label_draw_dsc) {
        label_draw_dsc->align = LV_TEXT_ALIGN_RIGHT;
    }
}

/*Make every 2nd row grayish*/
if((row != 0 && row % 2) == 0) {
    lv_draw_fill_dsc_t * fill_draw_dsc = lv_draw_task_get_fill_dsc(draw_task);
    if(fill_draw_dsc) {
        fill_draw_dsc->color = lv_color_mix(lv_palette_main(LV_PALETTE_GREY), fill_draw_dsc->color, LV_OPA_10);
        fill_draw_dsc->opa = LV_OPA_COVER;
    }
}
}

void lv_example_table_1(void)
{
    lv_obj_t * table = lv_table_create(lv_screen_active());

    /*Fill the first column*/
    lv_table_set_cell_value(table, 0, 0, "Name");
    lv_table_set_cell_value(table, 1, 0, "Apple");
    lv_table_set_cell_value(table, 2, 0, "Banana");
    lv_table_set_cell_value(table, 3, 0, "Lemon");
    lv_table_set_cell_value(table, 4, 0, "Grape");
    lv_table_set_cell_value(table, 5, 0, "Melon");
    lv_table_set_cell_value(table, 6, 0, "Peach");
    lv_table_set_cell_value(table, 7, 0, "Nuts");

    /*Fill the second column*/
    lv_table_set_cell_value(table, 0, 1, "Price");
    lv_table_set_cell_value(table, 1, 1, "$7");
    lv_table_set_cell_value(table, 2, 1, "$4");
    lv_table_set_cell_value(table, 3, 1, "$6");
    lv_table_set_cell_value(table, 4, 1, "$2");
    lv_table_set_cell_value(table, 5, 1, "$5");
    lv_table_set_cell_value(table, 6, 1, "$1");
    lv_table_set_cell_value(table, 7, 1, "$9");

    /*Set a smaller height to the table. It'll make it scrollable*/
    lv_obj_set_height(table, 200);
    lv_obj_center(table);

    /*Add an event callback to apply some custom drawing*/
    lv_obj_add_event_cb(table, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(table, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
}

#endif

```

Elenco leggero da tabella

```

#include "../../lv_examples.h"
#if LV_USE_TABLE && LV_BUILD_EXAMPLES

#define ITEM_CNT 200

static void draw_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);

    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    /*If the cells are drawn...*/
    if(base_dsc->part == LV_PART_ITEMS && lv_draw_task_get_type(draw_task) == LV_DRAW_TASK_TYPE_FILL) {
        /*Draw the background*/
        bool chk = lv_table_has_cell_ctrl(obj, base_dsc->id1, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
        lv_draw_rect_dsc_t rect_dsc;
        lv_draw_rect_dsc_init(&rect_dsc);
        rect_dsc.bg_color = chk ? lv_theme_get_color_primary(obj) : lv_palette_lighten(LV_PALETTE_GREY, 2);
        rect_dsc.radius = LV_RADIUS_CIRCLE;

        lv_area_t sw_area;
        sw_area.x1 = 0;
        sw_area.x2 = 40;
        sw_area.y1 = 0;
        sw_area.y2 = 24;
        lv_area_t draw_task_area;
        lv_draw_task_get_area(draw_task, &draw_task_area);
        lv_area_align(&draw_task_area, &sw_area, LV_ALIGN_RIGHT_MID, -15, 0);
        lv_draw_rect(base_dsc->layer, &rect_dsc, &sw_area);

        /*Draw the knob*/
        rect_dsc.bg_color = lv_color_white();
        lv_area_t knob_area;
        knob_area.x1 = 0;
        knob_area.x2 = 18;
        knob_area.y1 = 0;
        knob_area.y2 = 18;
        if(chk) {
            lv_area_align(&sw_area, &knob_area, LV_ALIGN_RIGHT_MID, -3, 0);
        }
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

    }
    else {
        lv_area_align(&sw_area, &knob_area, LV_ALIGN_LEFT_MID, 3, 0);
    }
    lv_draw_rect(base_dsc->layer, &rect_dsc, &knob_area);
}

static void change_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    uint32_t col;
    uint32_t row;
    lv_table_get_selected_cell(obj, &row, &col);
    bool chk = lv_table_has_cell_ctrl(obj, row, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
    if(chk) lv_table_clear_cell_ctrl(obj, row, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
    else lv_table_set_cell_ctrl(obj, row, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
}

/**
 * A very light-weighted list created from table
 */
void lv_example_table_2(void)
{
    /*Measure memory usage*/
    lv_mem_monitor_t mon1;
    lv_mem_monitor(&mon1);

    uint32_t t = lv_tick_get();

    lv_obj_t * table = lv_table_create(lv_screen_active());

    /*Set a smaller height to the table. It'll make it scrollable*/
    lv_obj_set_size(table, LV_SIZE_CONTENT, 200);

    lv_table_set_column_width(table, 0, 150);
    lv_table_set_row_count(table, ITEM_CNT); /*Not required but avoids a lot of memory reallocation lv_table_set_set_value*/
    lv_table_set_column_count(table, 1);

    /*Don't make the cell pressed, we will draw something different in the event*/
    lv_obj_remove_style(table, NULL, LV_PART_ITEMS | LV_STATE_PRESSED);

    uint32_t i;
    for(i = 0; i < ITEM_CNT; i++) {
        lv_table_set_cell_value_fmt(table, i, 0, "Item %" LV_PRIu32, i + 1);
    }

    lv_obj_align(table, LV_ALIGN_CENTER, 0, -20);

    /*Add an event callback to to apply some custom drawing*/
    lv_obj_add_event_cb(table, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_event_cb(table, change_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_add_flag(table, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);

    lv_mem_monitor_t mon2;
    lv_mem_monitor(&mon2);

    size_t mem_used = mon1.free_size - mon2.free_size;

    uint32_t elaps = lv_tick_elaps(t);

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text_fmt(label, "%s LV_PRIu32" " items were created in %" LV_PRIu32" ms\n"
                           "using %zu bytes of memory",
                           (uint32_t)ITEM_CNT, elaps, mem_used);

    lv_obj_align(label, LV_ALIGN_BOTTOM_MID, 0, -10);
}
#endif

```

3.7.30 Tabview

Tabview Semplice

```

#include "../../lv_examples.h"
#ifndef LV_USE_TABVIEW || !LV_BUILD_EXAMPLES

void lv_example_tabview_1(void)
{
    /*Create a Tab view object*/
    lv_obj_t * tabview;
    tabview = lv_tabview_create(lv_screen_active());

    /*Add 3 tabs (the tabs are page (lv_page) and can be scrolled*/
    lv_obj_t * tab1 = lv_tabview_add_tab(tabview, "Tab 1");
    lv_obj_t * tab2 = lv_tabview_add_tab(tabview, "Tab 2");
    lv_obj_t * tab3 = lv_tabview_add_tab(tabview, "Tab 3");

    /*Add content to the tabs*/
    lv_obj_t * label = lv_label_create(tab1);
    lv_label_set_text(label, "This the first tab\n\n"
                           "If the content\n"
                           "of a tab\n"
                           "becomes too\n"
                           "longer\n"
                           "than the\n"
                           "container\n"
                           "then it\n"
                           "automatically\n"
                           "becomes\n"
                           "scrollable.\n"
                           "\n"
                           "\n"
                           "\n"
                           "Can you see it?");

}

```

(continues on next page)

(continua dalla pagina precedente)

```

label = lv_label_create(tab2);
lv_label_set_text(label, "Second tab");

label = lv_label_create(tab3);
lv_label_set_text(label, "Third tab");

lv_obj_scroll_to_view_recursive(label, LV_ANIM_ON);

}
#endif

```

Tab a sinistra, con stile e nessuno scorrimento

```

#include "../../lv_examples.h"
#if LV_USE_TABVIEW && LV_BUILD_EXAMPLES

/*A vertical tab view with disabled scrolling and some styling*/
void lv_example_tabview_2(void)
{
    /*Create a Tab view object*/
    lv_obj_t * tabview;
    uint32_t tab_count = 0;
    uint32_t i = 0;

    tabview = lv_tabview_create(lv_screen_active());
    lv_tabview_set_tab_bar_position(tabview, LV_DIR_LEFT);
    lv_tabview_set_tab_bar_size(tabview, 80);

    lv_obj_set_style_bg_color(tabview, lv_palette_lighten(LV_PALETTE_RED, 2), 0);

    lv_obj_t * tab_buttons = lv_tabview_get_tab_bar(tabview);
    lv_obj_set_style_bg_color(tab_buttons, lv_palette_darken(LV_PALETTE_GREY, 3), 0);
    lv_obj_set_style_text_color(tab_buttons, lv_palette_lighten(LV_PALETTE_GREY, 5), 0);

    /*Add 5 tabs (the tabs are page (lv_page) and can be scrolled*/
    lv_obj_t * tab1 = lv_tabview_add_tab(tabview, "Tab 1");
    lv_obj_t * tab2 = lv_tabview_add_tab(tabview, "Tab 2");
    lv_obj_t * tab3 = lv_tabview_add_tab(tabview, "Tab 3");
    lv_obj_t * tab4 = lv_tabview_add_tab(tabview, "Tab 4");
    lv_obj_t * tab5 = lv_tabview_add_tab(tabview, "Tab 5");

    tab_count = lv_tabview_get_tab_count(tabview);
    for(i = 0; i < tab_count; i++) {
        lv_obj_t * button = lv_obj_get_child(tab_buttons, i);
        lv_obj_set_style_border_side(button, LV_BORDER_SIDE_RIGHT, LV_PART_MAIN | LV_STATE_CHECKED);
    }
    lv_obj_set_style_bg_color(tab2, lv_palette_lighten(LV_PALETTE_AMBER, 3), 0);
    lv_obj_set_style_bg_opa(tab2, LV_OPA_COVER, 0);

    /*Add content to the tabs*/
    lv_obj_t * label = lv_label_create(tab1);
    lv_label_set_text(label, "First tab");

    label = lv_label_create(tab2);
    lv_label_set_text(label, "Second tab");

    label = lv_label_create(tab3);
    lv_label_set_text(label, "Third tab");

    label = lv_label_create(tab4);
    lv_label_set_text(label, "Fourth tab");

    label = lv_label_create(tab5);
    lv_label_set_text(label, "Fifth tab");

    lv_obj_remove_flag(lv_tabview_get_content(tabview), LV_OBJ_FLAG_SCROLLABLE);
}
#endif

```

3.7.31 Area di testo

Area di testo semplice

```

#include "../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_BUILD_EXAMPLES

static void textarea_event_handler(lv_event_t * e)
{
    lv_obj_t * ta = lv_event_get_target_obj(e);
    LV_UNUSED(ta);
    LV_LOG_USER("Enter was pressed. The current text is: %s", lv_textarea_get_text(ta));
}

static void btm_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_obj_t * ta = (lv_obj_t *)lv_event_get_user_data(e);
    const char * txt = lv_buttonmatrix_get_button_text(obj), lv_buttonmatrix_get_selected_button(obj));

    if(lv_strcmp(txt, LV_SYMBOL_BACKSPACE) == 0) lv_textarea_delete_char(ta);
    else if(lv_strcmp(txt, LV_SYMBOL_NEW_LINE) == 0) lv_obj_send_event(ta, LV_EVENT_READY, NULL);
    else lv_textarea_add_text(ta, txt);
}

void lv_example_textarea_1(void)
{
    lv_obj_t * ta = lv_textarea_create(lv_screen_active());
    lv_textarea_set_one_line(ta, true);
    lv_obj_align(ta, LV_ALIGN_TOP_MID, 0, 10);
    lv_obj_add_event_cb(ta, textarea_event_handler, LV_EVENT_READY, ta);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_add_state(ta, LV_STATE_FOCUSED); /*To be sure the cursor is visible*/

static const char * btm_map[] = {"1", "2", "3", "\n",
                                "4", "5", "6", "\n",
                                "7", "8", "9", "\n",
                                LV_SYMBOL_BACKSPACE, "0", LV_SYMBOL_NEW_LINE, ""};

lv_obj_t * btm = lv_buttonmatrix_create(lv_screen_active());
lv_obj_set_size(btm, 200, 150);
lv_obj_align(btm, LV_ALIGN_BOTTOM_MID, 0, -10);
lv_obj_add_event_cb(btm, btm_event_handler, LV_EVENT_VALUE_CHANGED, ta);
lv_obj_remove_flag(btm, LV_OBJ_FLAG_CLICK_FOCUSABLE); /*To keep the text area focused on button clicks*/
lv_buttonmatrix_set_map(btm, btm_map);
}

#endif

```

Area di testo con campo password

```

#include "../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e);

static lv_obj_t * kb;

void lv_example_textarea_2(void)
{
    /*Create the password box*/
    lv_obj_t * pwd_ta = lv_textarea_create(lv_screen_active());
    lv_textarea_set_text(pwd_ta, "");
    lv_textarea_set_password_mode(pwd_ta, true);
    lv_textarea_set_one_line(pwd_ta, true);
    lv_obj_set_width(pwd_ta, lv_pct(40));
    lv_obj_set_pos(pwd_ta, 5, 20);
    lv_obj_add_event_cb(pwd_ta, ta_event_cb, LV_EVENT_ALL, NULL);

    /*Create a label and position it above the text box*/
    lv_obj_t * pwd_label = lv_label_create(lv_screen_active());
    lv_label_set_text(pwd_label, "Password:");
    lv_obj_align_to(pwd_label, pwd_ta, LV_ALIGN_OUT_TOP_LEFT, 0, 0);

    /*Create the one-line mode text area*/
    lv_obj_t * text_ta = lv_textarea_create(lv_screen_active());
    lv_textarea_set_one_line(text_ta, true);
    lv_textarea_set_password_mode(text_ta, false);
    lv_obj_set_width(text_ta, lv_pct(40));
    lv_obj_add_event_cb(text_ta, ta_event_cb, LV_EVENT_ALL, NULL);
    lv_obj_align(text_ta, LV_ALIGN_TOP_RIGHT, -5, 20);

    /*Create a label and position it above the text box*/
    lv_obj_t * oneline_label = lv_label_create(lv_screen_active());
    lv_label_set_text(oneline_label, "Text:");
    lv_obj_align_to(oneline_label, text_ta, LV_ALIGN_OUT_TOP_LEFT, 0, 0);

    /*Create a keyboard*/
    kb = lv_keyboard_create(lv_screen_active());
    lv_obj_set_size(kb, LV_HOR_RES, LV_VER_RES / 2);

    lv_keyboard_set_textarea(kb, pwd_ta); /*Focus it on one of the text areas to start*/

    /*The keyboard will show Arabic characters if they are enabled */
    #if LV_USE_ARABIC_PERSIAN_CHARS && LV_FONT_DEJAVU_16_PERSIAN_HEBREW
    lv_obj_set_style_text_font(kb, &lv_font_dejavu_16_persian_hebrew, 0);
    lv_obj_set_style_text_font(text_ta, &lv_font_dejavu_16_persian_hebrew, 0);
    lv_obj_set_style_text_font(pwd_ta, &lv_font_dejavu_16_persian_hebrew, 0);
    #endif
}

static void ta_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * ta = lv_event_get_target_obj(e);
    if(code == LV_EVENT_CLICKED || code == LV_EVENT_FOCUSED) {
        /*Focus on the clicked text area*/
        if(kb != NULL) lv_keyboard_set_textarea(kb, ta);
    }

    else if(code == LV_EVENT_READY) {
        LV_LOG_USER("Ready, current text: %s", lv_textarea_get_text(ta));
    }
}
#endif

```

Formattazione automatica del testo

```

#include "../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e);

static lv_obj_t * kb;

/***
 * Automatically format text like a clock. E.g. "12:34"
 * Add the ':' automatically.
 */
void lv_example_textarea_3(void)
{
    /*Create the text area*/
    lv_obj_t * ta = lv_textarea_create(lv_screen_active());

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_add_event_cb(ta, ta_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
lv_textarea_set_accepted_chars(ta, "0123456789:");
lv_textarea_set_max_length(ta, 5);
lv_textarea_set_one_line(ta, true);
lv_textarea_set_text(ta, "");

/*Create a keyboard*/
kb = lv_keyboard_create(lv_screen_active());
lv_obj_set_size(kb, LV_HOR_RES, LV_VER_RES / 2);
lv_keyboard_set_mode(kb, LV_KEYBOARD_MODE_NUMBER);
lv_keyboard_set_textarea(kb, ta);
}

static void ta_event_cb(lv_event_t * e)
{
    lv_obj_t * ta = lv_event_get_target_obj(e);
    const char * txt = lv_textarea_get_text(ta);
    if(txt[0] >= '0' && txt[0] <= '9' &&
       txt[1] >= '0' && txt[1] <= '9' &&
       txt[2] != ':') {
        lv_textarea_set_cursor_pos(ta, 2);
        lv_textarea_add_char(ta, ':');
    }
}
#endif

```

Stile del cursore dell'area di testo

```

#include "../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void create_styled_textarea_cursor(const char * txt, int32_t y_ofs, lv_style_t * cursor_style)
{
    lv_obj_t * ta = lv_textarea_create(lv_screen_active());
    lv_textarea_set_text(ta, txt);
    lv_obj_set_width(ta, 280);
    lv_obj_align(ta, LV_ALIGN_TOP_MID, 0, y_ofs);
    lv_textarea_set_one_line(ta, true);
    lv_obj_add_state(ta, LV_STATE_FOCUSED);
    lv_obj_add_style(ta, cursor_style, LV_PART_CURSOR | LV_STATE_FOCUSED);
    lv_textarea_set_cursor_pos(ta, 0);
}

void lv_example_textarea_4(void)
{
    static lv_style_t style_simple, style_block, style_underline;

    /* Thin left bar cursor (simple) */
    lv_style_init(&style_simple);
    lv_style_set_border_color(&style_simple, lv_palette_main(LV_PALETTE_RED));

    /* Underline cursor */
    lv_style_init(&style_underline);
    lv_style_set_bg_opa(&style_underline, LV_OPA_TRANSP);
    lv_style_set_border_color(&style_underline, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_border_side(&style_underline, LV_BORDER_SIDE_BOTTOM);
    lv_style_set_pad_hori(&style_underline, 1); /* set width of cursor using pad */
    lv_style_set_border_width(&style_underline, 3); /* set thickness of underline cursor */

    /* Full block cursor with many styles */
    lv_style_init(&style_block);
    lv_style_set_bg_opa(&style_block, LV_OPA_COVER);
    lv_style_set_bg_color(&style_block, lv_palette_main(LV_PALETTE_ORANGE));
    lv_style_set_bg_grad_color(&style_block, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_bg_grad_dir(&style_block, LV_GRAD_DIR_VER);
    lv_style_set_border_color(&style_block, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_border_side(&style_block, LV_BORDER_SIDE_FULL);
    lv_style_set_border_width(&style_block, 1);
    lv_style_set_radius(&style_block, 4);
    lv_style_set_text_color(&style_block, lv_color_white());
    lv_style_set_pad_all(&style_block, 1); /* set width of cursor using pad */

    /* Create 3 independent textareas, each with a unique styled cursor */
    create_styled_textarea_cursor("This is a simple red cursor", 10, &style_simple);
    create_styled_textarea_cursor("This is an underline blue cursor", 110, &style_underline);
    create_styled_textarea_cursor("This is a complex block cursor", 60, &style_block);
}
#endif

```

3.7.32 Tileview

Tileview con contenuto

```

#include "../../lv_examples.h"
#if LV_USE_TILEVIEW && LV_BUILD_EXAMPLES

/***
 * Create a 2x2 tile view and allow scrolling only in an "L" shape.
 * Demonstrate scroll chaining with a long list that
 * scrolls the tile view when it can't be scrolled further.
 */
void lv_example_tileview_1(void)
{
    lv_obj_t * tv = lv_tileview_create(lv_screen_active());

    /*Tile1: just a label*/
    lv_obj_t * tile1 = lv_tileview_add_tile(tv, 0, 0, LV_DIR_BOTTOM);
    lv_obj_t * label = lv_label_create(tile1);
    lv_label_set_text(label, "Scroll down");
    lv_obj_center(label);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

/*Tile2: a button*/
lv_obj_t * tile2 = lv_tileview_add_tile(tv, 0, 1, (lv_dir_t)(LV_DIR_TOP | LV_DIR_RIGHT));

lv_obj_t * btn = lv_button_create(tile2);

label = lv_label_create(btn);
lv_label_set_text(label, "Scroll up or right");

lv_obj_set_size(btn, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
lv_obj_center(btn);

/*Tile3: a list*/
lv_obj_t * tile3 = lv_tileview_add_tile(tv, 1, 1, LV_DIR_LEFT);
lv_obj_t * list = lv_list_create(tile3);
lv_obj_set_size(list, LV_PCT(100), LV_PCT(100));

lv_list_add_button(list, NULL, "One");
lv_list_add_button(list, NULL, "Two");
lv_list_add_button(list, NULL, "Three");
lv_list_add_button(list, NULL, "Four");
lv_list_add_button(list, NULL, "Five");
lv_list_add_button(list, NULL, "Six");
lv_list_add_button(list, NULL, "Seven");
lv_list_add_button(list, NULL, "Eight");
lv_list_add_button(list, NULL, "Nine");
lv_list_add_button(list, NULL, "Ten");

}

#endif

```

3.7.33 Window

Window semplice

```

#include "../../lv_examples.h"
#if LV_USE_WIN && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    LV_UNUSED(obj);
    LV_LOG_USER("Button %d clicked", (int)lv_obj_get_index(obj));
}

void lv_example_win_1(void)
{
    lv_obj_t * win = lv_win_create(lv_screen_active());
    lv_obj_t * btn;
    btn = lv_win_add_button(win, LV_SYMBOL_LEFT, 40);
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_win_add_title(win, "A title");

    btn = lv_win_add_button(win, LV_SYMBOL_RIGHT, 40);
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    btn = lv_win_add_button(win, LV_SYMBOL_CLOSE, 60);
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_obj_t * cont = lv_win_get_content(win); /*Content can be added here*/
    lv_obj_t * label = lv_label_create(cont);
    lv_label_set_text(label, "This isn't\n"
                           "a pretty\n"
                           "long text\n"
                           "to see how\n"
                           "the window\n"
                           "becomes\n"
                           "scrollable.\n"
                           "\n"
                           "\n"
                           "Some more\n"
                           "text to be\n"
                           "sure it\n"
                           "overflows. .:)");
}

#endif

```

CAPITOLO 4

Integrazione di LVGL

4.1 Panoramica

4.1.1 Ottenerne LVGL

LVGL è disponibile su GitHub: <https://github.com/lvgl/lvgl>.

Lo si può clonare o eseguire il [Download](#) dell'ultima versione della libreria da GitHub.

La libreria grafica vera e propria si trova nella directory `lvgl`. Contiene diverse directory, ma per usare LVGL sono necessari solo i file `.c` e `.h` nella directory `src`, più `lvgl/lvgl.h`, e `lvgl/lv_version.h`.

Demo ed Esempi

La directory `lvgl` contiene anche le directory `examples` e `demos`. Se il progetto necessita di esempi e/o demo, aggiungere queste directory al progetto. Se `make` o `CMake` gestiscono le directory `examples` e `demos`, non è richiesta alcuna azione aggiuntiva.

4.1.2 Compilazione di LVGL

Make e CMake

LVGL è già pronto per supportare i sistemi di compilazione `make` e `CMake`. Per aggiungere LVGL al sistema basato su `Makefile`, si aggiungono queste righe al `Makefile` principale:

```
LVGL_DIR_NAME ?= lvgl      #The name of the lvgl folder (change this if you have renamed it)
LVGL_DIR ?= ${shell pwd}  #The path where the lvgl folder is
include $(LVGL_DIR)/$(LVGL_DIR_NAME)/lvgl.mk
```

Per l'integrazione con `CMake`, consultare questa sezione della [Documentazione](#).

Build gestite

TODO [da fare]

4.1.3 Configurazione

lv_conf.h

Creazione di lv_conf.h

Quando si imposta il progetto per la prima volta, si deve copiare `lvgl/lv_conf_template.h` in `lv_conf.h` accanto alla cartella `lvgl`. Si modifica il primo `#if 0` a `1` per abilitare il contenuto del file e si imposta la definizione `LV_COLOR_DEPTH` in modo che si allinei alla profondità di colore utilizzata dal proprio pannello di visualizzazione. Per i dettagli vedere i commenti in `lv_conf.h`.

Il layout dei file dovrebbe apparire così:

```
lvgl/
lv_conf.h
other files and folders in your project
```

In alternativa, `lv_conf.h` può essere copiato in un'altra posizione, ma in tal caso è necessario aggiungere la definizione `LV_CONF_INCLUDE_SIMPLE` alle opzioni del compilatore (ad esempio `-DLV_CONF_INCLUDE_SIMPLE` per il compilatore GCC) e impostare manualmente il path dell'include (ad esempio `-I../include/gui`). In questo caso, LVGL tenterà di includere `lv_conf.h` semplicemente con `#include "lv_conf.h"`.

È anche possibile utilizzare un nome diverso per `lv_conf.h`. Il path personalizzato può essere impostato tramite la definizione `LV_CONF_PATH`. Per esempio `-DLV_CONF_PATH="/home/joe/my_project/my_custom_conf.h"`. Se questa definizione è settata, `LV_CONF_SKIP` si presume sia `0`. Si noti che, quando si definisce `LV_CONF_PATH`, è necessario assicurarsi che sia definito come una stringa, altrimenti verrà generato un errore di compilazione.

Se `LV_CONF_SKIP` è definito, LVGL non tenterà di includere `lv_conf.h`. Invece, è possibile passare le definizioni di configurazione utilizzando le opzioni di compilazione. Ad esempio `"-DLV_COLOR_DEPTH=32 -DLV_USE_BUTTON=1"`. Le opzioni non impostate otterranno un valore predefinito che è lo stesso del contenuto di `lv_conf_template.h`.

Impostazioni di Configurazione

Una volta installato il file `lv_conf.h`, è possibile modificare questo header per configurare il comportamento di LVGL, disabilitare moduli e funzionalità inutilizzati, regolare la dimensione dei buffer, ecc.

I commenti in `lv_conf.h` spiegano il significato di ciascuna impostazione. Si deve impostare almeno `LV_COLOR_DEPTH` in base alla profondità di colore del display. Nota: gli esempi e le demo devono essere esplicitamente abilitati in `lv_conf.h` se sono necessarie.

TODO: Aggiungere tutto ciò che riguarda il file `lv_conf.h` e il suo contenuto.

Istanze Multiple di LVGL

È possibile eseguire più istanze indipendenti di LVGL. Per abilitare la funzionalità multi-istanza, impostare `LV_GLOBAL_CUSTOM` in `lv_conf.h` e fornire una funzione personalizzata a `lv_global_default()` utilizzando `_thread` o `pthread_key_t`. Questo consentirà l'esecuzione di più istanze di LVGL memorizzando le variabili globali di LVGL in TLS (Thread-Local Storage).

Ad esempio:

```
lv_global_t * lv_global_default(void)
{
    static __thread lv_global_t lv_global;
    return &lv_global;
}
```

Kconfig

LVGL può essere configurato anche tramite Kconfig. Per ora, questa funzionalità è disponibile solo tramite cmake. Internamente, utilizza la porta Python di Kconfig `kconfiglib` per poterlo utilizzare su diverse piattaforme. `kconfiglib` offre l'API Python e alcuni comandi CLI. Ecco un elenco di alcuni comandi utili:

- `menuconfig`: Apre un'interfaccia a menù della console per modificare i valori di configurazione.
- `guiconfig` (richiede `tkinter`): Apre un'interfaccia grafica per modificare i valori di configurazione.
- `savedefconfig`: Salva il file `.config` corrente come `defconfig`, elencando solo i valori non di default.
- `alldefconfig`: Crea un file `.config` con tutti i valori di default.
- `genconfig`: Genera un header C dalla configurazione, seguendo il formato `autoconf.h`.

Prerequisiti

Installare i prerequisiti usando `scripts/install_prerequisites.sh/bat`.

Creazione della configurazione (.config)

A questo punto, il comando `menuconfig` dovrebbe essere disponibile:

```
cd <lvgl_repo>
menuconfig
```

Apportare le modifiche alla configurazione e uscire premendo `Esc` o `Q`, quindi salvare la configurazione. Il file `.config` è stato creato e contiene l'elenco dei valori di configurazione.

Configurazione con cmake

Una volta creato il file `.config`, eseguire `cmake` con il flag `-DLV_USE_KCONFIG=ON`:

```
cd <lvgl_repo>
cmake -B build -DLV_USE_KCONFIG=ON
cmake --build build
```

Per utilizzare un file `defconfig`, è possibile utilizzare il flag `-DLV_DEFCONFIG_PATH=<path_to_defconfig>`:

```
cd <lvgl_repo>
cmake -B build -DLV_USE_KCONFIG=ON -DLV_DEFCONFIG_PATH=<path_to_defconfig>
cmake --build build
```

Alcuni file `defconfig` sono disponibili nella cartella `configs/defconfigs`.

Salvataggio di un defconfig

È possibile salvare un `defconfig` utilizzando il comando `savedefconfig`:

```
cd <lvgl_repo>
menuconfig # make your changes to the default config
savedefconfig
cp defconfig configs/defconfigs/my_custom_defconfig # save it where you want
# Then use it to build LVGL
cmake -B build -DLV_USE_KCONFIG=ON -DLV_DEFCONFIG_PATH=configs/defconfigs/my_custom_defconfig
cmake --build build
```

4.1.4 Collegamento di LVGL all'Hardware

Inizializzazione di LVGL

Dopo aver:

- *acquisito LVGL*,
- aggiunto i file LVGL appropriati al progetto e
- *creato un file lv_conf.h* per il progetto,

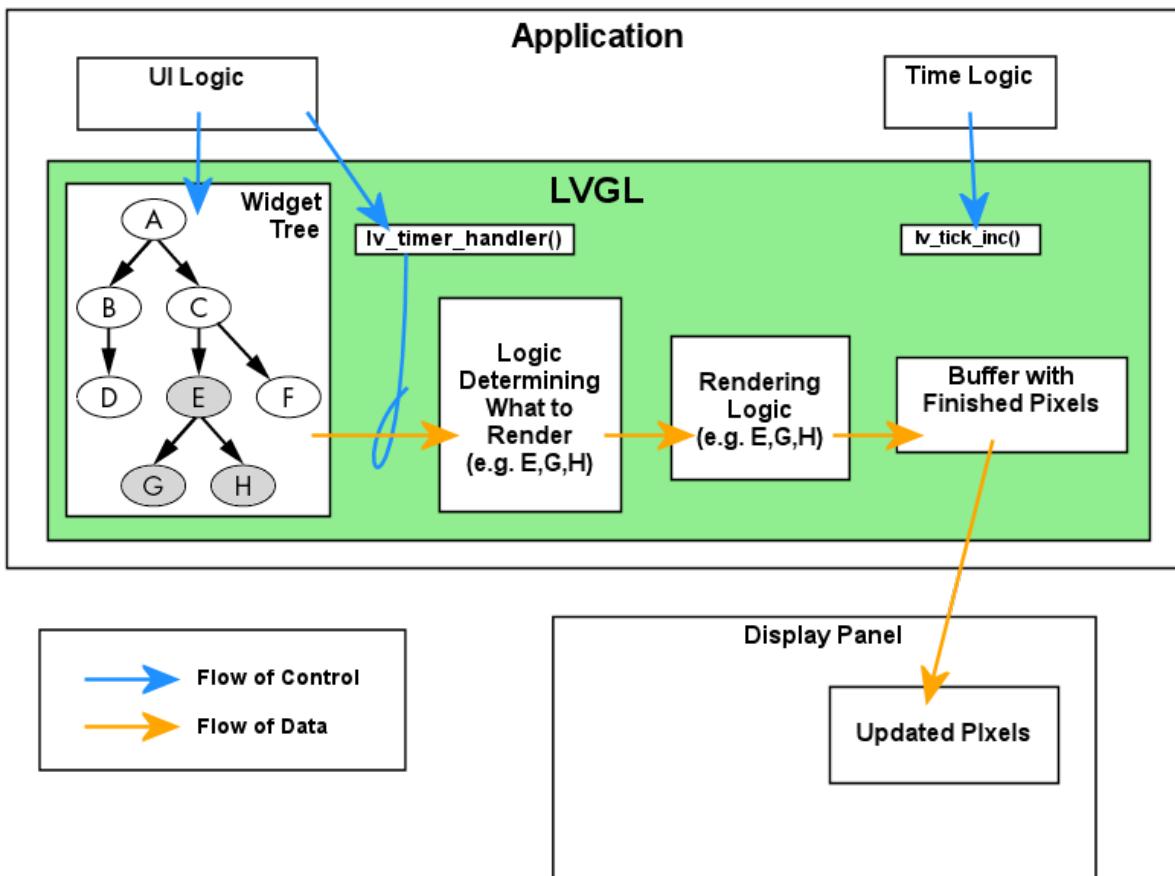
saranno necessari alcuni passaggi aggiuntivi per avviare il progetto con LVGL.

1. Inizializzare LVGL una sola volta all'inizio dell'esecuzione del sistema chiamando `lv_init()`. Questa operazione deve essere eseguita prima di qualsiasi altra chiamata a LVGL.
2. Inizializzare i driver.
3. Collegare l'*Interfaccia ai Tick*.
4. Collegare l'*Interfaccia al Display*.
5. Collegare l'*Interfaccia al Device di Input*.
6. Gestire le attività LVGL relative al tempo chiamando `lv_timer_handler()` ogni pochi millisecondi per gestire i timer LVGL. Consultare *Gestione del Timer* per diversi modi per farlo.
7. Facoltativamente, si imposta un tema con `lv_display_set_theme()`.

8. Dopodiché `#include "lvgl/lvgl.h"` nei sorgenti ovunque ci sia bisogno di utilizzare le funzioni LVGL.

Interfaccia ai Tick

LVGL ha bisogno di conoscere l'ora esatta (ovvero il tempo trascorso in millisecondi) per tutte le attività in cui il tempo è un fattore determinante: aggiornamento dei display, lettura dell'input dell'utente, attivazione di eventi, animazioni, ecc.



Esistono due modi per fornire queste informazioni a LVGL:

1. Fornire a LVGL una funzione di callback per recuperare i millisecondi di sistema trascorsi chiamando `lv_tick_set_cb(my_get_milliseconds)`. `my_get_milliseconds()` deve restituire il numero di millisecondi trascorsi dall'avvio del sistema. Molte piattaforme dispongono di funzioni native utilizzabili così come sono. Ad esempio:
 - SDL: `lv_tick_set_cb(SDL_GetTicks);`
 - Arduino: `lv_tick_set_cb(my_tick_get_cb);`, dove `my_tick_get_cb` è: `static uint32_t my_tick_get_cb(void) { return millis(); }`
 - FreeRTOS: `lv_tick_set_cb(xTaskGetTickCount);`
 - STM32: `lv_tick_set_cb(HAL_GetTick);`
 - ESP32: `lv_tick_set_cb(my_tick_get_cb);`, dove `my_tick_get_cb` è un wrapper per `esp_timer_get_time() /1000;`
2. Chiamare periodicamente `lv_tick_inc(x)`, dove `x` rappresenta i millisecondi trascorsi dall'ultima chiamata. Se `lv_tick_inc()` viene chiamato da un ISR, dovrebbe provenire da un interrupt ad alta priorità o da un interrupt che non può essere perso quando il sistema è sotto carico elevato.

Nota

`lv_tick_inc()` è solo una delle due funzioni LVGL che possono essere chiamate da un interrupt se la scrittura su un valore `uint32_t` è atomica sulla piattaforma. Vedere di seguito la sezione *Considerazioni sui Thread* per saperne di più.

In entrambi i casi, la scrittura del valore Tick `uint32_t` dev'essere *atomica*, il che è solitamente il caso di una piattaforma a 32 o 64 bit. Se si utilizza un sistema a 16 bit (per cui l'aggiornamento del valore Tick non è atomico) e la piattaforma utilizza il set di istruzioni Harvard, è possibile impostare una funzione come questa come callback passata a `lv_tick_set_cb(my_get_milliseconds)`:

```
/** @brief Safe read from 'elapsed_power_on_time_in_ms'
 */
uint32_t my_get_milliseconds()
{
    register uint32_t u32result;
    /* Disable priority 1-6 interrupts for 2 Fcys. */
    __builtin_disi(2);
    u32result = elapsed_power_on_time_in_ms; /* Cost: 2 Fcys */
    /* Generally looks like this in assembly:
     *   mov    elapsed_power_on_time_in_ms, W0
     *   mov    0x7898, W1
     * requiring exactly 2 clock cycles.
     * Now value is copied to register pair W0:W1
     * where it can be written to any destination. */
    return u32result;
}
```

Affidabilità

L'avanzamento del valore di tick dovrebbe essere eseguito in modo tale che la sua temporizzazione sia affidabile e non dipenda da nulla che consumi una quantità di tempo sconosciuta. Un esempio di cosa *non* fare: questo può "sembrare" funzionare, ma la temporizzazione di LVGL sarà errata perché il tempo di esecuzione di `lv_timer_handler()` varia da chiamata a chiamata e quindi il ritardo che introduce non può essere noto.

```
// Bad idea
lv_timer_handler();
lv_tick_inc(5);
my_delay_ms(5);
```

Interfaccia al Display

LVGL deve essere dotato di informazioni su ciascun pannello di visualizzazione che si desidera utilizzare. In particolare:

- il formato e la dimensione dei pixel (*Creazione di un Display*),
- dove renderizzare i pixel (*Buffer di Disegno*), e
- come inviargli tali pixel renderizzati (*Callback Flush*).

Consultare i rispettivi link per informazioni su come fornire queste informazioni a LVGL.

Interfaccia al Device di Input

LVGL deve sapere come ricevere input da tutti i dispositivi di input utente che verranno utilizzati nel progetto. LVGL supporta un'ampia varietà di dispositivi di input utente:

- touch-screen,
- touch-pad,
- mouse,
- crown [rotellina],
- encoder,
- keypad,
- keyboard,

- ecc.

Consultare [Dispositivi di input](#) per vedere come fare.

API

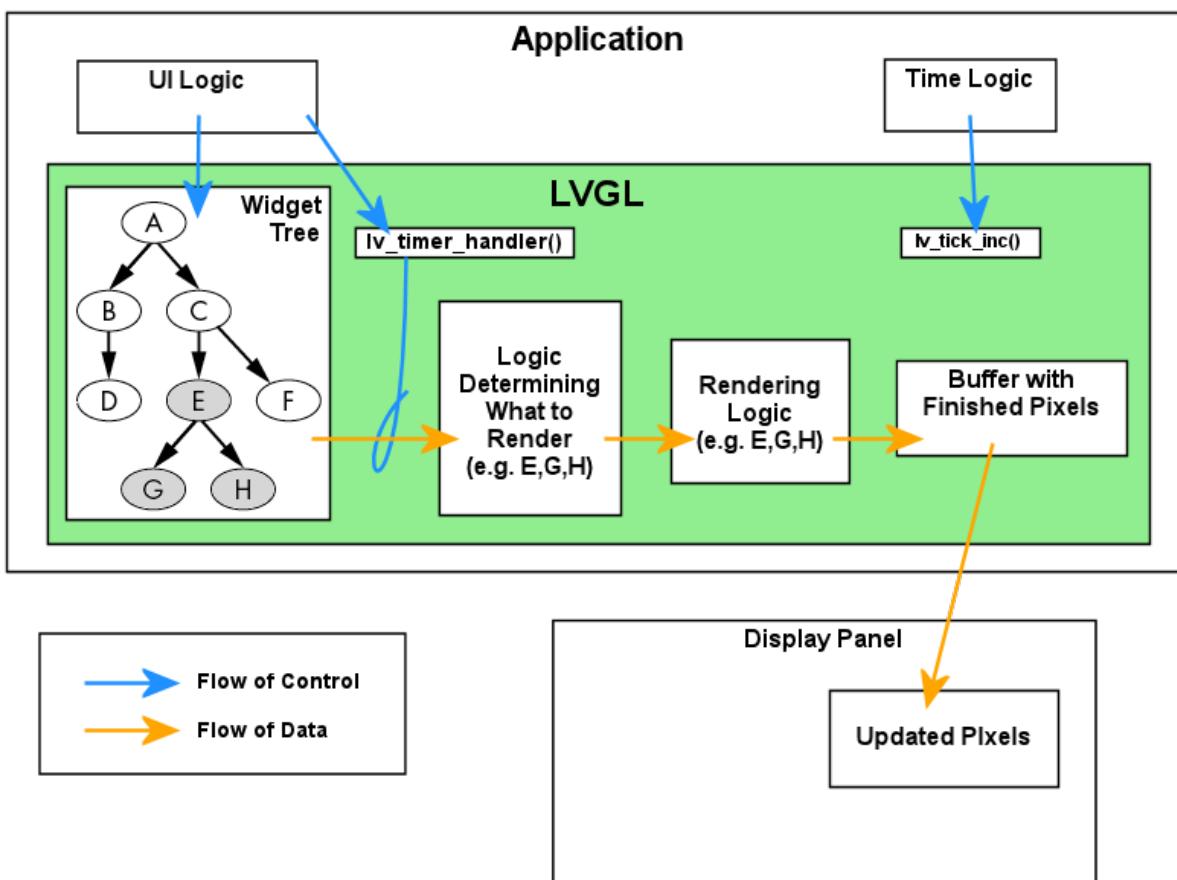
[lv_tick.h](#)

[lv_tick_private.h](#)

4.1.5 Gestione del Timer

Per gestire i timer di LVGL è necessario chiamare periodicamente `lv_timer_handler()` in uno dei seguenti modi:

- `while(1)` della funzione `main()`, o
- periodicamente, un task del sistema operativo. (Vedere [LVGL e i Thread](#).)



Esempio:

```
while(1) {
    uint32_t time_till_next = lv_timer_handler();
    if(time_till_next == LV_NO_TIMER_READY) time_till_next = LV_DEF_REFR_PERIOD; /*handle LV_NO_TIMER_READY. Another option is to 'sleep' for longer*/
    my_delay_ms(time_till_next);
}
```

Per utilizzare `lv_timer_handler()` in un super-loop, è disponibile la funzione helper `lv_timer_handler_run_in_period()` per semplificare l'impostazione di LVGL con la gestione del tempo:

```
while(1) {
    ...
    lv_timer_handler_run_in_period(5); /* run lv_timer_handler() every 5ms */
    ...
}
```

Oppure utilizzare il tempo di sleep calcolato automaticamente da LVGL:

```
while(1) {
...
lv_timer_periodic_handler();
...
}
```

In un ambiente di sistema operativo, è possibile utilizzarlo insieme a **delay** o a **sleep** fornito dal sistema operativo per rilasciare la CPU quando possibile:

```
while (1) {
    uint32_t time_till_next = lv_timer_handler();
    if(time_till_next == LV_NO_TIMER_READY) time_till_next = LV_DEF_REFR_PERIOD; /*handle LV_NO_TIMER_READY. Another option is to 'sleep' for longer*/
    os_delay_ms(time_till_next); /* delay to avoid unnecessary polling */
}
```

Leggere la sezione *Timer (lv_timer)* per saperne di più sui timer.

Quando Nessun Timer è Pronto

`lv_timer_handler()` restituirà `LV_NO_TIMER_READY` (`UINT32_MAX`) se non ci sono timer in esecuzione. Questo può accadere se non ci sono "indev" [input device] o se sono disabilitati con `lv_indev_enable()`, animazioni in esecuzione o timer creati dall'utente in esecuzione. `lv_timer_handler()` continuerà a restituire `LV_NO_TIMER_READY` finché non ci sarà un timer in esecuzione. I timer del display rimarranno in pausa quando non c'è motivo di aggiornarli. `lv_timer_handler()` dovrebbe essere chiamata dopo che qualcosa è stato creato, eliminato o modificato, in modo che venga eseguito un aggiornamento se necessario. In pratica, questo significa attendere senza timeout un evento esterno. Dopo che gli eventi esterni sono stati ricevuti e gestiti, `lv_timer_handler()` dovrebbe essere chiamato nuovamente.

```
while (1) {
    uint32_t time_till_next = lv_timer_handler();
    int timeout;

    /* Wait forever for events upon LV_NO_TIMER_READY, because there
     * is no reason to call lv_timer_handler sooner. */
    if(time_till_next == LV_NO_TIMER_READY) timeout = -1; /*infinite timeout*/

    /*Otherwise, wait for events at least until the timeout expires.*/
    else timeout = time_till_next;

    if(poll(..., timeout)) {
        /*Handle events before calling 'lv_timer_handler' again.*/
    }
}
```

Se non c'è una sorgente di eventi esterni, si può scegliere di uscire dal ciclo o semplicemente ritardare a lungo.

Se un altro *thread* sta chiamando funzioni LVGL, si potrebbe voler chiamare nuovamente `lv_timer_handler()` molto presto per gestire gli effetti di quegli altri thread.

API

lv_timer.h

4.1.6 Considerazioni sui Thread

Definizioni

Thread

Nel contesto di questo documento, un *thread* è una qualsiasi sequenza di istruzioni della CPU. Nelle implementazioni "bare-metal" (ovvero senza sistema operativo), i *thread* includono:

- il *thread* principale che esegue un ciclo `while(1)` che esegue il sistema e
- Le "interrupt service routine (ISR)".

Quando vengono eseguiti su un sistema operativo, i *thread* includono:

- ogni task (o processo),
- ISR, e
- i sistemi operativi avanzati possono avere più "thread in esecuzione" all'interno di un processo.

Operazione Atomica

Se l'operazione X è atomica, significa che qualsiasi thread che la osserva la vedrà come non ancora iniziata o come completata, e non in uno stato parzialmente completato.

Se altri thread possono vedere l'operazione in uno stato parzialmente eseguito o interferire con essa, allora l'operazione X non è atomica.

Se un'operazione atomica può fallire, la sua implementazione deve riportare la risorsa allo stato precedente all'avvio dell'operazione. Agli altri thread, deve apparire come se l'operazione non fosse ancora iniziata.

Dati Atomici

Un dato (ovvero il contenuto di una variabile o di una struttura dati) è atomico se qualsiasi thread che lo osserva lo vedrà sempre in uno stato coerente, come se le operazioni su di esso non fossero ancora iniziate o fossero state completate con successo, e non in uno stato parzialmente modificato o altrimenti incoerente.

Quando la lettura o la scrittura di un valore viene avviata e completata con una sola istruzione CPU, è automaticamente atomica, poiché non può mai essere vista in uno stato incoerente (parzialmente modificato), nemmeno a causa di un'interruzione o di un'eccezione CPU. Con tali valori, i programmati non richiedono alcuna protezione speciale per garantire che tutti i thread li vedano in uno stato coerente.

LVGL e i Thread

LVGL non è thread-safe.

Ciò significa che è responsabilità del programmatore assicurarsi che nessuna funzione LVGL venga chiamata mentre è in corso un'altra chiamata LVGL in un altro thread. Questo include le chiamate a `lv_timer_handler()`.

i Nota

Supponendo che quanto sopra sia il caso, è sicuro chiamare le funzioni LVGL nelle

- *callback di eventi* e nelle
- *callback del timer*

perché il thread che gestisce entrambi è il thread che chiama `lv_timer_handler()`.

Motivo:

LVGL gestisce molte strutture dati complesse, che sono "risorse di sistema" che devono essere protette dall'essere "viste" da altri thread in uno stato incoerente. Un'alta percentuale di funzioni LVGL (funzioni che iniziano con `lv_`) leggono o modificano tali strutture dati. Quelle che le modificano pongono i dati in uno stato incoerente durante l'esecuzione della chiamata (perché tali modifiche sono sequenze multi-step), ma li riportano a uno stato coerente prima che tali funzioni terminino. Per questo motivo, l'esecuzione di ciascuna funzione LVGL deve essere completata prima che venga avviata qualsiasi altra funzione LVGL.

i Eccezioni a quanto Sopra

Queste due funzioni LVGL possono essere chiamate da qualsiasi thread:

- `lv_tick_inc()` (se la scrittura su `uint32_t` è atomica sulla piattaforma; vedere *Interfaccia ai Tick* per maggiori informazioni) e
- `lv_display_flush_ready()` (*Callback Flush* per maggiori informazioni)

Il motivo per cui questo è accettabile è che i dati LVGL da loro modificati sono *atomici*.

Se un interrupt DEVE trasmettere informazioni a una parte dell'applicazione che chiama funzioni LVGL, imposta un flag o un altro valore atomico che il thread che chiama LVGL (o un *Timer LVGL* che si crea) possa leggere ed eseguire azioni.

Se si sta utilizzando un sistema operativo, ci sono alcune altre opzioni. Vedere di seguito.

Garantire che gli Aggiornamenti Temporali Siano Atomici

Affinché i task LVGL relativi al tempo siano affidabili, gli aggiornamenti temporali tramite l'interfaccia Tick devono essere affidabili e il valore di Tick deve apparire *atomico* a LVGL. [Interfaccia ai Tick](#) per i dettagli.

Task

In un sistema operativo, è comune avere molti thread di esecuzione ("task" in alcuni sistemi operativi) che eseguono servizi per l'applicazione. In alcuni casi, tali thread possono acquisire dati che dovrebbero essere visualizzati (o altrimenti riflessi) nell'interfaccia utente, e ciò richiede chiamate LVGL per visualizzare tali dati (o modifiche).

Tuttavia, rimane responsabilità del programmatore assicurarsi che nessuna funzione LVGL venga chiamata mentre è in corso un'altra chiamata LVGL.

Come si fa?

Metodo 1: Utilizzare un Gateway Thread

Un "Gateway Thread" (o "Gateway Task" in alcuni sistemi operativi) è un thread (task) che il progettista del sistema designa per gestire *esclusivamente* una risorsa di sistema. Un esempio è la gestione di un chip remoto, come una EEPROM o un altro dispositivo, che deve sempre essere portato in uno stato coerente prima dell'avvio di qualcosa di nuovo. Un altro esempio è la gestione di più dispositivi su un bus I2C (o qualsiasi bus dati). In questo caso, il bus I2C è la "risorsa gestita in modo esclusivo" e avere un solo thread che lo gestisce garantisce che ogni azione avviata possa essere completata prima che venga avviata un'altra azione con essa.

Le strutture dati di LVGL sono una risorsa di sistema che richiede tale protezione.

Utilizzando questo metodo, la creazione, la modifica e l'eliminazione di tutti i widget e delle altre risorse LVGL (ovvero tutte le chiamate di funzione LVGL, escluse le *eccezioni* menzionate in precedenza) vengono chiamate da quel thread. Ciò significa che il thread è anche l'UNICO chiamante di `lv_timer_handler()`. (Vedere [Panoramica sull'Integrazione](#) per ulteriori informazioni).

Questo garantisce che le strutture dati di LVGL "appaiano" *atomiche* (tutti i thread che utilizzano questi dati li "vedono" in uno stato coerente) perché nessun altro thread "visualizza" tali strutture dati. Questo è imposto dalla disciplina del programmatore che garantisce che il *Gateway Thread* sia l'unico thread a effettuare chiamate LVGL (escluse le *exceptions* menzionate sopra).

Se i *dati atomici* relativi all'interfaccia utente vengono aggiornati in un altro thread (ad esempio da un altro task o in un interrupt), il thread che chiama le funzioni LVGL può leggere tali dati direttamente senza preoccuparsi che siano in uno stato incoerente. (Per evitare un inutile sovraccarico della CPU, è possibile fornire un meccanismo [ad esempio, un flag generato dal thread di aggiornamento] in modo che l'interfaccia utente venga aggiornata solo quando ciò comporterà una modifica visibile all'utente finale).

Se i *dati non atomici* relativi all'interfaccia utente vengono aggiornati in un altro thread (ad esempio, da un altro task o durante un'interruzione), un modo alternativo (e sicuro) per trasmettere tali dati al thread che chiama le funzioni LVGL consiste nel passare una copia privata di tali dati a tale thread tramite una coda o un altro meccanismo del sistema operativo che protegga tali dati dalla visualizzazione in uno stato incoerente.

L'utilizzo di un *Gateway Thread* evita il sovraccarico di CPU (e di codice) derivante dall'utilizzo di un MUTEX per proteggere le strutture dati LVGL.

Metodo 2: Utilizzare un MUTEX

MUTEX sta per "MUTually EXclusive" (MUtualmente esclusivo) ed è una primitiva di sincronizzazione che protegge lo stato di una risorsa di sistema da modifiche o accessi simultanei da parte di più thread di esecuzione. In altre parole, fa sì che i dati così protetti "appaiano" atomici (tutti i thread che utilizzano questi dati li "vedono" in uno stato coerente). La maggior parte dei sistemi operativi fornisce MUTEX.

Il progettista del sistema assegna un singolo MUTEX per produrre una singola risorsa di sistema. Una volta assegnato, il MUTEX esegue tale protezione da parte dei programmatori:

1. acquisendo il MUTEX (ovvero bloccandolo) prima di accedere o modificare tale risorsa, e
2. rilasciando il MUTEX (ovvero sbloccandolo) al termine dell'accesso o della modifica.

Le buone pratiche di progettazione per l'utilizzo di un MUTEX sottolineano che il periodo di tempo in cui il MUTEX è bloccato deve essere il più breve possibile.

Se un thread tenta di acquisire (lock) il MUTEX mentre un altro thread ne è "proprietario", quel thread attende che l'altro thread lo rilasci (unlock) prima di poter continuare l'esecuzione.

Per essere chiari: questo deve essere fatto *sia* dai thread che LEGGONO da quella risorsa, sia dai thread che MODIFICANO quella risorsa.

Se un MUTEX viene utilizzato per proteggere le strutture dati LVGL, ciò significa *ogni* chiamata di funzione LVGL (o gruppo di chiamate di funzione) deve essere preceduta da #1 e seguita da #2, incluse le chiamate a `lv_timer_handler()`.

Nota

Se il sistema operativo è integrato con LVGL (la macro `LV_USE_OS` ha un valore diverso da `LV_OS_NONE` in `lv_conf.h`) si possono usare `lv_lock()` e `lv_unlock()` per eseguire i punti #1 e #2.

In questo caso, `lv_timer_handler()` chiama internamente `lv_lock()` e `lv_unlock()`, quindi non si devono includere le chiamate a `lv_timer_handler()` tra di esse.

Se il sistema operativo NON è integrato con LVGL, queste chiamate o terminano immediatamente senza alcun effetto, oppure vengono ottimizzate dal linker.

Per abilitare `lv_lock()` e `lv_unlock()`, si imposta `LV_USE_OS` su un valore diverso da `LV_OS_NONE`.

Questo pseudocodice illustra il concetto di utilizzo di un MUTEX:

```
void lvgl_thread(void)
{
    while(1) {
        uint32_t time_till_next;
        time_till_next = lv_timer_handler(); /* lv_lock/lv_unlock is called internally */
        if(time_till_next == LV_NO_TIMER_READY) time_till_next = LV_DEF_REFR_PERIOD; /*try again soon because the other thread can make the timer ready*/
        thread_sleep(time_till_next); /* sleep for a while */
    }
}

void other_thread(void)
{
    /* You must always hold (lock) the MUTEX while calling LVGL functions. */
    lv_lock();
    lv_obj_t *img = lv_image_create(lv_screen_active());
    lv_unlock();

    while(1) {
        lv_lock();
        /* Change to next image. */
        lv_image_set_src(img, next_image);
        lv_unlock();
        thread_sleep(2000);
    }
}
```

Gestione della Modalità Sleep

L'MCU può entrare in modalità sleep quando non riceve input dall'utente per un certo periodo di tempo. In questo caso, il `while(1)` principale potrebbe apparire così:

```
while(1) {
    /* Normal operation (no sleep) in < 1 sec inactivity */
    if(lv_display_get_inactive_time(NULL) < 1000) {
        lv_timer_handler();
    }
    /* Sleep after 1 sec inactivity */
    else {
        timer_stop(); /* Stop the timer where lv_tick_inc() is called */
        sleep(); /* Sleep the MCU */
    }
    my_delay_ms(5);
}
```

Si devono anche aggiungere le seguenti righe alla funzione di lettura del dispositivo di input per segnalare l'avvenuta riattivazione (pressione, tocco, clic, ecc.):

```
lv_tick_inc(LV_DEF_REFR_PERIOD); /* Force task execution on wake-up */
timer_start(); /* Restart timer where lv_tick_inc() is called */
lv_timer_handler(); /* Call 'lv_timer_handler()' manually to process the wake-up event */
```

Oltre a `lv_display_get_inactive_time()` si può controllare `lv_anim_count_running()` per verificare se tutte le animazioni sono terminate.

4.1.7 Altre Piattaforme e Tool

Consultare [Integrazione](#) per vedere come utilizzare LVGL su diverse piattaforme. Lì si troveranno molte descrizioni specifiche per ogni piattaforma, ad esempio per ESP32, Arduino, NXP, RT-Thread, NuttX, ecc.

4.2 Esecuzione su un PC

Si può provare LVGL **utilizzando solo il PC** (ovvero senza alcuna scheda di sviluppo).

Sono disponibili molti progetti pronti all'uso per iniziare facilmente su Windows, Linux, macOS e tramite browser web. Anche lo sviluppo di un BIOS UEFI è un'opzione.

4.2.1 Linux

Panoramica

LVGL include numerosi driver e supporto integrati per Linux:

- Driver di visualizzazione Wayland, DRM, SDL e fbdev
- Integrazione con pthread
- Supporto per [CMake](#)

Sono disponibili diversi repository per aiutare a iniziare facilmente. Basta seguire il README dei repository elencati di seguito.

Progetti Pronti-All'uso

VSCode

LVGL è disponibile per [VSCode](#), un noto editor di codice multipiattaforma.

Utilizza [SDL](#) per aprire una finestra, visualizzare il contenuto renderizzato e gestire mouse e tastiera.

Un progetto LVGL pronto all'uso per VSCode è disponibile all'indirizzo https://github.com/lvgl/lv_port_pc_vscode.

Il progetto utilizza [CMake](#) per compilare LVGL, quindi può essere utilizzato anche come esempio per CMake.

Questo repository include anche l'integrazione con FreeRTOS, semplificando il test dell'interfaccia utente con un sistema operativo embedded.

Porting Linux Generico

Il progetto [lv_port_linux](#) è in genere pensato per l'uso su hardware embedded, ma funziona perfettamente anche su PC.

Supporta tutti i driver LVGL integrati in un sistema di build [CMake](#) preconfigurato, facilitando l'integrazione in qualsiasi piattaforma o ambiente.

Eclipse

Il progetto per [Eclipse CDT](#) utilizza anche [SDL](#). È disponibile all'indirizzo https://github.com/lvgl/lv_port_pc_eclipse.

Il progetto utilizza CMake per compilare LVGL, quindi può essere utilizzato anche come esempio di [CMake](#).

4.2.2 Windows

Panoramica

LVGL include driver integrati per Windows, che includono:

- Supporto per display, mouse e tastiera

- Driver del sistema operativo

Utilizzando questi o altri driver, sono disponibili diversi repository per iniziare facilmente. Basta seguire il README dei repository elencati di seguito.

Progetti Pronti-all'uso

VSCode

LVGL è disponibile per VSCode, un noto editor di codice multipiattaforma.

Utilizza [SDL](#) per aprire una finestra, visualizzare il contenuto renderizzato e gestire mouse e tastiera.

Un progetto LVGL pronto all'uso per VSCode è disponibile all'indirizzo https://github.com/lvgl/lv_port_pc_vscode.

Il progetto utilizza [CMake](#) per compilare LVGL, quindi può essere utilizzato anche come esempio per CMake.

Questo repository include anche l'integrazione FreeRTOS integrata, semplificando il test dell'interfaccia utente con un sistema operativo embedded.

Visual Studio

Visual Studio è un IDE completo di Microsoft che contiene tutti gli strumenti necessari per lo sviluppo pronti all'uso.

Per utilizzare LVGL in Visual Studio, visitare il repository [lvgl/lv_port_pc_visual_studio](https://github.com/lvgl/lv_port_pc_visual_studio).

Utilizza i driver Windows integrati di LVGL per aprire una finestra in cui è possibile visualizzare l'interfaccia utente renderizzata da LVGL e utilizzare il mouse come sostituto dell'input touch.

Eclipse

Il progetto per Eclipse CDT utilizza anche [SDL](#). È disponibile all'indirizzo https://github.com/lvgl/lv_port_pc_eclipse.

Il progetto utilizza [CMake](#) per compilare LVGL e funge da esempio di CMake.

Driver Windows

È anche facile portare LVGL su qualsiasi IDE o toolchain Windows utilizzando il driver [SDL](#) integrato o il driver Windows.

Per saperne di più sul driver [SDL](#), visitare [Driver SDL](#).

I dettagli del driver Windows sono forniti di seguito.

Il [driver](#) di display/input di [Windows](#) offre supporto per la simulazione del display LVGL e degli input da tastiera/mouse in una finestra Win32 di Windows.

Modalità

Lo scopo principale di questo driver è testare/debuggare applicazioni LVGL in una finestra di simulazione di [Windows](#) tramite la **simulator mode**, o sviluppare applicazioni desktop standard di [Windows](#) con LVGL tramite la **application mode**.

Modalità simulatore e modalità applicazione:

- Supporta l'integrazione di puntatori, tastiere e dispositivi encoder LVGL
- Supporta l'input touch di Windows
- Supporta l'integrazione del metodo di input di Windows
- Supporta il riconoscimento DPI per monitor (sia V1 che V2)
- Offre interoperabilità basata su HWND per altre infrastrutture dell'interfaccia utente di Windows

Tuttavia, **Simulator Mode**:

- è progettata per simulare il rendering LVGL su display di tipo hardware,
- mantiene costante la risoluzione del display LVGL per simulare i layout dell'interfaccia utente di produzione e
- utilizza il ridimensionamento DPI di Windows per estendere i contenuti,

mentre **Application Mode**:

- è progettata per lo sviluppo di applicazioni desktop Windows native,
- supporta il ridimensionamento dinamico delle finestre con le relative modifiche alla risoluzione del display LVGL,
- aggiorna i DPI del display LVGL quando cambiano i DPI di Windows,
- utilizza le dimensioni della finestra (non l'area client) quando si imposta la risoluzione tramite `lv_windows_create_display()`, e
- richiede alle app di gestire correttamente le modifiche alla risoluzione del display.

La modalità può essere selezionata chiamando `lv_windows_create_display()`.

Prerequisiti

Il requisito minimo del sistema operativo Windows per questo driver è Windows Vista RTM.

Se si utilizzano librerie shim API Windows come [YY-Thunks](#), la versione minima del sistema operativo testata è Windows XP RTM.

A causa dell'utilizzo delle API GDI di Windows, la versione minima teorica potrebbe essere Windows 2000 RTM.

Utilizzo

Abilitare il supporto del driver Windows in `lv_conf.h`, tramite la definizione del compilatore CMake o KConfig:

```
#define LV_USE_WINDOWS 1
```

Una volta abilitato, il driver Windows può essere utilizzato in questo modo:

```
#include <Windows.h>
#include "lvgl/lvgl.h"
#include "lvgl/examples/lv_examples.h"
#include "lvgl/demos/lv_demos.h"

int main()
{
    lv_init();

    int32_t zoom_level = 100;
    bool allow_dpi_override = false;
    bool simulator_mode = false;
    lv_display_* display = lv_windows_create_display(
        L"LVGL Display Window",
        800, 480,
        zoom_level, allow_dpi_override, simulator_mode);

    if (!display) return -1;

    lv_lock();

    lv_indev_t* pointer_device = lv_windows_acquire_pointer_indev(display);
    if (!pointer_device) return -1;

    lv_indev_t* keypad_device = lv_windows_acquire_keypad_indev(display);
    if (!keypad_device) return -1;

    lv_indev_t* encoder_device = lv_windows_acquire_encoder_indev(display);
    if (!encoder_device) return -1;

    lv_demo_widgets();

    lv_unlock();

    while (1)
    {
        uint32_t time_till_next = lv_timer_handler();
        // handle LV_NO_TIMER_READY. Another option is to always sleep a few milliseconds
        if (time_till_next == LV_NO_TIMER_READY) time_till_next = LV_DEF_REFR_PERIOD;
        lv_sleep_ms(time_till_next);
    }

    return 0;
}
```

4.2.3 macOS

Panoramica

Sono disponibili progetti pronti all'uso per eseguire LVGL anche su macOS. Questi progetti utilizzano [Driver SDL](#).

Progetti Pronti-All'uso

VSCode

LVGL è disponibile per [VSCode](#), un noto editor di codice multipiattaforma.

Utilizza [SDL](#) per aprire una finestra, visualizzare il contenuto renderizzato e gestire mouse e tastiera.

Un progetto LVGL pronto all'uso per VSCode è disponibile all'indirizzo https://github.com/lvgl/lv_port_pc_vscode.

Il progetto utilizza [CMake](#) per compilare LVGL, quindi può essere utilizzato anche come esempio per CMake.

Questo repository include anche l'integrazione con FreeRTOS, semplificando il test dell'interfaccia utente anche con un sistema operativo embedded.

Eclipse

Il progetto per [Eclipse CDT](#) utilizza anche [SDL](#). È disponibile all'indirizzo https://github.com/lvgl/lv_port_pc_eclipse.

Il progetto utilizza CMake per compilare LVGL, quindi può essere utilizzato anche come esempio di [CMake](#).

4.2.4 Browser

Panoramica

Esistono diversi modi per visualizzare le interfacce utente LVGL in un browser o persino svilupparle direttamente in un ambiente browser:

- Compilare l'interfaccia utente in HTML utilizzando [Emscripten](#)
- Sviluppare in VSCode online utilizzando [GitHub Codespaces](#)
- Visualizzare l'anteprima delle interfacce utente nella [preview online di LVGL UI Editor](#)

Compilare le interfacce utente in HTML (Emscripten)

Emscripten può compilare file sorgenti C in un singolo output HTML che viene eseguito in un browser. Questo è un modo pratico per condividere le interfacce utente sviluppate con gli stakeholder, poiché possono testarle senza dover configurare un programma di sviluppo.

Invece di produrre un binario, Emscripten produce WebAssembly, supportato dai browser moderni.

Il progetto [lv_web_emscripten](#) fornisce una configurazione pronta all'uso per compilare qualsiasi interfaccia utente LVGL in un file HTML compatibile con il browser.

Utilizza un sistema di build specializzato, basato su [CMake](#): consultare il file README per i dettagli completi sulla configurazione.

Sviluppo in VSCode online (GitHub Codespaces)

GitHub Codespaces offre un ambiente VSCode ospitato sul cloud eseguito interamente nel browser.

LVGL fornisce un ambiente Codespace preconfigurato, avviabile con pochi clic. È particolarmente utile se:

- Si stanno riscontrando problemi con la configurazione di un ambiente di sviluppo locale
- Il proprio computer è lento o sottodimensionato
- Si sta lavorando da un dispositivo senza un IDE locale

Per iniziare:

1. Andare su <https://github.com/lvgl/lvgl>.
2. Cliccare sul pulsante verde **Code**.
3. Selezionare il tab **Codespaces**.
4. Cliccare su **Create Codespace on master**.
5. Attendere brevemente l'avvio dell'area di lavoro.
6. Aprire il terminale ed eseguire `./build_all.sh`.

Questo compilerà LVGL e "renderizzerà" l'interfaccia utente in una scheda VSCode. Si può continuare a scrivere e ricompilare semplicemente eseguendo nuovamente lo script.

Nota: La configurazione di Codespaces utilizza anche Emscripten per il rendering del browser.

Condivisione Online dell'Editor dell'Interfaccia Utente di LVGL

Prossimamente.

4.2.5 Driver SDL

Panoramica

SDL (Simple DirectMedia Layer) offre un modo multipiattaforma per gestire grafica, input e contenuti multimediali, rendendolo una scelta eccellente per l'esecuzione di applicazioni LVGL su PC.

I progetti [Eclipse](#), [VSCode](#) e [Generic Linux](#) di LVGL utilizzano SDL per aprire una finestra e gestire i dispositivi di input.

Poiché SDL è completamente multipiattaforma, è facile da integrare in qualsiasi ambiente.

Prerequisiti

Installare SDL in base alla piattaforma:

- Linux: `sudo apt install libsdl2-dev`
- macOS: `brew install sdl2`
- Windows: Consigliato: `vcpkg install sdl2` In alternativa, si possono scaricare i binari SDL2 precompilati dal [sito web di SDL](#) e seguire le relative istruzioni di installazione.

Configurazione

1. Librerie linkate richieste: `-lSDL2`
2. Abilitare il supporto per i driver SDL in `lv_conf.h`, tramite le definizioni del compilatore CMake o utilizzando KConfig.

```
#define LV_USE	SDL_ 1
```

Si è liberi di modificare le altre opzioni in base alle proprie esigenze, ma i valori di default sono solitamente sufficienti.

Utilizzo

```
#define SDL_MAIN_HANDLED /* To fix SDL's "undefined reference to WinMain" issue */
#include "lvgl/lvgl.h"

static lv_display_t *display;
static lv_indev_t *mouse;
static lv_indev_t *mouse_wheel;
static lv_indev_t *keyboard;

int main()
{
    /* Initialize LVGL */
    lv_init();

    display = lv_sdl_window_create(SDL_HOR_RES, SDL_VER_RES);
    mouse = lv_sdl_mouse_create();
    mouse_wheel = lv_sdl_mousewheel_create();
    keyboard = lv_sdl_keyboard_create();
```

(continues on next page)

(continua dalla pagina precedente)

```

/* Create widgets on the screen */
lv_demo_widgets();

while (1) {
    lv_timer_handler();
    lv_delay_ms(5);
}

return 0;
}

```

Quando si compila per un'architettura a 32 bit, aggiungere la seguente soluzione alternativa all'inizio di `main`:

```

// Workaround for SOL2 `~m32` crash
// https://bugs.launchpad.net/ubuntu/+source/libSDL2/+bug/1775067/comments/7
#ifndef WIN32
    setenv("DBUS_FATAL_WARNINGS", "0", 1);
#endif

```

4.2.6 UEFI

Panoramica

Il `driver` di visualizzazione/input **UEFI** offre supporto per l'utilizzo di LVGL per creare interfacce utente in ambienti UEFI.

Prerequisiti

Sono necessari i seguenti protocolli UEFI disponibili:

- `EFI_LOADED_IMAGE_PROTOCOL_GUID`: per il supporto del file system (utilizzato per determinare il file system utilizzato per caricare l'applicazione)
- `EFI_SIMPLE_FILE_SYSTEM_PROTOCOL_GUID`: per il supporto del file system
- `EFI_SIMPLE_TEXT_INPUT_EX_PROTOCOL_GUID`: per il supporto della tastiera
- `EFI_SIMPLE_POINTER_PROTOCOL_GUID`: per il supporto del mouse
- `EFI_ABSOLUTE_POINTER_PROTOCOL_GUID`: per il supporto touch
- `EFI_GRAPHICS_OUTPUT_PROTOCOL_GUID`: per il disegno
- `EFI_EDID_ACTIVE_PROTOCOL_GUID`: per il disegno (facoltativo)

Configurazione

1. Abilitare il supporto del driver UEFI in `lv_conf.h`:

```
#define LV_USE_UEFI 1
```

- Abilitare le funzioni core della memoria, che fungono da wrapper per `AllocatePool` e `FreePool` (utilizzando memoria di tipo `EfiBootServicesData`) se non si desidera utilizzare implementazioni personalizzate.

```
#define LV_UEFI_USE_MEMORY_SERVICES 1
```

3. Abilitare il supporto del file system per il volume da cui è stata caricata l'applicazione (per default, la lettera di unità E):

```
#define LV_USE_FS_UEFI 1
```

4. Includere le definizioni del tipo UEFI. Sono disponibili "include" predefinite per EDK2 e GNU-EFI:

```

#define LV_USE_UEFI_INCLUDE <lvgl/src/drivers/uefi/lv_uefi_edk2.h>
// or
#define LV_USE_UEFI_INCLUDE <lvgl/src/drivers/uefi/lv_uefi_gnu_efi.h>

```

Utilizzo

Un esempio minimo:

```
#include "lvgl/lvgl.h"
#include "lvgl/examples/lv_examples.h"
#include "lvgl/demos/lv_demos.h"

EFI_STATUS EFIAPI EfiMain(EFI_HANDLE ImageHandle, EFI_SYSTEM_TABLE *SystemTable)
{
    lv_uefi_init(ImageHandle, SystemTable);
    lv_init();

    if (!lv_is_initialized()) return EFI_NOT_READY;

    EFI_HANDLE handle = lv_uefi_display_get_active();
    if (!handle) handle = lv_uefi_display_get_any();
    if (!handle) {
        lv_deinit();
        return EFI_UNSUPPORTED;
    }

    lv_display_t *display = lv_uefi_display_create(handle);
    lv_display_set_default(display);

    lv_group_t *group = lv_group_create();
    lv_group_set_default(group);

    lv_obj_t *cursor = lv_image_create(lv_layer_top());
    lv_image_set_src(cursor, "E:cursor.png");

    lv_indev_t *indev;

    indev = lv_uefi_simple_text_input_indev_create();
    lv_indev_set_group(indev, group);
    lv_uefi_simple_text_input_indev_add_all(indev);

    indev = lv_uefi_simple_pointer_indev_create(NULL);
    lv_uefi_simple_pointer_indev_add_all(indev);
    lv_indev_set_cursor(indev, cursor);

    indev = lv_uefi_absolute_pointer_indev_create(NULL);
    lv_uefi_absolute_pointer_indev_add_all(indev);

    lv_demo_widgets();

    size_t counter = 0;
    while (counter < 10000) {
        counter++;
        gBS->Stall(1000);
        lv_tick_inc(1);
        lv_timer_handler();
    }
}

return EFI_SUCCESS;
}
```

4.3 Esecuzione su Embedded Linux

4.3.1 Panoramica

4.3.2 Panoramica su OpenGL

Introduzione

Panoramica

OpenGL (Open Graphics Library) è un'API grafica multipiattaforma con accelerazione hardware che fornisce un'interfaccia standardizzata per il rendering di grafica 2D e 3D. Sviluppata originariamente da Silicon Graphics nel 1992, OpenGL è diventata una delle API grafiche più ampiamente adottate su piattaforme desktop, mobili ed embedded.

Supporto OpenGL in LVGL

L'integrazione di OpenGL consiste in due componenti principali:

- OpenGL Display Drivers: Gestiscono l'output mostrando il contenuto renderizzato LVGL sul display, sulla finestra o sulla texture in ambienti basati su OpenGL
- OpenGL Draw Unit: Se abilitata, LVGL utilizza OpenGL per le operazioni di rendering con accelerazione hardware, incluso il caching delle texture per migliorare le prestazioni

Il supporto OpenGL affronta diversi casi d'uso chiave:

1. Ottimizzazione delle prestazioni: Sfrutta l'accelerazione GPU per il rendering.
2. Compatibilità Cross-Platform: L'ampio supporto di OpenGL consente l'utilizzo di interfacce utente LVGL su sistemi desktop, mobili ed embedded.

3. Integrazione con applicazioni esistenti: Integra elementi dell'interfaccia utente LVGL in giochi, strumenti di visualizzazione o altre applicazioni OpenGL.

Considerazioni sulle Prestazioni

L'implementazione OpenGL offre significativi vantaggi in termini di prestazioni nella maggior parte degli scenari:

- Prestazioni Ottimali: Le interfacce utente con contenuti statici o che cambiano raramente traggono i maggiori vantaggi dal caching delle texture.
- Prestazioni Moderate: I contenuti dinamici con pattern prevedibili subiscono comunque miglioramenti.
- Potenziale Overhead: I contenuti che cambiano continuamente (ad esempio, colori casuali a ogni fotogramma) possono avere prestazioni simili al rendering software.

Opzioni per il Driver OpenGL

LVGL fornisce tre driver OpenGL per soddisfare diverse esigenze applicative e piattaforme:

1. Driver GLFW

Il driver di visualizzazione/input GLFW offre supporto per la creazione di display LVGL e input da tastiera/mouse utilizzabili in un contesto OpenGL. Il driver GLFW fornisce la creazione automatica delle finestre e la gestione del contesto con una gestione completa degli input per uno sviluppo rapido su piattaforme simili a PC.

Per i dettagli completi sull'implementazione, vedere [Driver GLFW](#).

2. Driver EGL

Il driver di visualizzazione EGL offre supporto per la creazione di display LVGL utilizzando l'API EGL (Embedded-System Graphics Library), utilizzabile in un contesto OpenGL. Questo driver fornisce l'integrazione hardware di basso livello con accesso diretto al driver, supportando sia sistemi basati su DRM che implementazioni 'standalone' per piattaforme embedded.

Per i dettagli completi sull'implementazione, vedere [Driver EGL](#).

3. Driver Generico OpenGL

Il driver OpenGL generico offre supporto per la creazione di display LVGL come texture OpenGL integrabili in applicazioni OpenGL esistenti. Questo driver presuppone un contesto OpenGL esistente e genera texture con accelerazione hardware per l'integrazione in pipeline grafiche personalizzate.

Per i dettagli completi sull'implementazione, vedere [Driver OpenGL](#).

Supporto 3D/glTF

L'estensione glTF fornisce supporto per il caricamento e il rendering di modelli 3D utilizzando la specifica glTF 2.0 all'interno di applicazioni LVGL. Questa estensione richiede OpenGL ES 3.0 e offre funzionalità di rendering 3D complete, inclusi materiali PBR, animazioni e controlli interattivi della telecamera per la visualizzazione 3D embedded.

Per i dettagli completi sull'implementazione, vedere [glTF](#).

4.3.3 Supporto per OS

Buildroot

Buildroot è un set di Makefile e patch che semplifica e automatizza il processo di creazione di un ambiente Linux completo e avviabile per un sistema embedded, utilizzando al contempo la cross-compilazione per consentire la creazione per più piattaforme target su un singolo sistema di sviluppo basato su Linux.

Configurazione Rapida

È disponibile un repository Git che include tutto il necessario per testare la configurazione di Buildroot senza seguire la guida. È pensato per scopi di test e può essere utilizzato se si ha già familiarità con i concetti e l'architettura di Buildroot.

Ottenerne il repository

```
git clone --recurse-submodules https://github.com/lvgl/lv_buildroot.git
```

Creare l'immagine

```
./build.sh
```

A questo punto, si ha un'immagine che può essere flashata su una SD e utilizzata.

Ora compilare l'SDK e installarlo

```
cd output
make sdk
mkdir -p ~/sdk
tar -xzf images/aarch64-buildroot-linux-gnu_sdk-buildroot.tar.gz -C ~/sdk
```

L'SDK è installato. Compilare l'applicazione.

```
cd ../application/lv_benchmark
source ../setup-build-env.sh
cmake -B build -S .
make -j $(nproc) -C build
cd ../../
```

Installare il binario in rootfs_overlay

```
cp application/lv_benchmark/bin/lvgl-app resources/board/rootfs_overlay/usr/bin/
```

Rigenerare l'immagine con l'overlay rootfs aggiornato.

```
cd output
make
```

Andare su [Eseguire il flashing dell'immagine](#) per testare il sistema.

Immagine personalizzata RPi4

Questo capitolo offre una guida dettagliata per la creazione di un'immagine personalizzata per Raspberry Pi 4 (RPi4). Verranno introdotti i componenti e i concetti chiave di Buildroot. Il processo descritto qui può essere adattato per l'utilizzo con altre schede. Per funzionalità avanzate non trattate in questa guida, consultare il [Manuale di Buildroot](#)

Questa guida è stata realizzata su Ubuntu 22.04

Preparare l'area di lavoro

L'area di lavoro è strutturata per utilizzare Buildroot senza alcuna modifica. Eseguire i seguenti comandi:

```
mkdir rpi4_buildroot_lvgl && cd rpi4_buildroot_lvgl
mkdir resources && cd resources
mkdir board
mkdir configs
touch Config.in && touch external.desc && touch external.mk
echo "name: rpi4_lvgl" >> external.desc
echo "desc: rpi4_lvgl custom build" >> external.desc
cd ..
```

Ogni utilità di gestione cartelle verrà spiegata nel corso della guida.

Ottenerne Buildroot

Innanzitutto, secondo il suo [manuale](#), Buildroot richiede l'installazione di determinati pacchetti prima di avviare la compilazione. Installiamoli utilizzando il gestore pacchetti di Ubuntu.

```
sudo apt install sed make binutils gcc g++ bash patch gzip bzip2 perl tar \
    cpio python3 unzip rsync wget libncurses-dev
```

Ora che abbiamo soddisfatto tutti i prerequisiti, scarichiamo una versione stabile di Buildroot.

```
git clone --branch 2024.08 --depth 1 https://github.com/buildroot/buildroot.git
```

Creare un'immagine RPi4

Vediamo se esiste già una configurazione per RPi4 in Buildroot:

```
cd buildroot
make list-defconfigs | grep rasp
```

È disponibile una build per RPi4 a 62 bit: `raspberrypi4_64_defconfig`.

Si possono trovare tutte le configurazioni anche nel repository Buildroot `buildroot > configs`

Iniziare con questa configurazione come base, ma deve essere modificata per soddisfare i requisiti di lvgl. Quindi copiamo questa base e teniamola fuori da Buildroot.

```
# move to the project root
cp buildroot/configs/raspberrypi4_64_defconfig resources/configs/raspberrypi4_lvgl_defconfig
```

Creare un nuovo file `build.sh` nella radice del progetto con questo contenuto:

```
#!/bin/bash
SCRIPTPATH=$( cd "$( dirname "$0" )" >/dev/null 2>&1 ; pwd -P )
cd $SCRIPTPATH
mkdir -p output
cd buildroot
make defconfig O=../output BR2_DEFCONFIG=../resources/configs/raspberrypi4_lvgl_defconfig
```

Rendere questo file eseguibile.

```
chmod +x build.sh
./build.sh
```

L'esecuzione di questo script crea una cartella di output che contiene tutto ciò che il processo genererà:

- Risorse recuperate
- Toolchain
- Immagini

Questo aiuta a mantenere pulita la directory Buildroot. Quando lo script viene eseguito, viene creata una cartella (quasi) vuota denominata `output`, poiché il processo di build non è ancora iniziato.

Modifichiamo la configurazione con `menuconfig`.

```
cd output
make menuconfig
```

Modificare queste configurazioni:

- Nome host di sistema in `lvgl-buildroot` in `system_configuration > System hostname`.
- Password di root in `lvgl-buildroot` in `system_configuration > Root password`.
- Abilitare SSH in `target-packages > Networking applications > openssh`

Creare l'immagine

```
make
```

Cosa è successo?

Esploriamo il contenuto della cartella di output e cosa contiene.

build

Questa cartella contiene i file intermedi e gli artefatti della build per i vari pacchetti in fase di compilazione. Ogni pacchetto ha una propria sottodirectory all'interno della cartella `build`, dove avviene il processo di compilazione. Include file come file di configurazione, codice sorgente e file oggetto generati durante il processo di compilazione.

host

La cartella `host` contiene file e binari compilati per il sistema host anziché per il sistema target. Include strumenti e utilità necessari per compilare i pacchetti o per eseguire il sistema di build stesso. Può contenere compilatori, strumenti di compilazione e librerie necessari per supportare il processo di compilazione per il sistema target.

images

Questa directory contiene le immagini di output finali generate per il sistema target, come immagini del file system, immagini del kernel o immagini del bootloader. A seconda della configurazione, si potrebbero trovare file come `rootfs.tar`, `zImage`, `uImage`, o altri, pronti per essere distribuiti sull'hardware target.

target

La cartella `target` contiene i file specificamente destinati al sistema di destinazione. Questo include il file system di root e qualsiasi file aggiuntivo che verrà incluso nell'ambiente di destinazione. La struttura all'interno di questa cartella spesso imita la struttura delle directory di un sistema Linux standard, con directory come `bin`, `lib`, `etc`, `usr` e altre, che contengono i file binari, le librerie, i file di configurazione e altri componenti necessari per il corretto funzionamento del sistema target.

Eseguire il flashing dell'immagine

Inserire la scheda SD nel laptop e controllarne il punto di montaggio. In genere è etichettata come `sda` o `sdb`, ma si può usare il comando `lsblk` per confermare questa informazione.

Se è montata su `/dev/sda`, si esegue il seguente comando:

```
sudo dd if=images/sdcard.img of=/dev/sda
```

Come indicato in `images`, l'immagine di output si trova in `images` e si chiama `sdcard.img`.

Collegare un cavo Ethernet al RPi4 e assicurarsi che il laptop e l'RPi4 siano sulla stessa rete.

Si può usare `ifconfig` o `ip` a per trovare il proprio indirizzo IP. Poi, si usa `nmap` o qualsiasi altro strumento per scansionare la rete e trovare l'indirizzo IP del RPi4.

```
nmap -sn <YOUR_IP_ADDRESS>/24 | grep Lvgl
```

Tutto questo può essere fatto con l'UART invece che con SSH se non si vuol connettere l'RPi4 alla rete.

Applicazione LVGL

Questa sezione fornisce informazioni sui passaggi da seguire per ottenere un'applicazione personalizzata che utilizzi LVGL in esecuzione sulla scheda.

Aggiornare RootFS

A seconda dell'applicazione, potrebbe essere necessario aggiornare il `rootfs`. Prendiamo come esempio la compilazione di LVGL con DRM. Il sistema deve avere `libdrm` installato.

```
cd output
make menuconfig
```

Per cercare un pattern di stringhe nella configurazione, premere / seguito dal pattern desiderato. Ad esempio, cercare `libdrm`. Si dovrebbe trovare `BR2_PACKAGE_LIBDRM` impostato su `[=n]`. Il campo "Location" indica dove trovare questa opzione. Premendo il tasto numerico corrispondente (9-0), è possibile accedere direttamente all'opzione.

Nella finestra **Search Results**, la sezione **Depends on** elenca i pacchetti o le opzioni richiesti che devono essere abilitati (o disabilitati) per rendere configurabile il pacchetto target. Per qualsiasi pacchetto che si desidera aggiungere, queste dipendenze devono essere soddisfatte; in caso contrario, l'opzione rimarrà nascosta.

Una volta completata la configurazione del pacchetto, si compila l'ambiente per aggiungere i pacchetti (make nella cartella **output**).

Per verificare che la libreria sia stata installata, possiamo trovarla nella sysroot di destinazione:

```
find build/ -name "*libdrm*"
```

Si dovrebbero vedere la cartella "include" e i file .so.

Generare l'SDK e configurare l'ambiente

Generare un SDK utilizzabile per la cross-compilazione dell'applicazione per il target (RPi4).

make sdk

Viene generato un file .tar.gz in output/images. Questo è l'SDK!

Per utilizzarlo, estrarlo ovunque.

```
mkdir -p ~/sdk  
tar -xzf images/aarch64-buildroot-linux-gnu_sdk-buildroot.tar.gz -C ~/sdk  
cd ..
```

Per configurare l'ambiente

Lo script `setup-build-env.sh` si presenta così:

```
#!/bin/bash

SDK_PATH="$HOME/sdk/aarch64-buildroot-linux-gnu_sdk-buildroot"

export PATH="${SDK_PATH}/bin:${PATH}"

export SYSROOT="${SDK_PATH}/aarch64-buildroot-linux-gnu/sysroot"

export CROSS_COMPILE="aarch64-buildroot-linux-gnu-"

export CC="${CROSS_COMPILE}gcc"
export CXX="${CROSS_COMPILE}g++"
export LD="${CROSS_COMPILE}ld"
export AR="${CROSS_COMPILE}ar"
export AS="${CROSS_COMPILE}as"

export CFLAGS="--sysroot=${SYSROOT}"
export LDFLAGS="--sysroot=${SYSROOT}"
```

Build dell'applicazione

L'ambiente è ora configurato e siamo pronti per compilare un'applicazione utilizzando il repository `lv_benchmark` ispirato a `lv_port_linux`.

Tornare alla radice del progetto e clonare il repository:

```
git clone --recurse-submodules https://github.com/EDGEMTech/lv_benchmark.git
```

L'applicazione è configurata per essere eseguita su fbdev. È possibile mantenere la configurazione di default o modificarla in base alle proprie preferenze.

Compilare l'applicazione

```
cd lv_benchmark  
cmake -B build -S . -DCMAKE_C_COMPILER=${CROSS_COMPILE}gcc -DCMAKE_CXX_COMPILER=${CROSS_COMPILE}g++ -DCMAKE_SYSROOT=${SYSROOT} -DCMAKE_C_FLAGS="-sysroot=${SYSROOT}" -DCMAKE_CXX_FLAGS="-sysroot=${SYSROOT}"  
make -j $(nproc) -j build
```

Verificare che l'eseguibile di output sia stato compilato con la toolchain corretta:

```
file bin/lvgl-app  
cd /
```

L'output dovrebbe contenere queste informazioni:

- ARM aarch64
- interpreter /lib/ld-linux-aarch64.so.1

Impostare un overlay rootfs

In Buildroot, un overlay rootfs (o overlay del filesystem di root) è un meccanismo che consente di aggiungere file, directory e configurazioni personalizzate direttamente nel filesystem di root dell'immagine target durante il processo di build. È un modo per estendere o modificare il contenuto del filesystem di root senza alterare i pacchetti di base o ricompilare tutto.

Creare la struttura di overlay rootfs e includere l'eseguibile dell'applicazione compilata.

```
mkdir -p resources/board/rootfs_overlay/usr/bin
cp application/lv_benchmark/bin/lvgl-app resources/board/rootfs_overlay/usr/bin/
```

Aggiornare la configurazione.

```
cd output
make menuconfig
```

Trovare l'opzione dell'overlay rootfs `System configuration > Root filesystem overlay directories` e utilizzare il path relativo all'overlay rootfs con l'applicazione (`./resources/board/rootfs_overlay`) e salvare la configurazione.

Ora utilizzare il comando `make` per applicare l'overlay rootfs e verificare che la sysroot sia stata aggiornata come previsto.

```
find . -name lvgl-app
```

Eseguire il *flashing dell'immagine* ed eseguire l'applicazione di benchmark sulla board.

Yocto

The Yocto Project (YP) è un progetto collaborativo open source che aiuta gli sviluppatori a creare sistemi Linux personalizzati, indipendentemente dall'architettura hardware.

Il progetto fornisce un set flessibile di strumenti e uno spazio in cui gli sviluppatori embedded di tutto il mondo possono condividere tecnologie, stack software, configurazioni e best practice, utilizzabili per creare immagini Linux personalizzate per dispositivi embedded e IoT, o ovunque sia necessario un sistema operativo Linux personalizzato.

L'obiettivo di questa sezione è facilitare la comprensione dei concetti di base di Yocto e aiutare i principianti a iniziare a utilizzarlo.

Componenti principali del progetto Yocto

L'esecutore di task BitBake, insieme a vari tipi di file di configurazione, costituisce l'OpenEmbedded-Core (OE-Core). Questa sezione fornisce una panoramica di questi componenti, descrivendone l'utilizzo e le interazioni.

BitBake gestisce l'analisi e l'esecuzione dei file di dati. I dati stessi sono di vario tipo:

- *Ricette*: Forniscono dettagli su particolari componenti software.
- *Dati di Classi*: Estraggono informazioni di build comuni (ad esempio, come compilare un kernel Linux).
- *Dati di Configurazione*: Definiscono impostazioni specifiche della macchina, decisioni sulle policy e così via. I dati di configurazione fungono da collante per collegare tutto

BitBake sa come combinare più sorgenti di dati e fa riferimento a ciascuna sorgente dati come a un layer.

Ecco alcuni brevi dettagli su questi componenti principali.

BitBake

BitBake è lo strumento al centro del sistema di compilazione OpenEmbedded ed è responsabile dell'analisi dei metadati, della generazione di un elenco di task da essi e della loro esecuzione.

Questa sezione introduce brevemente BitBake. Per maggiori informazioni su BitBake, consultare il [Manuale Utente di BitBake](#).

Per visualizzare un elenco delle opzioni supportate da BitBake, utilizzare uno dei seguenti comandi:

```
$ bitbake -h
$ bitbake --help
```

L'uso più comune di BitBake è `bitbake recipename`, dove `recipename` è il nome della ricetta che si desidera compilare (definito come "target"). Il target spesso corrisponde alla prima parte del nome file di una ricetta (ad esempio, "foo" per una ricetta denominata `foo_1.3.0-r0.bb`). Quindi, per elaborare il file della ricetta `matchbox-desktop_1.2.3.bb`, si potrebbe digitare quanto segue:

```
$ bitbake matchbox-desktop
```

Potrebbero esistere diverse versioni di `matchbox-desktop`. BitBake sceglie quella selezionata dalla configurazione della distribuzione. Per maggiori dettagli su come BitBake sceglie tra diverse versioni target e provider, consultare la sezione "[Preferences](#)" del [Manuale Utente di BitBake](#).

BitBake tenta inoltre di eseguire prima eventuali attività dipendenti. Quindi, ad esempio, prima di compilare `matchbox-desktop`, BitBake compilerebbe un cross-compilatore e `glibc` se non fossero già stati compilati.

Un'opzione BitBake utile da considerare è `-k` o `--continue`. Questa opzione indica a BitBake di provare a continuare l'elaborazione del job il più a lungo possibile anche dopo aver riscontrato un errore. Quando si verifica un errore, il target che ha generato l'errore e quelli che dipendono da esso non possono essere ripristinati. Tuttavia, utilizzando questa opzione, è comunque possibile elaborare altre dipendenze.

Ricette

I file con suffisso `.bb` sono file "ricette" (recipe). In generale, una ricetta contiene informazioni su un singolo software. Queste informazioni includono la posizione da cui scaricare il codice sorgente non modificato, eventuali patch da applicare a tale codice sorgente (se necessario), quali opzioni di configurazione speciali applicare, come compilare i file sorgente e come pacchettizzare l'output compilato.

Il termine "package" [pacchetto] viene talvolta utilizzato per riferirsi alle ricette. Tuttavia, poiché il termine "package" viene utilizzato per l'output pacchettizzato dal sistema di build di OpenEmbedded (ad esempio, i file `.ipk` o `.deb`), questo documento evita di utilizzare il termine "package" quando si fa riferimento alle ricette.

Classi

I file di classe (`.bbclass`) contengono informazioni utili da condividere tra i file di ricette. Un esempio è la classe `autotools*`, che contiene impostazioni comuni per qualsiasi applicazione compilata con `GNU Autotools` <https://en.wikipedia.org/wiki/GNU_Autotools>. Il capitolo "[Classi](#)" nel [Manuale di Riferimento del Progetto Yocto](#) fornisce dettagli sulle classi e su come utilizzarle.

Configurazioni

I file di configurazione (`.conf`) definiscono diverse variabili di configurazione che governano il processo di compilazione di OpenEmbedded. Questi file rientrano in diverse aree che definiscono le opzioni di configurazione della macchina, le opzioni di configurazione della distribuzione, le opzioni di ottimizzazione del compilatore, le opzioni di configurazione comuni generali e le opzioni di configurazione utente in `conf/local.conf`, che si trova nella [Build Directory](#).

I Layer

I layer sono repository che contengono metadati correlati (ovvero set di istruzioni) che indicano al sistema di build di OpenEmbedded come compilare un target. Il modello a layer del progetto Yocto facilita la collaborazione, la condivisione, la personalizzazione e il riutilizzo all'interno dell'ambiente di sviluppo Yocto Project. I layer separano logicamente le informazioni per il progetto. Ad esempio, è possibile utilizzare un layer per contenere tutte le configurazioni di un particolare componente hardware. Isolare le configurazioni specifiche dell'hardware consente di condividere altri metadati utilizzando un layer diverso, in cui tali metadati potrebbero essere comuni a diversi componenti hardware.

Nell'ambiente di sviluppo Yocto Project sono presenti molti layer. Sia [Yocto Project Compatible Layer Index](#) che [OpenEmbedded Layer Index](#) contengono layer che è possibile utilizzare o sfruttare.

Per convenzione, i layer nel Progetto Yocto seguono una forma specifica. Conformarsi a una struttura nota consente a BitBake di fare ipotesi durante le build su dove trovare i tipi di metadati. È possibile trovare procedure e informazioni sugli strumenti (ad esempio `bitbake-layers`) per la creazione di layer adatti al Progetto Yocto nella sezione "comprensione e creazione di layer" dello Yocto Project Development Tasks Manual.

LVGL in Yocto

Questo capitolo serve come guida per aiutare a creare una ricetta per un'applicazione che utilizza LVGL. Sebbene il processo verrà illustrato per Raspberry Pi 3 (64 bit), può essere applicato a qualsiasi board in modo simile.

Pacchetti Host di Build

È necessario installare i pacchetti host essenziali sul proprio host di build. Il seguente comando installa i pacchetti host basati su una distribuzione Ubuntu

```
sudo apt install gawk wget git diffstat unzip texinfo gcc build-essential \
chrpath socat cpio python3 python3-pip python3-pexpect xz-utils \
debianutils iputils-ping python3-git python3-jinja2 python3-subunit zstd \
liblzf4-tool file locales libacl
```

Nota

Per i requisiti dei pacchetti host su tutte le distribuzioni Linux supportate, consultare la sezione [Required Packages for the Build Host](#) nello Yocto Project Reference Manual.

Utilizzare Git per clonare i repository richiesti

Dopo questa sezione, l'albero delle cartelle sarà simile a questo

```
lvgl_yocto_guide/
├── build/
│   ├── cache/
│   ├── conf/
│   ├── downloads/
│   └── ...
└── sources/
    ├── meta-openembedded
    ├── meta-raspberrypi
    └── poky
```

Iniziare la creazione dell'architettura delle cartelle

```
mkdir -p lvgl_yocto_guide/sources
```

Clonare Poky, meta-openembedded e meta-raspberrypi nei sorgenti

```
cd lvgl_yocto_guide/sources
git clone --branch scarthgap https://git.yoctoproject.org/poky.git
git clone --branch scarthgap https://git.openembedded.org/meta-openembedded
git clone --branch scarthgap git://git.yoctoproject.org/meta-raspberrypi
```

Creare un'immagine di base

Per capire meglio cosa sta succedendo, creiamo l'immagine come se fosse stata preparata per noi

```
cd .. # go back to the root folder
source sources/poky/oe-init-build-env
```

Ora si dovrebbe avere una cartella chiamata build accanto a sources. L'ultimo comando imposta anche la directory corrente sulla directory di build.

Nella directory di build, c'è una cartella conf con alcuni file al suo interno

```
└── build/
    └── conf/
        ├── bblayers.conf
        ├── conf-notes.txt
        ├── conf-summary.txt
        ├── local.conf
        └── templateconf.cfg
```

I file importanti per noi sono local.conf e bbayers.conf

Per aggiungere layer al progetto ci sono 2 opzioni:

Comandi Bitbake (Opzione 1)

Si deve stare nello stesso terminale in cui è stato eseguito il comando source. Si aggiungono i layer col comando Bitbake in questo modo:

```
bitbake-layers add-layer ../sources/meta-openembedded
bitbake-layers add-layer ../sources/meta-raspberrypi
```

Modifica del file di configurazione (Opzione 2)

Aprire il file conf/bblayers.conf e aggiungere manualmente i path:

```
# POKY_BBLAYERS_CONF_VERSION is increased each time build/conf/bblayers.conf
# changes incompatibly
POKY_BBLAYERS_CONF_VERSION = "2"

BBPATH = "${TOPDIR}"
BBFILES ?= ""

BBLAYERS ?= " \
${TOPDIR}../sources/poky/meta \
${TOPDIR}../sources/poky/meta-poky \
${TOPDIR}../sources/poky/meta-yocto-bsp \
${TOPDIR}../sources/meta-raspberrypi \
${TOPDIR}../sources/meta-openembedded/meta-oe \
${TOPDIR}../sources/meta-openembedded/meta-multimedia \
${TOPDIR}../sources/meta-openembedded/meta-networking \
${TOPDIR}../sources/meta-openembedded/meta-python \
"
```

Per assicurarsi che i layer siano stati aggiunti come previsto, eseguire il comando Bitbake per visualizzarli tutti:

```
bitbake-layers show-layers
```

Dovrebbero essere elencati i seguenti layer:

- core
- yocto
- yoctobsp
- raspberrypi
- openembedded-layer
- multimedia-layer
- networking-layer
- meta-python

Build per RaspberryPi3 64

Le configurazioni macchina disponibili per RaspberryPi possono essere elencate in questo modo:

```
ls ./sources/meta-raspberrypi/conf/machine/*.conf
```

Per compilare un'immagine per RaspberryPi3 64 bit, modificare il file `local.conf` sostituendo il valore di default `MACHINE ??=` in questo modo

```
MACHINE ??= "raspberrypi3-64"
```

Per compilare l'immagine da utilizzare, è necessario aggiungere anche quanto segue al file:

```
LICENSE_FLAGS_ACCEPTED = "synaptics-killswitch"
```

Nota

Ulteriori informazioni su questa variabile sono disponibili [qui](#).

Tutto è configurato, è ora di compilare l'immagine:

```
bitbake core-image-base
```

Panoramica della build

Esaminiamo le cartelle di build per capire cosa è successo.

Downloads

La posizione della cartella può essere modificata con la variabile `DL_DIR`.

La cartella Downloads è una parte fondamentale dell'ambiente di build di Yocto, in quanto funge da cache locale per tutte le sorgenti e le dipendenze esterne. Questo meccanismo di caching è essenziale per build efficienti, in quanto garantisce che build ripetute non comportino perdite di tempo nel download ripetuto degli stessi file.

tmp/deploy

Questa cartella contiene gli artefatti di output finali pronti per il "deployment" [distribuzione]. Questi artefatti includono immagini del kernel, file system di root, pacchetti, SDK, file del bootloader e altri componenti generati durante il processo di build di Yocto.

Ci sono alcune cartelle chiave:

- **images**: contiene le immagini che possono essere flashate o distribuite sul dispositivo target. File come il kernel Linux, il file system di root (ad esempio, .ext4, .tar.gz, .squashfs), i bootloader (ad esempio, U-Boot) e altre immagini avviabili per il dispositivo si trovano qui. È organizzato in base alla macchina (o board target) per cui è stata compilata l'immagine.
- **rmp/deb/ipk**: Queste cartelle contengono i singoli pacchetti software generati durante la compilazione, nel formato di pacchetto specificato (RPM, DEB o IPK). Questi pacchetti vengono in genere creati durante la compilazione del progetto Yocto con il supporto per la gestione dei pacchetti abilitato. Possono essere successivamente installati sul dispositivo di destinazione utilizzando gli strumenti di gestione dei pacchetti.
- **sdk**: Questa sottodirectory contiene i Software Development Kit (SDK) che è possibile utilizzare per la cross-compilazione delle applicazioni per il sistema di target. Gli SDK vengono generati quando si utilizza il comando `bitbake -c populate_sdk`.

tmp/sysroots-components

Questa cartella rappresenta un approccio modulare introdotto in Yocto per la gestione delle sysroot. Suddivide la sysroot in singoli componenti, che possono essere considerati come blocchi costitutivi o pacchetti che costituiscono l'intero ambiente sysroot.

Ogni componente corrisponde a un pacchetto o dipendenza specifica che viene inserita nella sysroot. Questo approccio consente un riutilizzo più efficiente delle sysroot su più pacchetti o build e facilita il monitoraggio e la gestione delle dipendenze durante il processo di build.

La directory sysroot-components aiuta a ottimizzare il processo di build perché, se un pacchetto cambia, Yocto deve solo aggiornare il componente corrispondente anziché ricostruire o copiare l'intera sysroot.

Se sono stati seguiti i passaggi precedenti, ecco le cartelle che si troveranno:

- **all**: File indipendenti dall'architettura.
- **cortexa53**: File per l'architettura Cortex-A53 (ARMv8-A).
- **manifests**: File "track" installati nella sysroot per pacchetto.
- **raspberrypi3_64**: File specifici per Raspberry Pi 3 (64 bit).
- **x86_64**: File per l'architettura x86_64 (PC), in genere per tool di cross-compilazione.
- **x86_64-nativesdk**: File relativi all'SDK per la cross-compilazione su un host x86_64.

Ogni cartella corrisponde a componenti rilevanti per l'architettura specifica e, nel loro insieme, formano l'ambiente completo necessario per compilare ed eseguire il software per i sistemi target e host.

tmp/sysroots-uninative

La directory sysroots-uninative in Yocto viene utilizzata per supportare la funzionalità "uninative", che consente build più riproducibili garantendo che l'ambiente di build rimanga coerente su diversi sistemi host. In sostanza, fornisce un modo per utilizzare gli stessi strumenti di build nativi su diverse distribuzioni Linux.

tmp/work

La cartella work in Yocto è una directory chiave nella cartella tmp che contiene tutti gli artefatti di build temporanei per ogni pacchetto durante il processo di build. È qui che avviene la build e la compilazione effettiva dei singoli pacchetti o ricette. Ogni pacchetto (o "recipe" [ricetta]) elaborato da BitBake genera file e directory temporanei all'interno di questa cartella work.

La cartella work è in genere strutturata in base all'architettura della macchina e al pacchetto. Ecco come funziona in genere:

```
work/
└── <architecture>/
    └── <package>/
        └── <version>/
            ├── temp/
            ├── work/
            └── sysroot-destdir/
                └── image/
```

Questa cartella è molto importante e di grande aiuto durante le fasi di debug. La sottocartella temp contiene log importanti che possono aiutare a diagnosticare i problemi di build. È qui che si andrà a cercare quando una build non riesce a capire cosa è andato storto.

Ricetta LVGL

Maggiori informazioni sulle ricette sono disponibili nella sezione [Ricette](#).

C'è una ricetta in meta-openembedded fin dalla versione honister.

Tabella 1: Versione della ricetta lvgl

Branch	Ricetta
scarthgap (Yocto Project 5.0)	lvgl 9.1.0
nanbield (Yocto Project 4.3)	lvgl 8.3.10
mickledore (Yocto Project 4.2)	lvgl 8.1.0
langdale (Yocto Project 4.1)	lvgl 8.1.0
langdale (Yocto Project 4.1)	lvgl 8.1.0
kirkstone (Yocto Project 4.0)	lvgl 8.0.3

In questa guida, ci troviamo sul branch `scarthgap`, quindi utilizziamo `lvgl 9.1.0`.

Approfondiamo questa ricetta per capire cosa viene fatto. L'obiettivo è aggiungere questa libreria come oggetto "shared" nel rootfs target e generare un SDK con lvgl.

Questo è il path delle ricette lvgl: `lvgl_yocto_guide/sources/meta-openembedded/meta-oe/recipes-graphics/lvgl`

Ecco l'architettura della cartella delle ricette lvgl:

```
lvgl
├── files
│   ├── 0002-fix-sdl-handle-both-LV_IMAGE_SRC_FILE-and-LV_IMAGE_S.patch
│   ├── 0003-Make-fbdev-device-node-runtime-configurable-via-envi.patch
│   ├── 0004-Factor-out-fbdev-initialization-code.patch
│   ├── 0005-Add-DRM-KMS-example-support.patch
│   ├── 0006-Add-SDL2-example-support.patch
│   ├── 0007-fix-cmake-generate-versioned-shared-libraries.patch
│   └── 0008-fix-fbdev-set-resolution-prior-to-buffer.patch
└── lv-conf.inc
└── lvgl_9.1.0.bb
└── lvgl-demo-fb_9.1.0.bb
```

- La cartella `file` contiene tutte le patch che possono essere applicate durante la compilazione della ricetta.
- `lv_conf.inc` è un file include, che di solito contiene impostazioni di configurazione comuni per LVGL che possono essere condivise tra più ricette.
- `lvgl_9.1.0.bb` è la ricetta per compilare la libreria lvgl.
- `lvgl-demo-fb_9.1.0.bb` è una ricetta per compilare un'applicazione utilizzando lvgl.

Per ora, comprendiamo la ricetta della libreria lvgl.

```
# SPDX-FileCopyrightText: Huawei Inc.
#
# SPDX-License-Identifier: MIT

HOMEPAGE = "https://lvgl.io/"
DESCRIPTION = "LVGL is an OSS graphics library to create embedded GUI"
SUMMARY = "Light and Versatile Graphics Library"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://LICENCE.txt;md5=bf1198c89ae87f043108cea62460b03a"

SRC_URI = " \
    git://github.com/lvgl/lvgl;protocol=https;branch=master \
    file://0002-fix-sdl-handle-both-LV_IMAGE_SRC_FILE-and-LV_IMAGE_S.patch \
    file://0007-fix-cmake-generate-versioned-shared-libraries.patch \
    file://0008-fix-fbdev-set-resolution-prior-to-buffer.patch \
    "
SRCREV = "elcb21b2723d391b885de4b2ee5cc997eccca91"

inherit cmake

EXTRA_OECMAKE = "-DLIB_INSTALL_DIR=${baselib} -DBUILD_SHARED_LIBS=ON"
S = "${WORKDIR}/git"

require lv-conf.inc

do_install:append() {
    install -d "${D}/${includedir}/${PN}"
    install -m 0644 "${S}/lv_conf.h" "${D}/${includedir}/${PN}/lv_conf.h"
}

FILES_${PN}-dev += "\ \
    ${includedir}/${PN}/ \ \
    "
```

LICENSE: Indica la licenza del software, indicando che è distribuito con licenza MIT.

LIC_FILES_CHKSUM: Questa variabile contiene un checksum (in questo caso, un hash MD5) per il file di licenza (qui si presume sia `LICENCE.txt`). Questo aiuta a garantire l'integrità del file di licenza incluso nel pacchetto. Nel repository LVGL, è presente un file `LICENCE.txt`. Per ottenere il valore dell'hash del file, è possibile eseguire questo comando: `md5sum LICENCE.txt`

SRC_URI: Specifica le posizioni del codice sorgente e delle patch per la ricetta:

- Il repository sorgente principale per LVGL che specifica il branch master.
- I seguenti argomenti sono i file di patch locali che verranno applicati al codice sorgente durante il processo di build.

SRCREV: Definisce il commit specifico (in questo caso, un hash Git SHA) da cui verrà recuperato il codice sorgente. Questo garantisce che la build utilizzi una versione coerente del codice.

inherit: Questa riga indica che la ricetta utilizza la classe `cmake`, che fornisce funzionalità per la creazione di progetti che utilizzano CMake come sistema di build.

EXTRA_OECMAKE: Opzioni aggiuntive passate a CMake durante la fase di configurazione.

S: Questa variabile definisce la directory sorgente in cui verrà posizionato il codice sorgente decompresso dopo il recupero. `${WORKDIR}` è una variabile standard in BitBake che punta alla directory di lavoro per la ricetta.

require: Questa riga include un altro file di configurazione, `lv-conf.inc`, che probabilmente contiene opzioni di configurazione aggiuntive o variabili specifiche della libreria LVGL.

FILES: Questa è una variabile BitBake utilizzata per specificare i file che devono essere inclusi in un particolare pacchetto. In questo caso, la variabile specifica i file per un pacchetto correlato allo sviluppo (ad esempio, file header).

I Task della Ricetta

Quando una ricetta viene compilata, eseguirà più task. È possibile eseguire ogni task manualmente per capire cosa viene generato in ogni passaggio, oppure è possibile eseguire `bitbake lvgl` per eseguire tutti i task.

Fetch (do_fetch)

```
bitbake lvgl -c fetch
```

Il task "fetch" recupera il codice sorgente del pacchetto dal repository locale o remoto.

L'indirizzo del Repository di recupero dev'essere memorizzato nella variabile **SRC_URI**. Nella variabile **SRCPREV** viene definito l'hash di commit del repository GitHub.

Una volta completato il task di recupero (fetch), è possibile trovare i codici sorgente recuperati in `build/downloads`.

Per questa ricetta, si troverà una nuova cartella qui: `lvgl_yocto_guide/build/downloads/git2/github.com.lvgl.lvgl`.

Si può anche trovare la cartella `architecture` creata in `lvgl_yocto_guide/build/tmp/work/cortexa53-poky-linux/lvgl` ma queste cartelle sono vuote poiché è stato eseguito solo il fetch.

Unpack (do_unpack)

```
bitbake lvgl -c unpack
```

Il task "unpack" scompatta il pacchetto scaricato co task Fetch.

Nella cartella `lvgl_yocto_guide/build/tmp/work/cortexa53-poky-linux/lvgl/9.1.0`, ora si può trovare il codice sorgente in `git` (come definito nella ricetta). Si vedranno anche le patch che verranno applicate nel passaggio successivo. Quindi, per ora, i sorgenti non sono modificati e sono uguali al commit specificato.

Patch (do_patch)

```
bitbake lvgl -c patch
```

Il task "patch" individua i file delle patch e applica le patch ai sorgenti, se disponibili. Questo è un task facoltativo, eseguito se la patch è disponibile.

Il file della patch è definito anche nella variabile **SRC_URI**. Per default viene eseguito nella directory sorgente corrente `${S}`.

Configure (do_configure)

```
bitbake lvgl -c configure
```

Il task di configurazione configura il codice sorgente abilitando e disabilitando eventuali opzioni di build e di configurazione per il software in fase di compilazione prima della compilazione, se disponibile.

Questo è un passaggio facoltativo, eseguito se disponibile.

In questo caso, crea una directory di build e richiama CMake per configurare il progetto, specificando le opzioni di build e i path in base alle variabili nella ricetta. Genera i Makefile o i file di progetto necessari per la build. Inoltre, ci sono operazioni aggiunte al task in `lv-conf.inc`.

Quindi, alla fine del task, in `lvgl_yocto_guide/build/tmp/work/ cortexa53-poky-linux/lvgl/9.1.0`, troverete una cartella `build` che è stata generata eseguendo il comando CMake, ma non è ancora stato compilato nulla. Inoltre, le sysroot contengono tutto il necessario per compilare la libreria lvgl.

Compile (do_compile)

```
bitbake lvgl -c compile
```

Il task Compilation compila il codice sorgente se sono disponibili passaggi di compilazione e genera un file binario.

Questo è un passaggio facoltativo, eseguito se la compilazione è disponibile.

Se sono presenti passaggi di compilazione, questi vengono definiti nella funzione `do_compile()` di bitbake.

Come nel task precedente, questa operazione viene gestita da `inherit cmake`.

Nella cartella di build, ora è possibile visualizzare la libreria compilata. I file `.so` sono disponibili in `lvgl_yocto_guide/build/tmp/work/ cortexa53-poky-linux/lvgl/9.1.0/build/lib`.

Una volta completata questa attività, tutto è pronto per l'installazione.

Install (do_install)

```
bitbake lvgl -c install
```

Il task Install copia i file da impacchettare nell'area di archiviazione `${D}`. Questo viene eseguito con la directory di lavoro corrente `${S}`, che è la directory di compilazione.

Crea la directory necessaria per i file header nella directory di installazione di destinazione. Installa il file header `lv_conf.h` dalla directory sorgente in quella degli include appropriata nel path di destinazione, assicurandosi che abbia i permessi corretti.

Anche la libreria lvgl (file `.so`) è pronta per essere installata nell'immagine finale. È stata creata una nuova cartella `image`.

■ Importante

Nel file `build/conf/local.conf`, aggiungere queste 2 righe alla fine del file:

```
IMAGE_INSTALL_append = " lvgl"
TOOLCHAIN_HOST_TASK_append = " lvgl"
```

Questo aggiungerà la libreria lvgl all'immagine generata e la aggiungerà anche all'SDK host che genereremo in seguito.

Con queste modifiche, ora si può eseguire nuovamente la ricetta dell'immagine:

```
bitbake core-image-base
```

Questo eseguirà tutte le attività descritte in precedenza.

Se tutto è andato bene, si troverà il file `build/tmp/deploy/ rpm/cortexa53/lvgl-9.1.0-r0.cortexa53.rpm` e altri file rpm relativi a lvgl.

Generazione dell'SDK

La generazione di un Software Development Kit (SDK) in Yocto svolge diverse funzioni importanti, in particolare nello sviluppo embedded:

- **Ambiente di sviluppo:** L'SDK fornisce agli sviluppatori un ambiente di sviluppo pronto all'uso, personalizzato per una specifica piattaforma hardware di destinazione. Questo include gli strumenti, le librerie e gli header necessari per creare applicazioni eseguibili sul dispositivo target.
- **Cross-Compilazione:** L'SDK consente agli sviluppatori di compilare in modo incrociato le applicazioni da una macchina host (tipicamente con architettura x86 o x86_64) per un'architettura diversa (ad esempio, ARM). Questo è essenziale per i sistemi embedded, dove lo sviluppo avviene spesso su macchine più potenti.
- **Sviluppo Semplificato:** Impacchettando tutti i componenti necessari, l'SDK semplifica il processo di configurazione di un ambiente di sviluppo. Gli sviluppatori non devono installare e configurare manualmente strumenti e librerie; tutto il necessario è incluso nell'SDK.
- **Ambiente di Build Coerente:** L'SDK garantisce che gli sviluppatori lavorino con le stesse versioni e gli stessi strumenti utilizzati nella build di Yocto, il che aiuta a evitare problemi di compatibilità e garantisce che le applicazioni si comportino come previsto sul dispositivo target.

Per generare un SDK dell'ambiente, eseguire il seguente comando:

```
bitbake core-image-base -c populate_sdk
```

Questo creerà uno script `build/tmp/deploy/sdk/poky-glibc-x86_64-core-image-base-cortexa53-raspberrypi3-64-toolchain-5.0.4.sh`. Questo script consente di installare l'SDK dove si preferisce. Ecco l'output di esecuzione dello script

```
$ ./sdk/poky-glibc-x86_64-core-image-base-cortexa53-raspberrypi3-64-toolchain-5.0.4.sh
Poky (Yocto Project Reference Distro) SDK installer version 5.0.4
=====
Enter target directory for SDK (default: /opt/poky/5.0.4): /opt/poky/sdk-with-lvgl
You are about to install the SDK to "/opt/poky/sdk-with-lvgl". Proceed [Y/n]? y
```

Per assicurarsi che l'SDK sia stato generato con lvgl installato, si va al path in cui è stato estratto l'SDK e si trovano tutti i file lvgl:

```
cd /opt/poky/5.0.4/sysroots/cortexa53-poky-linux
find . -name "*lvgl*"
```

I file .so che si troveranno dipenderanno dalla configurazione LVGL utilizzata.

Ora, per utilizzare l'ambiente SDK e cross-compilare un'applicazione:

```
source /opt/poky/5.0.4/environment-setup-cortexa53-poky-linux
```

Nota

I tool disponibili nella libreria LVGL dipenderanno dalla configurazione di build con cui è stata realizzata la ricetta.

Ricetta personalizzata

Fino a questa sezione, tutto era già stato preparato. Abbiamo utilizzato ricette esistenti. L'obiettivo qui è creare una ricetta da zero e aggiungere il binario generato all'immagine.

Creare un layer

Per prima cosa, si crea un layer e lo si aggiunge al file di configurazione.

```
bitbake-layers create-layer ../sources/meta-myvg
bitbake-layers add-layer ../sources/meta-myvg
```

Nella cartella `sources`, è stata creata una nuova cartella: `meta-myvg` e l'albero delle directory dovrebbe apparire come segue.

```

└── conf
    └── layer.conf
└── COPYING.MIT
└── README
└── recipes-example
    └── example
        └── example_0.1.bb

```

Creare una ricetta

Seguendo questa struttura, si crea una cartella contenente le ricette per compilare una o più applicazioni utilizzando lvgl.

```

cd ../../sources/meta-mylvgl
mkdir -p recipes-lvglapp/lvgl-fbdev-benchmark/files
touch recipes-lvglapp/lvgl-fbdev-benchmark/lvglbenchmarkfbdev_2.4.bb

```

Ci concentreremo su un'applicazione che clonerà un repository git lvgl e vi applicherà le patch per le nostre esigenze.

Il contenuto di `recipes-lvglapp/lvgl-fbdev-benchmark/lvglbenchmarkfbdev_2.4.bb`

```

DESCRIPTION = "My C++ lvgl app with CMake"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835ade698e0bcf8506ecda2f7b4f302"

# Update SRC_URI to point to the GitHub repository
SRC_URI = "git://github.com/lvgl/lv_port_linux.git;branch=master;protocol=https \
    file://0001-change-config-file-to-run-fbdev-with-2-threads.patch \
    file://0002-adapt-CMakeLists-file-to-compile-and-link-fbdev.patch \
    "

S = "${WORKDIR}/git"
inherit cmake

CMAKE_PROJECT_NAME = "lvgl_app"
CMAKE_PROJECT_VERSION = "2.0"

do_install() {
    install -d ${D}${bindir}
    install -m 0755 ${S}/bin/lvglbenchmark ${D}${bindir}
}

```

I sorgenti provengono dal repository `lv_port_linux`. Applichiamo 2 patch per modificare `CMakeLists.txt` e `lv_conf.h`.

Patch 1

Creazione del primo file patch

```
touch 0001-change-config-file-to-run-fbdev-with-2-threads.patch
```

Contenuto di `0001-change-config-file-to-run-fbdev-with-2-threads.patch`

```

From d49d475d57f69d6172a6b38095ebf3b887f0e731 Mon Sep 17 00:00:00 2001
From: Gabriel Catel Torres <gabriel.catel@edgemtech.ch>
Date: Tue, 1 Oct 2024 10:28:55 +0200
Subject: [PATCH] change config file to run fbdev with 2 threads

...
lv_conf.h | 34 ++++++-----.....
1 file changed, 17 insertions(+), 17 deletions(-)

diff --git a/lv_conf.h b/lv_conf.h
index 62a834f..58fbe7a 100644
--- a/lv_conf.h
+++ b/lv_conf.h
@@ -39,9 +39,9 @@
 * - LV_STDLIB_RTTHREAD:     RT-Thread implementation
 * - LV_STDLIB_CUSTOM:      Implement the functions externally
 */
#define LV_USE_STDLIB_MALLOC   LV_STDLIB_BUILTIN
#define LV_USE_STDLIB_STRING  LV_STDLIB_BUILTIN
#define LV_USE_STDLIB_SPRINTF LV_STDLIB_BUILTIN
+#define LV_USE_STDLIB_MALLOC   LV_STDLIB_CLIB
+#define LV_USE_STDLIB_STRING  LV_STDLIB_CLIB
+#define LV_USE_STDLIB_SPRINTF LV_STDLIB_CLIB

#define LV_STDINT_INCLUDE      <stdint.h>
#define LV_STDDEF_INCLUDE       <stddef.h>
@@ -89,7 +89,7 @@
 * - LV_OS_WINDOWS
 * - LV_OS_MQX
 * - LV_OS_CUSTOM */
#define LV_USE_OS   LV_OS_NONE
#define LV_USE_OS   LV_OS_PTHREAD

#if LV_USE_OS == LV_OS_CUSTOM
#define LV_OS_CUSTOM_INCLUDE <stdint.h>
@@ -117,12 +117,12 @@
 * and can't be drawn in chunks. */

```

(continues on next page)

(continua dalla pagina precedente)

```

/* The target buffer size for simple layer chunks. */
#define LV_DRAW_LAYER_SIMPLE_BUF_SIZE (24 * 1024) /* [bytes] */
#define LV_DRAW_LAYER_SIMPLE_BUF_SIZE (512 * 1024) /* [bytes] */

/* The stack size of the drawing thread.
 * NOTE: If FreeType or ThorVG is enabled, it is recommended to set it to 32KB or more.
 */
#define LV_DRAW_THREAD_STACK_SIZE (8 * 1024) /* [bytes] */
#define LV_DRAW_THREAD_STACK_SIZE (32 * 1024) /* [bytes] */

#define LV_USE_DRAW_SW
#if LV_USE_DRAW_SW == 1
@0 -147.7 +147.7 @0
/* Set the number of draw unit.
 * > 1 requires an operating system enabled in 'LV_USE_OS'
 * > 1 means multiple threads will render the screen in parallel */
- #define LV_DRAW_SW_DRAW_UNIT_CNT 1
+ #define LV_DRAW_SW_DRAW_UNIT_CNT 2

/* Use Arm-2D to accelerate the sw render */
#define LV_USE_DRAW_ARM2D_SYNC 0
@0 -979.8 +979.8 @0
#define LV_USE_LINUX_FBDEV 1
#define LV_USE_LINUX_FBDIV
    #define LV_LINUX_FBDIV_BSD 0
- #define LV_LINUX_FBDIV_RENDER_MODE LV_DISPLAY_RENDER_MODE_PARTIAL
- #define LV_LINUX_FBDIV_BUFFER_COUNT 0
+ #define LV_LINUX_FBDIV_RENDER_MODE LV_DISPLAY_RENDER_MODE_DIRECT
+ #define LV_LINUX_FBDIV_BUFFER_COUNT 2
    #define LV_LINUX_FBDIV_BUFFER_SIZE 60
#endif

@0 -1069.19 +1069.19 @0
#define LV_USE_DEMO_WIDGETS 1

/* Demonstrate the usage of encoder and keyboard */
#define LV_USE_DEMO_KEYPAD_AND_ENCODER 1
#define LV_USE_DEMO_KEYPAD_AND_ENCODER 0

/* Benchmark your system */
#define LV_USE_DEMO_BENCHMARK 1

/* Render test for each primitives. Requires at least 480x272 display */
#define LV_USE_DEMO_RENDER 1
#define LV_USE_DEMO_RENDER 0

/* Stress test for LVGL */
#define LV_USE_DEMO_STRESS 1
#define LV_USE_DEMO_STRESS 0

/* Music player demo */
#define LV_USE_DEMO_MUSIC 1
#define LV_USE_DEMO_MUSIC 0
#if LV_USE_DEMO_MUSIC
    #define LV_DEMO_MUSIC_SQUARE 0
    #define LV_DEMO_MUSIC_LANDSCAPE 0
@0 -1091.16 +1091.16 @0
#endif

/* Flex layout demo */
#define LV_USE_DEMO_FLEX_LAYOUT 1
#define LV_USE_DEMO_FLEX_LAYOUT 0

/* Smart-phone like multi-language demo */
#define LV_USE_DEMO_MULTILANG 1
#define LV_USE_DEMO_MULTILANG 0

/* Widget transformation demo */
#define LV_USE_DEMO_TRANSFORM 1
#define LV_USE_DEMO_TRANSFORM 0

/* Vector graphic demo */
#define LV_USE_DEMO_VECTOR_GRAPHIC 0
-
```

Patch 2

Creazione del primo file patch

[touch 0002-adapt-CMakeLists-file-to-compile-and-link-fbdev.patch](#)

Contenuto di 0002-adapt-CMakeLists-file-to-compile-and-link-fbdev.patch

```

From ad464e0683aced74085fb89895b8d040cad0206 Mon Sep 17 00:00:00 2001
From: Gabriel Catel Torres <gabriel.catel@edgentech.ch>
Date: Tue, 1 Oct 2024 10:31:29 +0200
Subject: [PATCH] adapt CMakeLists file to compile and link only for fbdev

...
CMakeLists.txt | 17 +++++-----...
1 file changed, 5 insertions(+), 12 deletions(-)

diff --git a/CMakeLists.txt b/CMakeLists.txt
index 658193f..ad56cc2 100644
--- a/CMakeLists.txt
+++ b/CMakeLists.txt
@@ -1,8 +1,8 @@
cmake_minimum_required(VERSION 3.10)
project(lvgl)

-set(CMAKE_C_STANDARD 99)#C99 # lvgl officially support C99 and above
-set(CMAKE_CXX_STANDARD 17)#C17
+set(CMAKE_C_STANDARD 99)

```

(continues on next page)

(continua dalla pagina precedente)

```
+set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

set(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/bin)
@@ -10,15 +10,8 @@ set(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/bin)
add_subdirectory(lvgl)
target_include_directories(lvgl PUBLIC ${PROJECT_SOURCE_DIR})

-add_executable(main main.c mouse_cursor_icon.c)
+add_executable(lvglbenchmark main.c mouse_cursor_icon.c)

#include(${CMAKE_CURRENT_LIST_DIR}/lvgl/tests/FindLibDRM.cmake)
include_directories(${Libdrm_INCLUDE_DIRS})
-
-find_package(SDL2)
-find_package(SDL2_image)
include_directories(${SDL2_INCLUDE_DIRS} ${SDL2_IMAGE_INCLUDE_DIRS})
-
-target_link_libraries(main lvgl lvgl::examples lvgl::demos lvgl::thorvg ${SDL2_LIBRARIES} ${SDL2_IMAGE_LIBRARIES} ${Libdrm_LIBRARIES} m pthread)
+target_link_libraries(lvglbenchmark lvgl lvgl::examples lvgl::demos lvgl::thorvg m pthread)
+add_custom_target (run COMMAND ${EXECUTABLE_OUTPUT_PATH}/lvglbenchmark DEPENDS main)

...
2.34.1
```

Compilare la ricetta

Ora si dovrebbe essere in grado di visualizzare la ricetta con l'elenco delle ricette esistenti.

```
bitbake-layers show-recipes | grep lvglbenchmarkfbdev
```

Si aggiunge la ricetta al progetto. Modificare il file build/conf/local.conf aggiungendo questa riga.

```
IMAGE_INSTALL_append = " lvglbenchmarkfbdev"
```

Ora compilare l'immagine.

```
bitbake core-image-base
```

Eseguire questo comando per assicurarsi che il binario sia stato generato e installato nel rootfs.

```
# Run this command in build/tmp folder
find . -wholename "*bin/lvglbench"
```

Terminologia dello Yocto Project

Iniziare a usare Yocto può essere complicato. Sono molti i termini utilizzati, specifici per Yocto e per l'ambiente Bitbake.

Un elenco di termini e definizioni che gli utenti alle prime armi con l'ambiente di sviluppo del progetto Yocto potrebbero trovare utili è disponibile [qui](#).

Glossario delle Variabili di Yocto

Questo capitolo elenca le variabili di base utilizzate nella guida LVGL di Yocto e fornisce una panoramica della loro funzione e del loro contenuto.

Un glossario più completo delle variabili è disponibile nel [Glossario delle variabili Yocto](#). Questa sezione tratta molte variabili utilizzate nel sistema di build OpenEmbedded.

S

Posizione nella "Build Directory" in cui risiede il codice sorgente della ricetta spaccettata. Per default, questa directory è \${WORKDIR}/\${BPN}-\${PV}, dove \${BPN} è il nome della ricetta di base e \${PV} è la versione della ricetta. Se il tarball sorgente estrae il codice in una directory con un nome diverso da \${BPN}-\${PV}, o se il codice sorgente viene recuperato da un SCM come Git o Subversion, è necessario impostare S nella ricetta in modo che il sistema di build di OpenEmbedded sappia dove trovare il codice sorgente decompresso.

Ad esempio, si supponga una "Source Directory" di primo livello denominata poky e una "Build Directory" di default in poky/build. In questo caso, la directory di lavoro utilizzata dal sistema di build per conservare la ricetta decompressa per il db (Berkeley DB) è la seguente:

```
poky/build/tmp/work/qemux86-poky-linux/db/5.1.19-r3/db-5.1.19
```

Il codice sorgente decompresso si trova nella cartella db-5.1.19.

Il prossimo esempio presuppone un repository Git. Per default, i repository Git vengono clonati in \${WORKDIR}/git durante il do_fetch. Poiché questo path è diverso dal valore predefinito di S, è necessario impostarlo specificamente in modo che il sorgente possa essere individuato:

```
SRC_URI = "git://path/to/repo.git;branch=main"
S = "${WORKDIR}/git"
```

D

Directory di destinazione. La posizione nella Build Directory in cui i componenti vengono installati dal task do_install. Questa posizione di defaults è:

```
 ${WORKDIR}/image
```

Nota

I task che leggono o scrivono in questa directory devono essere eseguiti con fakeroot.

B

La directory all'interno della Build Directory in cui il sistema di build OpenEmbedded inserisce gli oggetti generati durante il processo di build di una ricetta. Per default, questa directory è la stessa della directory S, definita come:

```
S = "${WORKDIR}/${BP}"
```

È possibile separare la directory (S) e la directory puntata dalla variabile B. La maggior parte delle ricette basate su Autotools supporta la separazione di queste directory. Il sistema di build utilizza per default directory separate per gcc e alcune ricette del kernel.

WORKDIR

Il percorso della directory di lavoro in cui il sistema di build di OpenEmbedded crea una ricetta. Questa directory si trova all'interno della struttura di directory TMPDIR ed è specifica della ricetta in fase di compilazione e del sistema per cui viene compilata.

La directory WORKDIR è definita come segue:

```
 ${TMPDIR}/work/${MULTIMACH_TARGET_SYS}/${PN}/${EXTENDPE}${PV}-${PR}
```

La directory effettiva dipende da diversi fattori:

- **TMPDIR**: Directory di output della build di livello superiore
- **MULTIMACH_TARGET_SYS**: Identificativo del sistema target
- **PN**: Nome della ricetta
- **EXTENDPE**: Epoch: se PE non è specificato, come solitamente accade per la maggior parte delle ricette, EXTENDPE è vuoto.
- **PV**: Versione della ricetta
- **PR**: Revisione della ricetta

Ad esempio, supponiamo una Source Directory di livello superiore denominata poky, una Build Directory predefinita in poky/build e un sistema target qemux86-poky-linux. Inoltre, supponiamo che la ricetta si chiami foo_1.3.0-r0.bb. In questo caso, la directory di lavoro utilizzata dal sistema di build per compilare il pacchetto sarebbe la seguente:

```
poky/build/tmp/work/qemux86-poky-linux/foo/1.3.0-r0
```

PN

Questa variabile può avere due funzioni distinte a seconda del contesto: un nome di ricetta o un nome di pacchetto risultante.

PN si riferisce a un nome di ricetta nel contesto di un file utilizzato dal sistema di build di OpenEmbedded come input per creare un pacchetto. Il nome viene normalmente estratto dal nome del file della ricetta. Ad esempio, se la ricetta si chiama expat_2.0.1.bb, il valore di default di PN sarà "expat".

La variabile si riferisce a un nome di pacchetto nel contesto di un file creato o prodotto dal sistema di build di OpenEmbedded.

Se applicabile, la variabile PN contiene anche eventuali suffissi o prefissi speciali. Ad esempio, utilizzando bash per compilare pacchetti per la macchina nativa, PN è bash-native. Utilizzando bash per compilare pacchetti per il target e per Multilib, PN sarebbe rispettivamente bash e lib64-bash.

PR

La revisione della ricetta. Il valore di default per questa variabile è "r0". Le revisioni successive della ricetta hanno convenzionalmente i valori "r1", "r2" e così via. Quando il PV aumenta, il PR viene convenzionalmente reimpostato a "r0".

Nota

Il sistema di build di OpenEmbedded non ha bisogno dell'ausilio del PR per sapere quando ri-buildare una ricetta. Il sistema di build utilizza i checksum di input dei task insieme ai meccanismi di "stamp" [timbro] e cache di stato condivisa.

La variabile PR diventa significativa principalmente quando un gestore di pacchetti installa dinamicamente i pacchetti su un'immagine già compilata. In questo caso, il PR, che è il valore di default di PKGR, aiuta il gestore di pacchetti a distinguere quale pacchetto sia il più recente nei casi in cui molti pacchetti abbiano lo stesso PV (ad esempio, PKGV). Un componente con molti pacchetti con lo stesso PV di solito significa che tutti i pacchetti installano la stessa versione upstream, ma con pacchetti di versioni successive (PR) che includono correzioni al packaging.

Nota

Il PR non deve essere aumentato per le modifiche che non alterano il contenuto o i metadati del pacchetto.

Poiché la gestione manuale delle PR può essere macchinosa e soggetta a errori, esiste una soluzione automatizzata. Per ulteriori informazioni, consultare la sezione "[working with a pr service](#)" nello Yocto Project Development Tasks Manual.

PV

La versione della ricetta. La versione viene normalmente estratta dal nome del file della ricetta. Ad esempio, se la ricetta si chiama expat_2.0.1.bb, il valore predefinito di PV sarà "2.0.1". PV generalmente non viene sovrascritto all'interno di una ricetta, a meno che non si stia compilando una versione instabile (ad esempio, di sviluppo) da un repository di codice sorgente (ad esempio Git o Subversion).

PV è il valore predefinito della variabile PKGV.

Torizon OS

Torizon OS è una piattaforma software open source che mira a semplificare lo sviluppo e la manutenzione di software Linux embedded.

Utilizzando i container Docker, consente agli sviluppatori di concentrarsi sull'applicazione anziché creare e gestire una distribuzione personalizzata.

Questa guida spiega come creare un'immagine Docker contenente LVGL e una semplice applicazione demo che può essere distribuita su qualsiasi dispositivo Toradex con TorizonOS.

Prerequisiti

Per seguire questa guida è ovviamente necessario disporre di un SoM Toradex e di una board carrier. Ulteriori informazioni sono disponibili sul [sito web](#) di Toradex.

i Nota

È necessario avere familiarità con alcuni principi fondamentali di Docker per poter comprendere e seguire questa guida. Inoltre, Docker deve essere presente sull'host di sviluppo.

La [documentazione di Toradex](#) è una risorsa utile. Questo articolo contiene molti riferimenti ad essa.

Configurazione della scheda

Si inizia installando TorizonOS utilizzando Toradex Easy Installer. Seguire le [guide](#) ufficiali di avvio.

Nell'Easy Installer, selezionare TorizonCore come sistema operativo da installare.

Copiare l'indirizzo IP assegnato al dispositivo dal server DHCP sulla propria rete. L'indirizzo viene visualizzato nell'angolo in basso a destra. Verrà utilizzato in seguito.

Una volta completata la configurazione, il dispositivo si avvierà in TorizonOS.

Creazione dell'immagine Docker

Toradex fornisce un'estensione [VSCode](#) che offre una raccolta di template utilizzati per configurare e automatizzare le attività necessarie per la cross-compilazione delle applicazioni e la creazione di immagini Docker.

Tuttavia, questa guida spiega come eseguire manualmente tali operazioni.

i Nota

Il template per un'applicazione LVGL è attualmente in fase di aggiunta all'estensione VSCode e sarà disponibile a breve.

Per creare un container Torizon, è necessario abilitare l'emulazione Docker ARM digitando i seguenti comandi:

```
docker run --rm -it --privileged torizon/binfmt
docker run --rm -it --pull always --platform linux/arm64/v8 debian arch
```

Il secondo comando dovrebbe stampare aarch64.

Ulteriori informazioni sull'ambiente di build per i container Torizon sono disponibili [qui](#).

Iniziamo creando una directory che conterrà il [Dockerfile](#) e il codice sorgente che verrà utilizzato per creare l'immagine.

Digitare i seguenti comandi in una finestra di terminale:

```
mkdir -p ~/lvgl_torizon_os/
cd ~/lvgl_torizon_os/
touch Dockerfile

git clone --depth 1 https://github.com/lvgl/lv_port_linux
git -C lv_port_linux submodule update --init
```

Questi comandi creano la directory del progetto e il [Dockerfile](#). git viene utilizzato per scaricare i repository [lv_port_linux](#) e [lvgl](#) da Github.

Nota

Per default, `lv_port_linux` è configurato per utilizzare il dispositivo framebuffer legacy `/dev/fb0`. È anche possibile utilizzare un altro backend di rendering abilitando le opzioni corrette in `lv_port_linux/lv_conf.h`.

Ora si modifica il Dockerfile. Copiare e incollare il blocco seguente nel file:

```
ARG CROSS_SDK_BASE_TAG=3.2.1-bookworm
ARG BASE_VERSION=3.2.1-bookworm
##
## Board architecture
## arm or arm64
##
ARG IMAGE_ARCH=arm64

##
## Directory of the application inside container
##
ARG APP_ROOT=/usr/lvgl_application

# BUILD -----
FROM torizon/debian-cross-toolchain:${IMAGE_ARCH}:${CROSS_SDK_BASE_TAG} AS build

ARG APP_ROOT
ARG IMAGE_ARCH

RUN apt-get -q -y update && \
    apt-get -q -y install && \
    apt-get clean && apt-get autoremove && \
    apt-get install -q -y curl git cmake file && \
    rm -rf /var/lib/apt/lists/*

COPY . ${APP_ROOT}
WORKDIR ${APP_ROOT}

# Compile lv_port_linux
RUN CC=aaarch64-linux-gnu-gcc cmake -S ./lv_port_linux -B build
RUN make -j 4 -C ${APP_ROOT}/build

# DEPLOY -----
FROM --platform=linux/${IMAGE_ARCH} torizon/debian:${BASE_VERSION} AS deploy

ARG IMAGE_ARCH
ARG APP_ROOT

RUN apt-get -y update && apt-get install -y --no-install-recommends \
&& apt-get clean && apt-get autoremove && rm -rf /var/lib/apt/lists/*

# Copy the lvglsim executable compiled in the build step to the $APP_ROOT directory
# path inside the container
COPY --from=build ${APP_ROOT}/lv_port_linux/bin/lvglsim ${APP_ROOT}

# Command executed during runtime when the container starts
ENTRYPOINT [ "./lvglsim" ]
```

Il Dockerfile funziona come una ricetta per creare due immagini: `build` e `deploy`.

Per prima cosa scaricare i pacchetti necessari per compilare il simulatore utilizzando il gestore di pacchetti Debian `apt-get`.

Dopo la compilazione, l'eseguibile risultante viene scritto in `lv_port_linux/bin/lvglsim`.

L'immagine `deploy` verrà distribuita sul dispositivo. L'eseguibile creato nell'immagine precedente viene copiato nella directory `/usr/bin` dell'immagine corrente.

Questo crea un'immagine più piccola che non include la tool chain e le dipendenze di building.

Le immagini vengono compilate con il seguente comando:

```
docker build . -t lvgl_app
```

Docker interpreterà il Dockerfile presente nella directory di lavoro corrente. L'argomento `-t` assegna un nome all'immagine risultante.

Al termine, assicurarsi che l'immagine sia elencata da Docker:

```
docker image list | grep lvgl_app
```

Dovrebbe visualizzare l'immagine insieme al suo ID che verrà utilizzato in seguito.

Deploy dell'immagine del container sul dispositivo

L'immagine è ora pronta per essere distribuita sul dispositivo. Esistono diversi modi per eseguire questa operazione.

Leggere [questo articolo](#) per maggiori informazioni.

In questa guida, configureremo un "container registry Docker" sull'host di sviluppo, accessibile da qualsiasi dispositivo della LAN. La scheda Toradex, essendo sulla stessa rete, sarà in grado di estrarre l'immagine dal registry.

Il registry viene creato come segue:

```
docker run -d -p 5000:5000 --name registry registry:2.7
```

Il flag `-d` esegue il container in modalità "detached". L'argomento `-p` specifica il mapping delle porte. Il "registry container" ascolterà sulla porta TCP/5000 e verrà mappato esternamente alla stessa porta.

Eseguire il push dell'immagine creata nel passaggio precedente nel registry appena creato:

```
docker tag <IMAGE_ID> 127.0.0.1:5000/lvgl-app
docker push 127.0.0.1:5000/lvgl-app
```

Per default un "container registry" locale utilizza HTTP in chiaro, quindi l'istanza Docker in esecuzione sul dispositivo deve essere configurata per consentire il recupero di immagini da un repository 'non sicuro'.

Ottenere l'indirizzo IP dell'host di sviluppo e aprire una shell remota sul dispositivo.

```
sudo su # When prompted type in the password of the torizon user
# Be sure to replace set the IP address of your host instead
cat << heredoc > /etc/docker/daemon.json
{
    "insecure-registries" : ["<IP_ADDR_OF_DEVELOPMENT_HOST>:5000"]
}
heredoc

systemctl restart docker.service

# Return to regular user
exit

# Pull the image from your development host
docker pull <IP_ADDR_OF_DEVELOPMENT_HOST>:5000/lvgl-app
```

Per default, i container Docker sono isolati dall'hardware del sistema host. Sebbene questo sia ottimo per la sicurezza, un'applicazione embedded avrà ovviamente bisogno di [accesso all'hardware](#).

Il container che esegue l'applicazione LVGL necessita di accesso al framebuffer `/dev/fb0` del dispositivo.

Utilizzando l'argomento `--device` è possibile mappare un dispositivo a un container.

Avviare il container in questo modo:

```
docker run --device /dev/fb0:/dev/fb0 <IMAGE_ID>
```

Conclusione

Ora si ha un'applicazione LVGL in esecuzione. Come procedere?

- Ora si è pronti per compilare l'applicazione LVGL per Torizon OS. Si consiglia di familiarizzare con l'estensione IDE VSCode, poiché semplificherà il flusso di lavoro.

VSCode, è il modo migliore per sviluppare per Torizon OS. Se si usa un altro editor o IDE, si può sempre scrivere script per automatizzare le operazioni di compilazione/pushing/pulling.

- Leggere questo [articolo](#) per capire come progettare applicazioni per Torizon OS.
- Si consiglia inoltre di familiarizzare con Torizon Cloud e Torizon OS Builder quando ci si avvicina alla produzione.

4.3.4 Driver

Driver Framebuffer Linux

Panoramica

Il framebuffer Linux (fbdev) è un sottosistema Linux utilizzato per visualizzare la grafica. Si tratta di un'API indipendente dall'hardware che consente al software nello spazio utente di accedere al framebuffer (la parte della memoria video di un computer contenente un frame video corrente) utilizzando solo le funzionalità di base del kernel Linux e la sua interfaccia del file system del dispositivo, evitando la necessità di librerie che implementino i driver video nello spazio utente.

Prerequisiti

Il sistema dispone di un dispositivo framebuffer configurato (solitamente in /dev/fb0).

Configurazione del driver

Abilitare il supporto del driver framebuffer in lv_conf.h, tramite la definizione del compilatore cmake o tramite KConfig. È inoltre possibile configurare la modalità di rendering.

```
#define LV_USE_LINUX_FBDEV      1
#define LV_LINUX_FBOEV_RENDER_MODE  LV_DISPLAY_RENDER_MODE_PARTIAL
```

Utilizzo

Per impostare un display basato su framebuffer, creare prima un display con `lv_linux_fbdev_create`. Successivamente, si imposta il nodo del dispositivo framebuffer sul display (di solito è /dev/fb0).

```
lv_display_t *disp = lv_linux_fbdev_create();
lv_linux_fbdev_set_file(disp, "/dev/fb0");
```

Se lo schermo rimane nero o disegna solo parzialmente, si può provare ad abilitare il rendering diretto tramite `LV_DIRECT_RENDER_MODE_DIRECT`. Inoltre, si può attivare una modalità di aggiornamento forzato con `lv_linux_fbdev_set_force_refresh(true)`. Questo di solito ha un impatto sulle prestazioni e non dovrebbe essere abilitato a meno che non sia realmente necessario.

Nascondere il cursore

Si potrebbe vedere un cursore lampeggiante sullo schermo. Il metodo per nasconderlo varia a seconda della piattaforma. Ad esempio, ecco come si può fare su un Raspberry Pi:

1. Modificare il file /boot/cmdline.txt.
2. Aggiungere `vt.global_cursor_default=0`.

Errori comuni

Problema della risoluzione di default

Quando il kernel Linux si inizializza, imposta sottosistemi come il framebuffer (fbdev) per gestire l'output del display. Se è collegato un display HDMI, il kernel lo rileva e alloca una porzione di RAM come framebuffer, che contiene i dati pixel per il rendering delle immagini.

Tuttavia, un problema comune si verifica quando il display non viene acceso durante il processo di avvio. Se la scheda madre viene accesa prima dello schermo, il kernel potrebbe configurare erroneamente la risoluzione del framebuffer. Di conseguenza, sia la risoluzione visibile che quella virtuale possono essere errate, causando problemi di visualizzazione.

Questo problema si verifica spesso con le connessioni HDMI in cui il display viene acceso dopo che il sistema si è già avviato.

Il seguente comando restituisce le impostazioni correnti del dispositivo framebuffer specificato, come risoluzione, profondità di pixel e timing.

```
fbset -fb /dev/fb0
```

Per evitare problemi relativi al display, si consiglia di assicurarsi che tutti i dispositivi, incluso il display HDMI, siano collegati e accesi prima di accendere la scheda madre.

DRM

Panoramica

Il driver del display **DRM** (Direct Rendering Manager) supporta il rendering LVGL direttamente sui dispositivi framebuffer Linux tramite il sottosistema DRM/KMS. Consente di eseguire LVGL senza un sistema a finestre come X11 o Wayland, rendendolo adatto a dispositivi embedded, computer a scheda singola e applicazioni direct-to-display.

Il driver DRM interagisce direttamente con la GPU o il controller video tramite i nodi `/dev/dri/cardX`.

Introduzione a DRM

Prerequisiti

Il driver DRM richiede:

- Un sistema Linux con supporto DRM/KMS abilitato nel kernel.
- Accesso a un device node DRM, in genere `/dev/dri/card0`.
- Autorizzazioni appropriate per accedere ai dispositivi DRM (ad esempio, esecuzione come root o aggiunta dell'utente al gruppo `video`).

Su sistemi basati su Debian/Ubuntu:

```
sudo apt-get install libdrm-dev
```

Configurare il driver DRM

1. Abilitare il supporto del driver DRM in `lv_conf.h`, tramite la definizione del compilatore CMake o tramite KConfig:

```
#define LV_USE_LINUX_DRM 1
```

2. Linkare a `libdrm` durante la compilazione.

Utilizzo di Base

```
#include "lvgl/lvgl.h"
#include "lvgl/demos/lv_demos.h"

int main(void)
{
    /* Initialize LVGL */
    lv_init();

    /* DRM device node */
    const char *device = "/dev/dri/card0";

    /* Create a DRM display */
    lv_display_t *disp = lv_linux_drm_create();

    /* Set DRM device file and connector */
    /* The 2nd argument is the DRM device path */
    /* The 3rd argument is the connector_id (-1 = auto-select first available) */
    lv_linux_drm_set_file(disp, device, -1);

    /* Create demo widgets */
    lv_demo_widgets();

    /* Handle LVGL tasks */
    while (1) {
        uint32_t time_until_next = lv_timer_handler();
        if(time_until_next == LV_NO_TIMER_READY) {
            time_until_next = LV_DEF_REFR_PERIOD;
        }
        lv_delay_ms(time_until_next);
    }

    return 0;
}
```

Note

- `connector_id` specifica quale uscita video (HDMI, eDP, DP, ecc.) deve essere utilizzata. Se viene passato `-1`, il driver DRM tenterà di selezionare automaticamente il primo connettore disponibile.
- DRM richiede un'impostazione di modalità corretta. Per default, LVGL selezionerà una modalità di visualizzazione preferita.

Per evitare di dover scrivere a mano il path della scheda del dispositivo, si può chiedere a LVGL di trovarne una connessa tramite `lv_linux_drm_find_device_path()`. Restituirà la prima scheda connessa che riesce a trovare.

```
lv_display_t * disp = lv_linux_drm_create();

/* Find the first connected card in /sys/class/drm */
char * device = lv_linux_drm_find_device_path();
lv_linux_drm_set_file(disp, device, -1);
/* Free the path pointer */
lv_free(device);
```

Utilizzo di DRM con GBM

Il driver DRM può facoltativamente utilizzare **GBM** (Generic Buffer Management) per l'allocazione del buffer. Questo consente al driver di utilizzare oggetti buffer compatibili con la GPU invece di semplici framebuffer.

1. Abilitare la seguente opzione in `lv_conf.h` (o tramite Kconfig/CMake):

```
#define LV_USE_LINUX_DRM_GBM_BUFFERS 1
```

2. Linkare a `libgbm` durante la compilazione.

Quando questa opzione è abilitata:

- I buffer verranno allocati tramite GBM.
- Questo può migliorare le prestazioni e la compatibilità sulle piattaforme in cui GBM è supportato.

Utilizzo di DRM con EGL

Il driver DRM può anche essere combinato con **EGL** per il rendering con accelerazione hardware tramite EGL/GLES.

Per abilitare questa funzionalità, si impostano le seguenti opzioni in `lv_conf.h` (o tramite Kconfig/CMake):

```
#define LV_USE_LINUX_DRM 1
#define LV_USE_LINUX_DRM_GBM_BUFFERS 1
#define LV_USE_LINUX_DRM_USE_EGL 1
#define LV_USE_OPENGL_ES 1
#define LV_USE_DRAW_OPENGL_ES 1 /* optional but recommended for performance */
```

Quando `LV_LINUX_DRM_USE_EGL` è abilitato, il driver DRM inizializzerà automaticamente EGL. Non è richiesta alcuna configurazione speciale oltre all'inizializzazione DRM di base mostrata in [Utilizzo di Base](#).

Per una panoramica dettagliata dell'utilizzo e della configurazione di EGL, consultare [EGL](#).

Selezione della Modalità di Display

Nota

La selezione della modalità custom è attualmente supportata solo quando si utilizza DRM con EGL (`LV_LINUX_DRM_USE_EGL` abilitato). Quando si utilizza DRM senza EGL, il driver utilizzerà sempre la modalità di visualizzazione preferita.

Per default, il driver DRM seleziona automaticamente la modalità di visualizzazione preferita per il display connesso. Tuttavia, è possibile personalizzare questo comportamento fornendo una callback per la selezione della modalità.

Selezione della Modalità Custom

Per implementare la logica di selezione della modalità personalizzata, definire una funzione di callback e registrarla con `lv_linux_drm_set_mode_cb()`:

```
#include "lvgl/lvgl.h"

/* Custom mode selection callback */
size_t my_mode_selector(lv_display_t * disp, const lv_linux_drm_mode_t * modes, size_t mode_count)
{
    /* Example: Select the first 1920x1080@60Hz mode */
    for(size_t i = 0; i < mode_count; i++) {
        int32_t width = lv_linux_drm_mode_get_horizontal_resolution(&modes[i]);
        int32_t height = lv_linux_drm_mode_get_vertical_resolution(&modes[i]);
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

int32_t refresh = lv_linux_drm_mode_get_refresh_rate(&modes[i]);

if(width == 1920 && height == 1080 && refresh == 60) {
    return i; /* Return the index of the selected mode */
}

/* Fallback: return the first mode */
return 0;
}

int main(void)
{
    lv_init();

    lv_display_t * disp = lv_linux_drm_create();

    /* Set custom mode selection callback */
    lv_linux_drm_set_mode_cb(disp, my_mode_selector);

    lv_linux_drm_set_file(disp, "/dev/dri/card0", -1);

    /* ... rest of your application ... */
}

```

La callback riceve un array di modalità disponibili e deve restituire l'indice della modalità desiderata.

API Informazioni Modalità

Sono disponibili le seguenti funzioni per interrogare le proprietà della modalità:

- `lv_linux_drm_mode_get_horizontal_resolution()` - Ottiene la larghezza in pixel
- `lv_linux_drm_mode_get_vertical_resolution()` - Ottiene l'altezza in pixel
- `lv_linux_drm_mode_get_refresh_rate()` - Ottiene la frequenza di aggiornamento in Hz
- `lv_linux_drm_mode_is_preferred()` - Verifica se la modalità è quella preferita/nativa del display

Esempio: Selezione della modalità preferita

```

size_t select_preferred_mode(lv_display_t * disp, const lv_linux_drm_mode_t * modes, size_t mode_count)
{
    /* Find and select the preferred mode */
    for(size_t i = 0; i < mode_count; i++) {
        if(lv_linux_drm_mode_is_preferred(&modes[i])) {
            return i;
        }
    }

    /* If no preferred mode found, return the first mode */
    return 0;
}

```

Esempio: Selezione della risoluzione più alta

```

size_t select_highest_resolution(lv_display_t * disp, const lv_linux_drm_mode_t * modes, size_t mode_count)
{
    size_t best_index = 0;
    int32_t max_pixels = 0;

    for(size_t i = 0; i < mode_count; i++) {
        int32_t width = lv_linux_drm_mode_get_horizontal_resolution(&modes[i]);
        int32_t height = lv_linux_drm_mode_get_vertical_resolution(&modes[i]);
        int32_t pixels = width * height;

        if(pixels > max_pixels) {
            max_pixels = pixels;
            best_index = i;
        }
    }

    return best_index;
}

```

Note

- La callback di selezione della modalità viene chiamata prima dell'inizializzazione del display.
- Se non è impostata alcuna callback, il driver utilizza la modalità preferita per default.
- Assicurarsi che la callback restituisca sempre un indice valido (da 0 a `mode_count - 1`).
- Per ripristinare il comportamento di default, chiamare `lv_linux_drm_set_mode_cb()` con `NULL` come callback.

Driver OpenGL

Panoramica

Il driver di display **OpenGL** è un driver generico che crea texture per l'embedding di contenuti LVGL in altre applicazioni. L'obiettivo è creare texture che gli utenti possano incorporare in altre applicazioni. Il contesto OpenGL deve essere creato dall'utente oppure è possibile utilizzare GLFW o EGL come backend.

Introduzione a OpenGL

Prerequisiti

Prima di utilizzare il driver OpenGL, è necessario creare un contesto OpenGL. È possibile crearlo utilizzando:

- GLFW (vedere il [driver GLFW](#))
- EGL (vedere il [driver EGL](#))
- Gestione personalizzata del contesto OpenGL

Configurare il driver OpenGL

1. Abilitare il supporto del driver OpenGL in lv_conf.h, tramite la definizione del compilatore cmake o tramite KConfig

```
#define LV_USE_OPENGLS 1
```

Utilizzo di Base

```
#include "lvgl/lvgl.h"

#define WIDTH 640
#define HEIGHT 480

/* This flush callback works with both FULL and DIRECT render modes*/
static void flush_cb(lv_display_t *disp, const lv_area_t *area, uint8_t *px_map)
{
    if (lv_display_flush_is_last(disp)) {
        const int32_t disp_width = lv_display_get_horizontal_resolution(disp);
        const int32_t disp_height = lv_display_get_vertical_resolution(disp);
        /* The texture occupies the full screen even if 'area' is not the full screen which happens with RENDER_MODE_DIRECT */
        lv_area_t full_area;
        lv_area_set(&full_area, 0, 0, disp_width, disp_height);

        /* Get the texture id containing LVGL generated UI */
        unsigned int texture_id = lv_opengles_texture_get_texture_id(disp);
        /* This function will render to the current context */
        lv_opengles_render_texture(texture_id, &full_area, LV_OPA_COVER, disp_width, disp_height, &full_area, false, true);
    }
    lv_display_flush_ready(disp);
}

int main()
{
    /* initialize lvgl */
    lv_init();
    /* Don't forget the tick callback */

    /* NOTE: OpenGL context must be created before this point */

    /* Create a display that flushes to a texture. The OpenGL texture will be created for you */
    lv_display_t * texture = lv_opengles_texture_create(WIDTH, HEIGHT);
    /* If you already have an OpenGL texture ready, you can use it instead:
     * lv_display_t * texture = lv_opengles_texture_create_from_texture_id(WIDTH, HEIGHT, my_texture_id); */

    /* Set the display render mode and flush callback
     * lv_display_set_render_mode(texture, LV_DISPLAY_RENDER_MODE_FULL);
     * lv_display_set_flush_cb(texture, flush_cb); */

    /* get the texture ID for use in your application */
    unsigned int texture_id = lv_opengles_texture_get_texture_id(texture);

    /* create Widgets on the screen */
    lv_demo_widgets();

    while (1)
    {
        uint32_t time_until_next = lv_timer_handler();
        if(time_until_next == LV_NO_TIMER_READY) time_until_next = LV_DEF_REFR_PERIOD;
        lv_delay_ms(time_until_next);
    }
    return 0;
}
```

Renderer con Cache delle Texture OpenGL

In LVGL è presente un renderer che memorizza nella cache le aree renderizzate tramite software come texture OpenGL. Le texture vengono recuperate dalla cache e riutilizzate quando viene rilevata una corrispondenza. Nella maggior parte dei casi, le prestazioni saranno notevolmente migliorate.

```
#define LV_USE_OPENGL 1
```

Limitazioni Note

- Le prestazioni saranno le stesse o leggermente peggiori se le aree disegnate non vengono mai trovate nella cache a causa di widget con colori o forme che variano continuamente. Un esempio è un'etichetta il cui colore è impostato su un valore casuale per ogni fotogramma, come nella scena "Multiple labels" della demo del benchmark.
- I layer con pixel trasparenti e una trasparenza complessiva del layer non si fonderanno correttamente. L'effetto può essere osservato nella scena "Containers with opa_layer" della demo del benchmark negli angoli del bordo.
- I layer con rotazione non sono attualmente supportati. Le immagini con rotazione funzionano correttamente.

GLFW

Panoramica

Il driver di display/input **GLFW** offre supporto per la creazione di display LVGL e input per tastiera/mouse utilizzabili in un contesto OpenGL. Può essere utilizzato come **Wayland**, **XCB**, **SDL** o **Qt** oppure per applicazioni più embedded.

Il driver GLFW è un modo rapido per iniziare a usare piattaforme simili a PC.

Introduzione a GLFW

Prerequisiti

Il driver GLFW utilizza GLEW GLFW per accedere al gestore di finestre di OpenGL.

- Installare GLEW e GLFW: `sudo apt-get install libglew-dev libglfw3-dev`

Configurare il driver GLFW

- Librerie linkate richieste: `-lGL -lGLEW -lglfw`
- Abilitare il supporto del driver OpenGL in `lv_conf.h`, tramite la definizione del compilatore `cmake` o tramite KConfig

```
#define LV_USE_OPENGL 1
```

Utilizzo di Base

```
#include "lvgl/lvgl.h"
#include "lvgl/examples/lv_examples.h"
#include "lvgl/demos/lv_demos.h"

#define WIDTH 640
#define HEIGHT 480

int main()
{
    /* initialize lvgl */
    lv_init();

    /* create a window and initialize OpenGL */
    lv_opengl_window_t *window = lv_opengl_glfw_window_create(WIDTH, HEIGHT, true);

    /* create a display that flushes to a texture */
    lv_display_t *texture = lv_opengl_texture_create(WIDTH, HEIGHT);
    lv_display_set_default(texture);

    /* add the texture to the window */
    unsigned int texture_id = lv_opengl_texture_get_texture_id(texture);
    lv_opengl_window_texture_t *window_texture = lv_opengl_window_add_texture(window, texture_id, WIDTH, HEIGHT);
```

(continues on next page)

(continua dalla pagina precedente)

```

/* get the mouse index of the window texture */
lv_indev_t * mouse = lv_opengles_window_texture_get_mouse_indev(window_texture);

/* add a cursor to the mouse index */
LV_IMAGE_DECLARE(mouse_cursor_icon);
lv_obj_t * cursor_obj = lv_image_create(lv_screen_active());
lv_image_set_src(cursor_obj, &mouse_cursor_icon);
lv_indev_set_cursor(mouse, cursor_obj);

/* create Widgets on the screen */
lv_demo_widgets();

while (1)
{
    uint32_t time_until_next = lv_timer_handler();
    if(time_until_next == LV_NO_TIMER_READY) time_until_next = LV_DEF_REFR_PERIOD; /*handle LV_NO_TIMER_READY. Another option is to `sleep` for longer*/
    lv_delay_ms(time_until_next);
}

return 0;
}

```

Utilizzo Avanzato

Il driver GLFW può disegnare le texture dall'utente. È possibile utilizzare una libreria di terze parti per aggiungere contenuto a una texture e il driver disegnerà la texture nella finestra.

```

#include "lvgl/lvgl.h"
#include <GL/glew.h>
#include <GLFW/glfw3.h>

#define WIDTH 640
#define HEIGHT 480

void custom_texture_example(void)
{
    //*****
    * MAIN WINDOW
    *****/

    /* create a window and initialize OpenGL */
    /* multiple windows can be created */
    lv_opengles_window_t * window = lv_opengles_glfw_window_create(WIDTH, HEIGHT, true);

    //*****
    * OPTIONAL MAIN TEXTURE
    *****/

    /* create a main display that flushes to a texture */
    lv_display_t * main_texture = lv_opengles_texture_create(WIDTH, HEIGHT);
    lv_display_set_default(main_texture);

    /* add the main texture to the window */
    unsigned int main_texture_id = lv_opengles_texture_get_texture_id(main_texture);
    lv_opengles_window_texture_t * window_main_texture = lv_opengles_window_add_texture(window, main_texture_id, WIDTH, HEIGHT);

    /* get the mouse index of this main texture */
    lv_indev_t * main_texture_mouse = lv_opengles_window_texture_get_mouse_indev(window_main_texture);

    /* add a cursor to the mouse index */
    LV_IMAGE_DECLARE(mouse_cursor_icon);
    lv_obj_t * cursor_obj = lv_image_create(lv_screen_active());
    lv_image_set_src(cursor_obj, &mouse_cursor_icon);
    lv_indev_set_cursor(main_texture_mouse, cursor_obj);

    /* create Widgets on the screen of the main texture */
    lv_demo_widgets();

    //*****
    * ANOTHER TEXTURE
    *****/

    /* create a sub display that flushes to a texture */
    const int32_t sub_texture_w = 300;
    const int32_t sub_texture_h = 300;
    lv_display_t * sub_texture = lv_opengles_texture_create(sub_texture_w, sub_texture_h);

    /* add the sub texture to the window */
    unsigned int sub_texture_id = lv_opengles_texture_get_texture_id(sub_texture);
    lv_opengles_window_texture_t * window_sub_texture = lv_opengles_window_add_texture(window, sub_texture_id, sub_texture_w, sub_texture_h);

    /* create Widgets on the screen of the sub texture */
    lv_display_set_default(sub_texture);
    lv_example_keyboard_2();
    lv_display_set_default(main_texture);

    /* position the sub texture within the window */
    lv_opengles_window_texture_set_x(window_sub_texture, 250);
    lv_opengles_window_texture_set_y(window_sub_texture, 150);

    /* optionally change the opacity of the sub texture */
    lv_opengles_window_texture_set_opa(window_sub_texture, LV_OPA_80);

    //*****
    * USE AN EXTERNAL OPENGL TEXTURE IN LVGL
    *****/

    unsigned int external_texture_id;
    glGenTextures(1, &external_texture_id);
    glBindTexture(GL_TEXTURE_2D, external_texture_id);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
    glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
    LV_IMAGE_DECLARE(img_cogwheel_argb);

```

(continues on next page)

(continua dalla pagina precedente)

```

#ifndef LV_COLOR_DEPTH
    const int texture_format = GL_RGB;
#elif LV_COLOR_DEPTH == 16
    const int texture_format = GL_RGB65;
#elif LV_COLOR_DEPTH == 24
    const int texture_format = GL_RGB;
#elif LV_COLOR_DEPTH == 32
    const int texture_format = GL_RGBA;
#else
#error("Unsupported color format")
#endif

glTexImage2D(GL_TEXTURE_2D, 0, texture_format, img_cogwheel_argb.header.w, img_cogwheel_argb.header.h, 0, GL_BGRA, GL_UNSIGNED_BYTE, img_cogwheel_argb.data);
glGenerateMipmap(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, 0);

/* add the external texture to the window */
lv_opengles_window_texture_t *window_external_texture = lv_opengles_window_add_texture(window, external_texture_id, img_cogwheel_argb.header.w, img_cogwheel_argb.header.h);

/* set the position and opacity of the external texture within the window */
lv_opengles_window_texture_set_x(window_external_texture, 20);
lv_opengles_window_texture_set_y(window_external_texture, 20);
lv_opengles_window_texture_set_opa(window_external_texture, LV_OPA_70);

/*
 * USE AN LVGL TEXTURE IN ANOTHER LIBRARY
 */
lv_ref_now(sub_texture);

/* the texture is drawn on by LVGL and can be used by anything that uses OpenGL textures */
third_party_lib_use_texture(sub_texture_id);
}

```

Miglioramento delle Prestazioni

In LVGL è presente un renderer che memorizza nella cache le aree renderizzate tramite software come texture OpenGL. Vedere [Renderer con Cache delle Texture OpenGL](#) per saperne di più.

EGL

Panoramica

Il driver **EGL** fornisce supporto per la creazione di display LVGL utilizzando l'API EGL (Embedded-System Graphics Library). EGL è un'API di livello inferiore, più strettamente legata ai driver sottostanti della piattaforma. Il supporto OpenGL in LVGL è progettato per essere portabile tra diverse API. Attualmente, sono supportati GLFW ed EGL. L'utilizzo di EGL richiede un'ulteriore integrazione con la piattaforma.

`lv_opengles_egl_window_create()` può essere utilizzato per creare una `lv_opengles_window_t` utilizzabile con le stesse API OpenGL generiche LVGL di una finestra GLFW.

EGL con DRM

EGL può essere utilizzato insieme al driver DRM per il rendering con accelerazione hardware.

Quando `LV_LINUX_DRM_USE_EGL` è abilitato, il driver DRM configurerà automaticamente EGL. Non è richiesta alcuna inizializzazione aggiuntiva oltre alla normale configurazione DRM.

Consultare [DRM](#) per la configurazione e un esempio di utilizzo di base.

EGL senza DRM (Sperimentale)



Questa funzionalità è sperimentale e l'API è privata. Si prevedono cambiamenti significativi.

Per utilizzare EGL senza essere vincolati a DRM, si può abilitare `LV_USE_EGL` utilizzando una definizione del compilatore. Questa API è attualmente privata e sperimentale e si attendono cambiamenti significativi.

```
# Enable standalone EGL (experimental)
-DLV_USE_EGL=1
```

Questo consente di utilizzare EGL con la propria gestione del contesto o altre piattaforme, ma l'API potrebbe cambiare senza preavviso nelle versioni future.

Miglioramento delle Prestazioni

In LVGL è presente un renderer che memorizza nella cache le aree renderizzate tramite software come texture OpenGL. Vedere [Renderer con Cache delle Texture OpenGL](#) per saperne di più.

Driver per Display/Input Wayland

Panoramica

Il **driver Wayland** offre supporto per la simulazione del display LVGL e degli input da tastiera/mouse in una finestra del desktop.

È un'alternativa a **X11** o a **SDL2**

Lo scopo principale di questo driver è testare/debuggare l'applicazione LVGL. Può anche essere utilizzato per eseguire applicazioni in "modalità chiosco".

Dipendenze

Il driver Wayland richiede alcune dipendenze.

Su Ubuntu

```
sudo apt-get install libwayland-dev libxkbcommon-dev libwayland-bin wayland-protocols
```

Su Fedora

```
sudo dnf install wayland-devel libxkbcommon-devel wayland-utils wayland-protocols-devel
```

Configurazione del driver Wayland

- Abilitare il driver Wayland in `lv_conf.h`

```
#define LV_USE_WAYLAND 1
```

- Opzioni di configurazione facoltative:

Alcune impostazioni opzionali dipendono dall'abilitazione del supporto del buffer DMA (`LV_WAYLAND_USE_DMABUF`). La tabella seguente riassume le combinazioni valide e le relative limitazioni:

Tabella 2: Possibilità di configurazione

Opzione di configurazione	Senza DMABUF	Con DMABUF
<code>LV_DRAW_USE_G2D</code>	Non richiesto	Obbligatorio
<code>LV_WAYLAND_BUF_COUNT</code>	1	1 o 2
<code>LV_WAYLAND_RENDER_MODE</code>	<code>LV_DISPLAY_RENDER_MODE_PARTIAL</code>	<code>LV_DISPLAY_RENDER_MODE_DIRECT</code> o <code>LV_DISPLAY_RENDER_MODE_FULL</code>
<code>LV_WAYLAND_WINDOW_DECORATIONS</code>	1 o 0	0

Note aggiuntive

- Il supporto DMABUF (`LV_WAYLAND_USE_DMABUF`) migliora le prestazioni e abilita più modalità di rendering, ma presenta requisiti e restrizioni specifici.
- `LV_WAYLAND_WINDOW_DECORATIONS` è richiesto solo per alcuni compositori (ad esempio, GNOME/Mutter o Weston).

Esempio

Un simulatore di esempio è disponibile in questo [repo](#)

Utilizzo

1. In `main.c` `#include "lv_drivers/wayland/wayland.h"`
2. Abilitare il driver Wayland in `lv_conf.h` con `LV_USE_WAYLAND 1`
3. `LV_COLOR_DEPTH` dovrebbe essere impostato su 32 o su 16 in `lv_conf.h`
4. Aggiungere un display utilizzando `lv_wayland_window_create()`, possibilmente con una callback di chiusura per tracciare lo stato di ciascun display:

```
#define H_RES (800)
#define V_RES (480)

/* Create a display */
lv_disp_t * disp = lv_wayland_window_create(H_RES, V_RES, "Window Title", close_cb);
```

Come parte della chiamata precedente, il driver Wayland registrerà quattro dispositivi di input per ciascun display:

- un KEYPAD connesso agli eventi della tastiera Wayland
- un POINTER connesso agli eventi touch Wayland
- un POINTER connesso agli eventi del puntatore Wayland
- un ENCODER connesso agli eventi degli assi del puntatore Wayland

Gli handle per i dispositivi di input di ciascun display possono essere ottenuti utilizzando rispettivamente `lv_wayland_get_indev_keyboard()`, `lv_wayland_get_indev_touchscreen()`, `lv_wayland_get_indev_pointer()` e `lv_wayland_get_indev_pointeraxis()`.

Modalità fullscreen

Per impostare la finestra a schermo intero tramite programmazione, utilizzare la funzione `lv_wayland_window_setFullscreen()` rispettivamente con `true` o `false` per l'argomento `fullscreen`.

Assegnazione del display fisico

Quando si utilizzano più display fisici, è possibile controllare su quale display viene visualizzata una finestra a schermo intero assegnandola a un display fisico specifico prima di entrare in modalità "fullscreen".

Utilizzare `lv_wayland_assign_physical_display()` per assegnare una finestra a un particolare display fisico, dove il parametro `display` specifica il numero del display fisico (in genere 0, 1, 2, ecc.):

```
/* Assign to physical display 0 and full screen on that display */
lv_wayland_assign_physical_display(display, 0);
lv_wayland_window_setFullscreen(window, true);
```

Per rimuovere l'assegnazione del display fisico e ripristinare il comportamento di default, si usa `lv_wayland_unassign_physical_display()`:

```
lv_wayland_unassign_physical_display(display);
```

Modalità massimizzata

Per massimizzare la finestra tramite programmazione, utilizzare la funzione `lv_wayland_window_setMaximized()` rispettivamente con `true` o `false` per l'argomento `maximized`.

Gestore timer personalizzato

Chiamare sempre `lv_wayland_timer_handler()` nel proprio loop del timer invece del normale `lv_timer_handler()`.

Nota: `lv_wayland_timer_handler()` chiama internamente `lv_timer_handler()`

Questo consente al client Wayland di funzionare correttamente su Weston; il ridimensionamento dei buffer di memoria condivisa durante un commit non funziona correttamente su Weston.

Il wrapping della chiamata a `lv_timer_handler()` è necessario per avere un maggiore controllo su quando viene chiamata la callback di flush LVGL.

Building del driver Wayland

Un simulatore di esempio è disponibile in questo [repo](#)

Se è necessario utilizzare il driver con un altro sistema di build. I file sorgente e gli header per la shell XDG devono essere generati dalle definizioni per il protocollo della shell XDG.

Nell'esempio, CMake viene utilizzato per eseguire l'operazione richiamando l'utilità `wayland-scanner`

Per eseguire questa operazione manualmente,

Assicurarsi che le dipendenze elencate all'inizio dell'articolo siano installate.

Il protocollo Wayland è definito utilizzando file XML presenti in `/usr/share/wayland-protocols`

Per generare i file richiesti, eseguire i seguenti comandi:

```
wayland-scanner client-header $SYSROOT/usr/share/wayland-protocols/stable/xdg-shell/xdg-shell.xml wayland_xdg_shell.h
wayland-scanner private-code $SYSROOT/usr/share/wayland-protocols/stable/xdg-shell/xdg-shell.xml wayland_xdg_shell.c
```

Quando `LV_WAYLAND_USE_DMABUF` è impostato a 1, devono essere generati anche i seguenti protocoli:

```
wayland-scanner client-header $SYSROOT/usr/share/wayland-protocols/stable/linux-dmabuf/linux-dmabuf-v1.xml wayland_linux_dmabuf.h
wayland-scanner private-code $SYSROOT/usr/share/wayland-protocols/stable/linux-dmabuf/linux-dmabuf-v1.xml wayland_linux_dmabuf.c
```

I file risultanti possono quindi essere integrati nel progetto. È consigliabile eseguire nuovamente `wayland-scanner` su ogni build per garantire che vengano generate le versioni corrette, che devono corrispondere alla versione della libreria `wayland-client` linkata dinamicamente installata sul sistema.

Stato attuale e obiettivi

- Aggiungere la modalità di rendering diretto
- Refactoring delle integrazioni della shell per evitare un'eccessiva compilazione condizionale
- Tecnicamente, il driver Wayland consente di creare più finestre, ma questa funzionalità è sperimentale.
- Aggiungere in futuro un supporto avanzato per la shell XDG per consentire la creazione di applicazioni desktop su piattaforme Unix-like, in modo simile a quanto avviene con il driver Win32.
- Aggiungere il supporto per Mesa; attualmente viene utilizzato `wl_shm`, ma non è la tecnica più efficace.

Segnalazioni di bug

Il driver Wayland è attualmente in fase di sviluppo, segnalazioni di bug, contributi e feedback sono sempre benvenuti.

È tuttavia importante creare problemi dettagliati quando si riscontra un problema; log e screenshot del problema sono di grande aiuto.

Abilitare `LV_USE_LOG` e avviare l'eseguibile del simulatore in questo modo.

```
WAYLAND_DEBUG=1 ./path/to/simulator_executable > /tmp/debug 2>&1
```

Questo creerà un file di log chiamato `debug` nella directory `/tmp`, eseguire il copia e incolla del contenuto del file nell'issue su GitHub. Il file di log contiene i log LVGL e i messaggi Wayland.

Assicurarsi di replicare il problema rapidamente, altrimenti i log diventano troppo grandi.

Driver Display/Input X11

Panoramica

Il **driver Display/Input X11** offre supporto per la simulazione del display LVGL e degli input da tastiera/mouse in una finestra del desktop X11.

È un'alternativa a **Wayland, XCB, SDL o Qt**.

Lo scopo principale di questo driver è testare/eseguire il debug dell'applicazione LVGL in una finestra di simulazione **Linux**.

Prerequisiti

Il driver X11 utilizza XLib per accedere al gestore delle finestre Linux.

1. Installare XLib: `sudo apt-get install libx11-6` (dovrebbe essere già installato)
2. Installare il pacchetto di sviluppo XLib: `sudo apt-get install libx11-dev`

Configurare il driver X11

1. **Abilitare il supporto del driver X11 in lv_conf.h, tramite la definizione del compilatore cmake o tramite KConfig**

```
#define LV_USE_X11 1
```

2. **Opzioni di configurazione facoltative:**

- **Direct Exit**

```
#define LV_X11_DIRECT_EXIT 1 /* preferred default - ends the application automatically if last window has been closed */
// or
#define LV_X11_DIRECT_EXIT 0 /* application is responsible for ending the application (e.g. by own LV_EVENT_DELETE handler */
```

- **Double buffering**

```
#define LV_X11_DOUBLE_BUFFER 1 /* preferred default */
// or
#define LV_X11_DOUBLE_BUFFER 0 /* not recommended */
```

- **Render mode**

```
#define LV_X11_RENDER_MODE_PARTIAL 1 /* LV_DISPLAY_RENDER_MODE_PARTIAL, preferred default */
// or
#define LV_X11_RENDER_MODE_DIRECT 1 /* LV_DISPLAY_RENDER_MODE_DIRECT, not recommended for X11 driver */
// or
#define LV_X11_RENDER_MODE_DULL 1 /* LV_DISPLAY_RENDER_MODE_FULL, not recommended for X11 driver */
```

Utilizzo

Inizializzazione minima che apre una finestra e abilita il supporto per tastiera/mouse
(ad esempio, in main.c, `LV_X11_DIRECT_EXIT` dev'essere 1):

```
int main(int argc, char ** argv)
{
    ...

    /* initialize X11 display driver */
    lv_display_t * disp = lv_x11_window_create("LVGL X11 Simulation", monitor_hor_res, monitor_ver_res);

    /* initialize X11 input drivers (for keyboard, mouse & mousewheel) */
    lv_x11_inputs_create(disp, NULL);

    ...

    while(true)
    {
        ...

        /* Periodically call the lv_timer handler */
        lv_timer_handler();
    }
}
```

Inizializzazione completa con simbolo del puntatore del mouse e gestione dell'uscita dall'applicazione (dipendente da LV_X11_DIRECT_EXIT (può essere 1 o 0))

```
bool terminated = false;

#if !LV_X11_DIRECT_EXIT
static void on_close_cb(lv_event_t * e)
{
    ...
    terminate = true;
}
#endif

int main(int argc, char ** argv)
{
    ...

    /* initialize X11 display driver */
    lv_display_t * disp = lv_x11_window_create("LVGL X11 Simulation", monitor_hor_res, monitor_ver_res);
    lv_display_add_event_cb(disp, on_close_cb, LV_EVENT_DELETE, disp);

    /* initialize X11 input drivers (for keyboard, mouse & mousewheel) */
    LV_IMAGE_DECLARE(my_mouse_cursor_icon);
    lv_x11_inputs_create(disp, &my_mouse_cursor_icon);

    #if !LV_X11_DIRECT_EXIT
    /* set optional window close callback to enable application cleanup and exit */
    lv_x11_window_set_close_cb(disp, on_close_cb, disp);
    #endif

    ...

    while(!terminated)
    {
        ...
        /* Periodically call the lv_timer handler */
        lv_timer_handler();
    }
}
```

evdev

Panoramica

"Event device" (evdev) di Linux, è un'API indipendente dall'hardware che consente l'accesso agli eventi di input, ad esempio da un mouse o da un touchscreen. Viene esposto tramite l'interfaccia del file system del device Linux.

Prerequisiti

Il sistema dispone di un dispositivo di input configurato (solitamente in /dev/input/, ad esempio /dev/input/event0).

Configurazione del driver

Abilitare il supporto del driver evdev LVGL Linux in lv_conf.h.

```
#define LV_USE_EVDEV 1
```

Utilizzo

Per impostare un input di evento, creare prima un "input device" con lv_eudev_create impostandolo sul dispositivo di eventi Linux corretto. Quindi collegarlo al display LVGL con lv_indev_set_display.

```
lv_indev_t *touch = lv_eudev_create(LV_INDEV_TYPE_POINTER, "/dev/input/event0");
lv_indev_set_display(touch, disp);
```

Assicurarsi che un oggetto lv_display_t sia già stato creato per disp. Un esempio è mostrato di seguito, utilizzando il driver framebuffer Linux.

```
lv_display_t * disp = lv_linux_fbdev_create();
lv_linux_fbdev_set_file(disp, "/dev/fb0");
```

Individuazione del dispositivo di input

Se non si riesce a determinare il dispositivo di input, eseguire prima

```
`$cat /proc/bus/input/devices`
```

Questo dovrebbe mostrare i dispositivi di input e ci saranno voci con la parola event che forniscono un indizio sul dispositivo da utilizzare, ad esempio event1 sarebbe /dev/input/event1.

Si può usare evtest per visualizzare i dati da quella sorgente di eventi e verificare se è effettivamente quella desiderata.

Provare:

```
$evtest /dev/input/event1 sostituendo eventX con il dispositivo di eventi indicato sopra.
```

Rilevamento automatico dei dispositivi di input

È disponibile il supporto per la ricerca e l'aggiunta automatica di dispositivi di input in /dev/input/. I nuovi dispositivi verranno aggiunti automaticamente quando vengono connessi. Per abilitare questa funzionalità, è sufficiente chiamare `lv_evdev_discovery_start(NULL, NULL)`.

Si potrebbe voler reagire all'aggiunta di un nuovo dispositivo in modo che possa essere applicata un'immagine del cursore, ad esempio. È possibile fornire una funzione di callback che verrà richiamata quando viene aggiunto un nuovo dispositivo.

```
#include "lvgl/src/core/lv_global.h"

static void indev_deleted_cb(lv_event_t * e)
{
    if(LV_GLOBAL_DEFAULT()>deinit_in_progress) return;
    lv_obj_t * cursor_obj = lv_event_get_user_data(e);
    lv_obj_delete(cursor_obj);
}

static void discovery_cb(lv_indev_t * indev, lv_evdev_type_t type, void * user_data)
{
    LV_LOG_USER("new '%s' device discovered", type == LV_EVDEV_TYPE_REL ? "REL" :
    type == LV_EVDEV_TYPE_ABS ? "ABS" :
    type == LV_EVDEV_TYPE_KEY ? "KEY" :
    "unknown");

    if(type == LV_EVDEV_TYPE_REL) {
        LV_IMAGE_DECLARE(mouse_cursor_icon);
        lv_obj_t * cursor_obj = lv_image_create(lv_screen_active());
        lv_image_set_src(cursor_obj, &mouse_cursor_icon);
        lv_indev_set_cursor(indev, cursor_obj);
        lv_indev_add_event_cb(indev, indev_deleted_cb, LV_EVENT_DELETE, cursor_obj);
    }
}

int main()
{
    /* ... */
    lv_evdev_discovery_start(discovery_cb, NULL);
    /* ... */
}
```

Al momento della stesura di questo documento, questa funzionalità non è supportata in BSD.

libinput

Panoramica

Libinput è uno stack di input per processi che devono fornire eventi da dispositivi di input di uso comune. Tra questi rientrano mouse, tastiere, touchpad, touchscreen e tavolette grafiche. Libinput gestisce le peculiarità specifiche del dispositivo e fornisce un'API di facile utilizzo per ricevere eventi dai dispositivi.

Prerequisiti

È installata la versione di sviluppo di libinput (di solito `libinput-dev`). Se il dispositivo di input richiede peculiarità, assicurarsi che siano installate (di solito in `/usr/share/libinput/*.quirks`). Per verificare se il dispositivo è configurato correttamente per l'utilizzo con libinput, è possibile eseguire `libinput list-devices`.

```
$ sudo libinput list-devices
...
Device:      ETPS/2 Elantech Touchpad
Kernel:     /dev/input/event5
Group:      10
Seat:       seat0, default
Size:       102x74mm
```

(continues on next page)

(continua dalla pagina precedente)

```
Capabilities: pointer gesture
Tap-to-click: disabled
Tap-and-drag: enabled
...
```

Se il dispositivo non venisse visualizzato, potrebbe essere necessario configurare udev e le regole udev appropriate per connetterlo.

Inoltre, per il supporto completo della tastiera, incluse lettere e modificatori, è necessario installare la versione di sviluppo di libxkbcommon (solitamente `libxkbcommon-dev`).

Configurazione del driver

Abilitare il supporto del driver libinput in `lv_conf.h`, tramite la definizione del compilatore `cmake` o tramite KConfig.

```
#define LV_USE_LIBINPUT 1
```

Il supporto completo della tastiera deve essere abilitato separatamente.

```
#define LV_LIBINPUT_XKB 1
#define LV_LIBINPUT_XKB_KEY_MAP { .rules = NULL, .model = "pc101", .layout = "us", .variant = NULL, .options = NULL }
```

Per trovare i valori corretti per la mappatura dei tasti, è possibile utilizzare il comando `setxkbmap -query`.

Utilizzo

Per impostare un dispositivo di input tramite il driver libinput, è sufficiente chiamare `lv_libinput_create` con il rispettivo tipo di dispositivo (LV_INDEV_TYPE_POINTER o LV_INDEV_TYPE_KEYPAD) e il path del nodo del dispositivo (ad esempio `/dev/input/event5`).

```
lv_indev_t *indev = lv_libinput_create(LV_INDEV_TYPE_POINTER, "/dev/input/event5");
```

Si noti che i touchscreen vengono trattati come dispositivi di puntamento (assoluti) dal driver libinput e richiedono LV_INDEV_TYPE_POINTER.

A seconda del sistema, i path dei nodi dei dispositivi potrebbero non essere stabili dopo i riavvii. In questo caso, è possibile utilizzare `lv_libinput_find_dev` per trovare il primo dispositivo con una funzionalità specifica.

```
char *path = lv_libinput_find_dev(LV_LIBINPUT_CAPABILITY_TOUCH, true);
```

Il secondo argomento controlla se tutti i dispositivi vengono nuovamente scansionati o meno. Se si hanno molti dispositivi collegati, l'operazione può risultare piuttosto lenta. Pertanto, è consigliabile specificare `true` solo alla prima chiamata quando si richiama questo metodo più volte di seguito. Per trovare tutti i dispositivi con una funzionalità specifica, utilizzare `lv_libinput_find_devs`.

Per collegare una tastiera a un'area di testo, creare un gruppo di input dedicato e impostarlo sia sull'indev che sulla textarea.

```
lv_obj_t *textarea = lv_textarea_create(...);
...
lv_group_t *keyboard_input_group = lv_group_create();
lv_indev_set_group(indev, keyboard_input_group);
lv_group_add_obj(keyboard_input_group, textarea);
```

4.4 Supporto per RTOS

4.4.1 FreeRTOS

TODO [da fare]

4.4.2 MQX RTOS

Consultare la homepage di MQX RTOS

TODO [da fare]

4.4.3 NuttX RTOS

Cos'è NuttX?

NuttX è un sistema operativo real-time (RTOS) maturo e sicuro, con particolare attenzione alla conformità agli standard tecnici e alle dimensioni ridotte. È scalabile da microcontrollori e microprocessori da 8 bit a 64 bit ed è conforme agli standard Portable Operating System Interface (POSIX) e American National Standards Institute (ANSI), nonché a molti sottosistemi simili a Linux. Il modo migliore per pensare a NuttX è considerarlo come un piccolo Unix/Linux per microcontrollori.

Punti di Forza di NuttX

- **Piccolo** - Si adatta e funziona su microcontrollori con soli 32 kB di Flash e 8 kB di RAM.
- **Conforme** - Si impegna a essere il più compatibile possibile con POSIX e Linux.
- **Versatile** - Supporta numerose architetture (ARM, ARM Thumb, AVR, MIPS, OpenRISC, RISC-V a 32 e 64 bit, RX65N, x86-64, Xtensa, Z80/Z180, ecc.).
- **Modulare** - Il suo design modulare consente agli sviluppatori di selezionare solo ciò che conta davvero e di utilizzare i moduli per includere nuove funzionalità.
- **Popolare** - NuttX è utilizzato da molte aziende in tutto il mondo. Probabilmente è già stato utilizzato un prodotto con NuttX senza sapere che eseguiva NuttX.
- **Prevedibile** - NuttX è un kernel "Realtime preemptible", quindi lo si può usare per creare applicazioni prevedibili per il controllo in tempo reale.

Perché NuttX + LVGL?

Sebbene NuttX abbia una propria libreria grafica chiamata [NX](#), LVGL è una buona alternativa perché gli utenti potrebbero trovare demo più accattivanti e riutilizzare il codice di progetti precedenti. LVGL è una libreria GUI di alto livello [Basata su Componenti Object-Oriented](#), che potrebbe adattarsi perfettamente a un RTOS con funzionalità avanzate come NuttX. LVGL è implementato in C e le sue API sono in C.

Ecco alcuni vantaggi dell'utilizzo di LVGL in NuttX

- Sviluppare prima la GUI in Linux e, una volta completata, compilarla per NuttX. Niente di più, nessuna perdita di tempo.
- Di solito, lo sviluppo di GUI per RTOS di basso livello richiede più iterazioni per ottenere il risultato desiderato, dove ogni iterazione consiste in **Modifica del codice > Build > Flash > Run**. Utilizzando LVGL, Linux e NuttX si può ridurre questo processo e testare tutto sul computer e, una volta completato, compilarlo su NuttX e il gioco è fatto.

NuttX + LVGL potrebbero essere utilizzati per

- Delle demo della GUI per dimostrare le capacità grafiche della scheda.
- Prototipazione rapida della GUI per la presentazione MVP (Minimum Viable Product).
- Visualizzazione dei dati dei sensori direttamente e facilmente sulla scheda senza utilizzare un computer.
- Prodotti finali con una GUI senza touchscreen (ad esempio, un'interfaccia per stampante 3D che utilizza un encoder rotativo per l'inserimento dei dati).
- Prodotti finali con touchscreen (e tutti i tipi di ammennicoli).

Come iniziare con NuttX e LVGL?

Ci sono molte board nella [NuttX mainline](#) con supporto per LVGL. Usiamo la [STM32F429IDISCOVERY](#) come esempio, perché è una scheda molto popolare.

Per prima cosa si devono installare i prerequisiti sul sistema.

Utilizziamo Windows Subsystem for Linux

```
$ sudo apt-get install automake bison build-essential flex gcc-arm-none-eabi gperf git libncurses5-dev libtool libusb-dev libusb-1.0.0-dev pkg-config kconfig-frontends openocd
```

Ora creiamo un'area di lavoro in cui salvare i nostri file.

```
$ mkdir ~/nuttxspace
$ cd ~/nuttxspace
```

Cloniamo i repository NuttX e Apps.es

```
$ git clone https://github.com/apache/incubator-nuttx nuttx
$ git clone https://github.com/apache/incubator-nuttx-apps apps
```

Simulatore NuttX

Il simulatore NuttX consente di eseguire NuttX e LVGL sul PC. Questo può essere particolarmente utile per sviluppare interfacce utente LVGL con feedback visivo immediato e anche per sviluppare app NuttX indipendenti dalla piattaforma.

Configurare NuttX per utilizzare la board sim e LVGL Demo

```
$ ./tools/configure.sh sim:lvgl_fb
$ make
```

Se tutto è andato bene, si avrà il file nuttx

```
$ ls -l nuttx
-rwxr-xr-x 1 root root 2819920 May 12 15:01 nuttx
```

Esecuzione del Simulatore NuttX

```
$ ./nuttx
```

Utilizzando il terminale NSH> avviare la demo LVGL:

```
nsh> lvgldemo
```

NuttX On-Device

Se non si ha una STM32F429I-DISCO, questa procedura sarà simile per altre schede.

Configurare NuttX per utilizzare la scheda stm32f429i-disco e la demo LVGL

```
$ ./tools/configure.sh stm32f429i-disco:lvgl
$ make
```

Se tutto andasse a buon fine, si dovrebbe avere il file nuttx.bin da flashare sulla scheda:

```
$ ls -l nuttx.bin
-rwxrwxr-x 1 alan alan 287144 Jun 27 09:26 nuttx.bin
```

Flashare il firmware sulla scheda usando OpenOCD

```
$ sudo openocd -f interface/stlink-v2.cfg -f target/stm32f4x.cfg -c init -c "reset halt" -c "flash write_image erase nuttx.bin 0x08000000"
```

Resetare la scheda e, usando il terminale 'NSH>', avviare la demo LVGL:

```
nsh> lvgldemo
```

Utilizzo Personalizzato

L'app di esempio chiamata `lvgldemo` usa l'integrazione NuttX di LVGL per eseguire le demo in NuttX.

<https://github.com/apache/nuttx-apps/tree/master/examples/lvgldemo>

Viene utilizzata da tutte le configurazioni di LVGL. Vedere `sim:lvgl_fb` e `sim:lvgl_lcd`.

La si può usare come riferimento per aggiungere LVGL alla propria app NuttX. Iniziamo copiando il contenuto di `lvgldemo.c`. Gestisce i driver dei display basati su fb e lcd, il driver di input touchscreen e libuv.

Supporto Driver NuttX

- **fbdev** - LVGL può creare un display per i dispositivi `/dev/fb*`.
- **lcd** - LVGL può creare un display per i dispositivi `/dev/lcd*`.
- **input/uinput touchscreen** - LVGL può creare uno o più indev per dispositivi touch che utilizzano il sistema input/uinput NuttX.
- **libuv** - LVGL supporta l'utilizzo di libuv come "event loop manager". Le app libuv possono integrarsi perfettamente.

Non è ancora disponibile il supporto per le tastiere input/uinput. Il driver della tastiera sim NuttX X11 emette codici tasto non standard, quindi l'integrazione di LVGL con esso non sarebbe generica, se fosse aggiunto.

Altre integrazioni

- Integrazione del profiler LVGL+NuttX
- Heap opzionale della cache immagini dedicato

Configurazioni

La libreria LVGL è una "app" di NuttX. È possibile configurare LVGL in `menuconfig`.

```
$ ./tools/configure.sh stm32f429i-disco:lvgl
$ make menuconfig
```

Il file Kconfig di LVGL fornisce le opzioni di configurazione disponibili per il sistema di build di NuttX. Le configurazioni LVGL in `menuconfig` si trovano in **Application Configuration > Graphics Support > Light and Versatile Graphic Library (LVGL) > LVGL configuration**.

Per trovare la posizione di una configurazione specifica, premere il tasto "/" per aprire un'interfaccia di ricerca.

Ecco alcune configurazioni utilizzabili per personalizzare la configurazione di NuttX e LVGL:

- **LV_USE_NUTTX_INDEPENDENT_IMAGE_HEAP** - È possibile abilitare o disabilitare l'heap delle immagini LVGL in NuttX. Per default, è disabilitato. Se lo si abilita, LVGL utilizzerà invece l'heap NuttX.

Utilizzare `LV_STDLIB_CLIB` per `LV_USE_STDLIB_MALLOC`, `LV_USE_STDLIB_STRING` o `LV_USE_STDLIB_SPRINTF` è ragionevole, poiché NuttX implementa completamente queste API della libreria standard. La scelta di utilizzare `malloc` di NuttX dipende dal fatto che si desideri che LVGL allochi dall'heap globale di NuttX o utilizzi il proprio.

Dov'è `LV_OS_NUTTX`?

NuttX cerca di essere compatibile con POSIX ove possibile, il che significa che supporta i pthread (thread POSIX). Per abilitare le funzionalità del sistema operativo in LVGL su NuttX, impostare `LV_USE_OS` a `LV_OS_PTH-READ`. Il motivo principale per cui si abilitano le funzionalità del sistema operativo è il rendering multi-core. Vedere `LV_DRAW_SW_DRAW_UNIT_CNT`. Altrimenti non è necessario impostarlo.

Utilizzo di una Versione Specifica di LVGL

Ecco i passaggi per utilizzare l'ultima versione (o qualsiasi) di LVGL in NuttX.

Assicurarsi innanzitutto che gli artefatti di build siano stati cancellati. Questo rimuoverà il file LVGL e lo ZIP decompressi esistenti, se presenti.

```
$ make distclean
```

Ora si può usare `git clone` per l'ultima versione di LVGL nell'albero delle app di NuttX. Persisterà anche dopo `make distclean` perché il sistema di build non rimuoverà lvgl se si tratta di un repository Git.

```
$ cd /path/to/nuttxspace/apps/graphics/lvgl
$ git clone https://github.com/lvgl/lvgl.git
```

Infine, si deve copiare il contenuto di `apps/graphics/lvgl/lvgl/Kconfig` al centro di `apps/graphics/lvgl/Kconfig`. Vedere la sezione "PASTE THE CONTENTS ..." qui sotto.

`apps/graphics/lvgl/Kconfig`:

```
# For a description of the syntax of this configuration file,
# see the file kconfig-language.txt in the NuttX tools repository.
#
menuconfig GRAPHICS_LVGL
    bool "Light and Versatile Graphic Library (LVGL)"
    default n
    ---help---
        Enable support for the LVGL GUI library.

if GRAPHICS_LVGL
(PASTE THE CONTENTS OF apps/graphics/lvgl/lvgl/Kconfig HERE)

config LV_OPTLEVEL
    string "Customize compilation optimization level"
    default ""

endif # GRAPHICS_LVGL
```

Dove si possono trovare maggiori informazioni?

- Questo post del blog: [LVGL on LPCXpresso54628](#)
- Mailing list NuttX: [Apache NuttX Mailing List](#)

4.4.4 PX5 RTOS

Vedere la [homepage](#) di PX5 RTOS

TODO [da fare]

4.4.5 QNX

Cos'è QNX?

QNX è un sistema operativo commerciale rilasciato per la prima volta nel 1980. Il sistema operativo si basa su un design a microkernel, con file system, stack di rete e vari altri driver, ognuno dei quali eseguito in un proprio processo con uno spazio di indirizzamento separato.

Per maggiori dettagli, consultare www.qnx.com.

Punti di forza di QNX

- Solo a 64 bit, funziona su x86_64 e ARMv8
- Richiede una MMU poiché la progettazione impone la separazione tra i processi
- Supporto per migliaia di processi e milioni di thread
- Fino a 64 core, fino a 16 TB di RAM
- Supporto per la virtualizzazione (come host e guest)
- Compatibilità POSIX completa
- Certificazione di sicurezza secondo vari standard automobilistici, industriali e medicali

Come eseguire LVGL su QNX?

Esistono due modi per utilizzare LVGL nei progetti QNX. Il primo è simile a come LVGL viene utilizzato su altri sistemi. Il secondo consiste nel compilare LVGL come libreria 'shared' o statica.

Includere LVGL nel Progetto

Seguire le istruzioni generiche per iniziare a usare LVGL. Dopo aver copiato `lv_conf_template.h` in `lv_conf.h` si apportano le seguenti modifiche a quest'ultimo:

1. Abilitare il supporto QNX:

```
#define LV_USE_QNX 1
```

2. Impostare la profondità di colore a 32:

```
#define LV_COLOR_DEPTH 32
```

3. (Facoltativo) Abilitare il doppio buffering:

```
#define LV_QNX_BUF_COUNT 2
```

Compilare LVGL come libreria

Notare che questo metodo è un'alternativa all'inclusione di LVGL nel progetto. Se si sceglie di compilare una libreria, non è necessario seguire le istruzioni della sezione precedente.

La directory di livello superiore qnx include un file make ricorsivo per compilare LVGL, sia come libreria condivisa che come libreria statica per le architetture supportate. Per compilare tutte le librerie, è sufficiente richiamare make in questa directory:

```
# cd $(LVGL_ROOT)/env_support/qnx
# make
```

Se si preferisce compilare per un'architettura e una variante specifiche, si va alla directory appropriata e si esegue make da lì. Ad esempio, per creare una libreria condivisa per ARMv8:

```
# cd $(LVGL_ROOT)/env_support/qnx/aarch64/so.lib
# make
```

Come regola generale, se si desidera avere una sola applicazione LVGL nel sistema, è meglio utilizzare una libreria statica. Se ce n'è più di una, e soprattutto se vengono eseguite contemporaneamente, è meglio utilizzare la libreria condivisa.

Prima di creare la libreria, è consigliabile modificare `$(LVGL_ROOT)/env_support/qnx/lv_conf.h`, ad esempio per aggiungere font o disabilitare il doppio buffering.

Scrivere un'Applicazione LVGL

Per creare un'applicazione LVGL per QNX, seguire questi passaggi nel codice:

1. Inizializzare la libreria.
2. Creare una finestra.
3. Aggiungere i dispositivi di input.
4. Creare l'interfaccia utente.
5. Eseguire il "loop" di eventi.

I passaggi 2, 3 e 5 utilizzano chiamate specifiche per QNX, ma il resto del codice dovrebbe essere identico a quello di un'applicazione LVGL scritta per qualsiasi altra piattaforma.

Il codice seguente mostra come creare un'applicazione "Hello World":

```
#include <lvgl.h>

int
main(int argc, char **argv)
{
    /* Initialize the library. */
    lv_init();

    /* Create a 800x480 window. */
    lv_display_t *disp = lv_qnx_window_create(800, 480);
    lv_qnx_window_set_title(disp, "LVGL Example");

    /* Add keyboard and mouse devices. */
    lv_qnx_add_keyboard_device(disp);
    lv_qnx_add_pointer_device(disp);

    /* Generate the UI. */
    lv_obj_set_style_bg_color(lv_screen_active(), lv_color_hex(0x003a57), LV_PART_MAIN);

    lv_obj_t *label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Hello world");
    lv_obj_set_style_text_color(lv_screen_active(), lv_color_hex(0xffffffff), LV_PART_MAIN);
    lv_obj_align(label, LV_ALIGN_CENTER, 0, 0);

    /* Run the event loop until it exits. */
    return lv_qnx_event_loop(disp);
}
```

Building dell'Applicazione

Il building dell'applicazione consiste nel compilare il codice sorgente con gli header LVGL e poi nel linkarlo alla libreria. Questo può essere fatto in molti modi, utilizzando diversi sistemi di build. Di seguito è riportato un semplice file make per l'esempio precedente, che compila per ARMv8 con la libreria "shared":

```
CC=gcc -Vgcc_ntoarch64le
LVGL_ROOT=$(HOME)/src/lvgl
CFLAGS=-I$(LVGL_ROOT)/env_support/qnx -I$(LVGL_ROOT)
LDFLAGS=-lscreen -llvgl -L$(LVGL_ROOT)/env_support/qnx/aarch64/so.lib

lvgl_example: lvgl_example.c
    $(CC) $(CFLAGS) -Wall -o $@ $< $(LDFLAGS)

clean:
    rm -f *.o *~ lvgl_example
```

4.4.6 RT-Thread RTOS

Cos'è RT-Thread?

RT-Thread è un sistema operativo real-time (RTOS) open source, neutrale e basato sulla community. RT-Thread è disponibile in **versione Standard** e **versione Nano**. Per i sistemi a microcontrollore (MCU) con risorse limitate, la versione Nano, che richiede solo 3 KB di memoria Flash e 1,2 KB di RAM, può essere personalizzata con strumenti di facile utilizzo. Per i dispositivi IoT ricchi di risorse, RT-Thread può utilizzare lo strumento di gestione dei **pacchetti software online**, insieme agli strumenti di configurazione del sistema, per ottenere un taglio modulare rapido e intuitivo, importare senza problemi pacchetti software complessi e ottenere così funzioni complesse come l'interfaccia grafica di Android e gli effetti di scorrimento touch, gli effetti di interazione vocale intelligente e così via.

Caratteristiche Principali

- Progettato per dispositivi con risorse limitate, il kernel minimo richiede solo 1,2 KB di RAM e 3 KB di Flash.
- Una varietà di interfacce standard, come POSIX, CMSIS, ambiente applicativo C++.
- Ha componenti ricchi e un **ecosistema di pacchetti** prospero e in rapida crescita
- Stile di codice elegante, facile da usare, leggere e padroneggiare.
- Elevata scalabilità. RT-Thread ha un'architettura software scalabile di alta qualità, "loose coupling" [accoppiamento libero], modularità, è facile da personalizzare ed espandere.
- Supporta applicazioni ad alte prestazioni.
- Supporta tutti i principali strumenti di compilazione come GCC, Keil e IAR.
- Supporta un'ampia gamma di **architetture e chip**

Come eseguire LVGL su RT-Thread?

 [documentazione cinese]

LVGL è registrato come software package di RT-Thread. Utilizzando Env tool o RT-Thread Studio IDE, gli utenti di RT-Thread possono scaricare facilmente il codice sorgente di LVGL e integrarlo nel progetto RT-Thread.

La community RT-Thread ha portato LVGL su diversi BSP [Board Support Package]:

BSP	BSP
Simulatore QEMU	Infineon psoc6-evaluationkit-062S2
Simulatore di Visual Studio	Renesas ra6m3-ek
Nuvoton numaker-iot-m487	Renesas ra6m4-cpk
Nuvoton numaker-pfm-m487	Renesas ra6m3-hmi
Nuvoton nk-980iot	STM32H750 ART-Pi
Nuvoton numaker-m2354	STM32F469 Discovery
Nuvoton nk-n9h30	STM32F407 explorer
Nuvoton numaker-m032ki	STM32L475 pandora
Nuvoton numaker-hmi-ma35d1	NXP imxrt1060-evk
Nuvoton numaker-iot-m467	Raspberry PICO
Nuvoton numaker-m467hj	NXP LPC55S69
synwit swm341	

Tutorial

- Introduzione a RT-Thread e a come eseguire LVGL su RT-Thread nei simulatori
- Come importare un progetto BSP con il codice più recente in RT-Thread Studio
- Come usare LVGL con RT-Thread Studio nella board STM32F469 Discovery
- Canale YouTube di RT-Thread
- Centro di documentazione di RT-Thread

4.4.7 Zephyr

Cos'è Zephyr?

Zephyr è un sistema operativo real-time (RTOS) open source facile da implementare, proteggere, connettere e gestire. Dispone di un set crescente di librerie software utilizzabili in diverse applicazioni e settori industriali, come l'IoT industriale, i dispositivi indossabili, il machine learning e altro ancora. Zephyr è progettato con particolare attenzione all'ampio supporto di chipset, alla sicurezza, all'affidabilità, alle release con supporto a lungo termine e a un ecosistema open source in crescita.

Punti di Forza di Zephyr

- **Piccolo** - Funziona su microcontrollori con soli 8 kB di Flash e 5 kB di RAM.
- **Scalabile** - Utilizzabile per sistemi multicore complessi.
- **Personalizzabile** - Supporto immediato per oltre 500 schede e elevata portabilità.
- **Sicuro** - Progettato pensando alla sicurezza e alla protezione, offre supporto "Long-term" [a lungo termine].
- **Ecosistema** - Zephyr non fornisce solo il kernel RTOS, ma anche strumenti di sviluppo, device driver, connettività, logging, tracing, gestione dell'alimentazione e molto altro.
- **Disaccoppiamento** - Sfrutta il devicetree per descrivere e configurare il sistema target.
- **Conforme** - Le app sono eseguibili come applicazioni Linux native, semplificando il debug e la profilazione.

Come eseguire LVGL su Zephyr?

Per configurare l'ambiente di sviluppo, consultare la [guida introduttiva](#).

Dopo aver completato la configurazione sopra descritta, si possono consultare tutti gli [esempi forniti](#) per diverse schede. Si può consultare l'elenco delle schede disponibili usando:

```
$ west boards
```

Dopo aver scelto una scheda, si può compilare una delle demo LVGL per essa. Qui utilizziamo la scheda `native_posix`, che consente di eseguire l'applicazione sul sistema host compatibile con Posix:

```
$ west build -b native_posix samples/modules/lvgl/demos
```

Per eseguire l'applicazione sul proprio host:

```
$ west build -t run
```

Se è stata scelta una qualsiasi delle altre schede supportate, si può flashare sul dispositivo con:

```
$ west flash
```

Per compilare una qualsiasi delle altre applicazioni demo, consultare gli esempi [README](#).

Sfruttare le funzionalità di Zephyr

Shell

Zephyr include una potente implementazione della shell abilitabile con i simboli Kconfig `CONFIG_SHELL` e `CONFIG_LV_Z_SHELL` (le demo precedenti la abilitano di default).

La shell offre la possibilità di abilitare/disabilitare i dispositivi LVGL:

```
# Create a new monkey with the given indev type
uart$ lvgl monkey create [pointer|keypad|button|encoder]

# Enable/Disable a monkey
uart$ lvgl monkey set <index> <inactive/active>
```

Questo è utile per verificare la presenza di "memory leak" [perdite di memoria] e altri bug nell'applicazione. A proposito di memory leak, è anche possibile acquisire statistiche sulla memoria utilizzata da LVGL.

```
uart$ lvgl stats memory
```

Per maggiori dettagli, consultare la documentazione della shell.

Devicetree

Zephyr utilizza il linguaggio di descrizione devicetree per creare e gestire dispositivi di input LVGL.

Le descrizioni dei binding degli pseudo dispositivi sono disponibili all'indirizzo:

- [button input](#)
- [pointer input](#)
- [encoder input](#)
- [keypad input](#)

In sostanza, questi bufferizzano l'evento `input_event` generato dal dispositivo puntato dal phandle `input` oppure, se lasciato vuoto, il binding cattura tutti gli eventi indipendentemente dalla sorgente. Non è necessario istanziare o gestire i dispositivi manualmente, poiché vengono creati all'avvio dell'applicazione prima dell'esecuzione di `main()`.

La maggior parte delle schede o degli shield dotati di display o connettore per display hanno il dispositivo di input del puntatore già dichiarato:

```
lvgl_pointer {
    compatible = "zephyr,lvgl-pointer-input";
    input = <&ft5336_touch>;
};
```

È possibile accedere al `lv_indev_t` di lvgl sottostante per la configurazione. Esempio con il dispositivo encoder per assegnare un `lv_group_t`:

```
const struct device *lvgl_encoder = DEVICE_DT_GET(DT_COMPAT_GET_ANY_STATUS_OKAY(zephyr_lvgl_encoder_input));

lv_obj_t *arc;
lv_group_t *arc_group;

arc = lv_arc_create(lv_screen_active());
lv_obj_align(arc, LV_ALIGN_CENTER, 0, 0);
lv_obj_set_size(arc, 150, 150);

arc_group = lv_group_create();
lv_group_add_obj(arc_group, arc);
lv_indev_set_group(lvgl_input_get_indev(lvgl_encoder), arc_group);
```

Kconfig

Oltre ad abilitare la shell, è possibile utilizzare Kconfig per ottimizzare il footprint dell'applicazione.

```
# Size of the memory region from which lvgl memory is allocated
CONFIG_LV_Z_MEM_POOL_SIZE=8192

# Do not include every widget/theme by default, enable them as needed.
CONFIG_LV_CONF_MINIMAL=y
```

Gli overlay possono essere utilizzati per abilitare/disabilitare funzionalità per schede specifiche o target di build. Per ulteriori informazioni, consultare la guida allo sviluppo di applicazioni.

Ottimizzazione delle prestazioni in LVGL

Per ottimizzare le prestazioni di LVGL, è possibile configurare diverse opzioni kconfig:

- **CONFIG_LV_Z_VDB_SIZE**: Imposta la dimensione del buffer di rendering come percentuale dell'area di visualizzazione, regolabile dall'1% al 100%. Buffer più grandi possono migliorare le prestazioni, soprattutto se utilizzati con **CONFIG_LV_Z_FULL_REFRESH**.
- **CONFIG_LV_Z_DOUBLE_VDB**: Abilita l'utilizzo di due buffer di rendering, consentendo il rendering parallelo e il flushing dei dati, migliorando così la reattività e riducendo la latenza.
- **CONFIG_LV_Z_VDB_ALIGN**: Garantisce che il buffer di rendering sia correttamente allineato, il che è fondamentale per un accesso efficiente alla memoria in base alla profondità del colore.
- **CONFIG_LV_Z_VBD_CUSTOM_SECTION**: Consente di posizionare i buffer di rendering in una sezione di memoria personalizzata (ad esempio, `.lvgl_buf`), utile per sfruttare tipi di memoria specifici come quella strettamente accoppiata o esterna per migliorare le prestazioni.

Opzioni specifiche per Zephyr ≤ 3.7.0

Per le versioni di Zephyr 3.7.0 e precedenti, sono disponibili opzioni aggiuntive per gestire il flushing dei frame LVGL:

- **CONFIG_LV_Z_FLUSH_THREAD**: Abilita il flushing dei frame LVGL in un thread separato, consentendo al thread principale di continuare a eseguire il rendering del frame successivo simultaneamente. Questa opzione può essere disabilitata se non si desidera ottenere un miglioramento delle prestazioni.
 - **CONFIG_LV_Z_FLUSH_THREAD_STACK_SIZE**: Specifica la dimensione dello stack per il thread di flushing, con un valore di default di 1024 byte.
 - **CONFIG_LV_Z_FLUSH_THREAD_PRIO**: Imposta la priorità del thread di flushing, con un valore predefinito di 0, che indica la priorità cooperativa.

Per le versioni più recenti di Zephyr, è possibile utilizzare OSAL (Operating System Abstraction Layer), che si occupa del flushing.

Dove si possono trovare maggiori informazioni?

- Documentazione di Zephyr
- Mailing list di Zephyr
- Server Discord di Zephyr

4.5 Supporto del Framework

4.5.1 Arduino

La libreria [LVGL library](#) è disponibile direttamente come libreria Arduino.

Si noti che è necessario scegliere una scheda sufficientemente potente per eseguire LVGL e la propria GUI. Vedere i [requisiti di LVGL](#).

Ad esempio, ESP32 è un buon candidato per creare interfacce utente con LVGL.

Se si utilizza Arduino GIGA Display Shield (la documentazione [qui](#)), utilizzare la libreria [Arduino_H7_Video](#), inclusa nel pacchetto della scheda Arduino Mbed OS ([qui](#)), che fornisce compatibilità LVGL integrata.

Ottenere la libreria LVGL per Arduino

LVGL può essere installato tramite l'Arduino IDE Library Manager o come libreria .ZIP.

Si può fare il [Download](#) dell'ultima versione di LVGL da GitHub e copiarla semplicemente nella cartella della libreria di Arduino.

Configurare i driver

Per iniziare, l'opzione consigliata è quella di utilizzare la libreria LovyanGFX come driver TFT, poiché semplifica notevolmente i test. Si dovrà creare un file di configurazione del display, come `my_display.hpp`, simile all'esempio in [LovyanGFX user_setting.ino](#). Nell'IDE di Arduino, si può creare un nuovo "tab" e chiamarlo `my_display.hpp`, quindi incollarvi il contenuto del proprio file di configurazione. Se si preferisce di non utilizzare LovyanGFX, si possono integrare anche altre librerie grafiche implementando una classe wrapper nello stesso modo in cui viene gestita LovyanGFX; [lv_lgfd_user.hpp](#) ne è un buon esempio. Una volta pronto il file di configurazione, aggiornare `lv_conf.h` per includerlo, ad esempio:

```
/* Interface for Lovyan_GFX */
#define LV_USE_LOVYAN_GFX 1

#if LV_USE_LOVYAN_GFX
    #define LV_LGFD_USER_INCLUDE "my_display.hpp"
#endif /*LV_USE_LOVYAN_GFX*/
```

In alternativa, si può usare la libreria [TFT_eSPI](#). Per farla funzionare, si configura `TFT_eSPI` in base al tipo di display TFT modificando:

- `User_Setup.h`
- oppure selezionando una configurazione in `User_Setup_Select.h`

Entrambi i file si trovano nella cartella della libreria `TFT_eSPI`.

Configurare LVGL

LVGL ha un proprio file di configurazione chiamato `lv_conf.h`. Una volta installato LVGL, seguire questi passaggi di configurazione:

1. Andare alla directory delle librerie installate di Arduino
2. Andare a `lvgl` e copiare `lv_conf_template.h` come `lv_conf.h` nella directory delle Librerie Arduino accanto alla cartella delle librerie di `lvgl`.
3. Aprire `lv_conf.h` e modificare il primo `#if 0` in `#if 1` per abilitare il contenuto del file
4. Impostare la profondità di colore del display in `LV_COLOR_DEPTH`

Infine, il layout con `lv_conf.h` dovrebbe apparire così:

```
arduino
|libraries
|lvgl
|other_lib_1
|other_lib_2
|lv_conf.h
```

Inizializzare ed eseguire LVGL

Consultare [LVGL_Arduino.ino](#) per vedere come inizializzare LVGL. `TFT_eSPI` viene utilizzato come driver del display.

Nel file INO si può vedere come registrare un display e un touchpad per LVGL e chiamare un esempio.

Usare gli esempi e le demo

Notare che non esiste un file INO dedicato per ogni esempio. Invece, si può caricare un esempio chiamando la funzione `lv_example_....`. Ad esempio `lv_example_btn_1()`.

importante

A causa di alcune limitazioni del sistema di compilazione di Arduino, è necessario copiare `lvgl/examples` in `lvgl/src/examples`. Analogamente, per le demo `lvgl/demos` in `lvgl/src/demos`.

Debug e logging

LVGL può visualizzare informazioni di debug in caso di problemi. Nell'esempio `LVGL_Arduino.ino` è presente il metodo `my_print`, che invia queste informazioni di debug all'interfaccia seriale. Per abilitare questa funzionalità, è necessario modificare il file `lv_conf.h` e abilitare il logging nella sezione `log settings`:

```
/* Log settings */
#define USE_lv_LOG      1 /* Enable/disable the log module */
#if USE_lv_LOG
/* How important log should be added:
 * LV_LOG_LEVEL_TRACE   A lot of logs to give detailed information
 * LV_LOG_LEVEL_INFO    Log important events
 * LV_LOG_LEVEL_WARN   Log if something unwanted happened but didn't cause a problem
 * LV_LOG_LEVEL_ERROR  Only critical issue, when the system may fail
 * LV_LOG_LEVEL_NONE   Do not log anything
*/
#define LV_LOG_LEVEL LV_LOG_LEVEL_WARN
```

Dopo aver abilitato il modulo di log e impostato `LV_LOG_LEVEL` di conseguenza, il log di output viene inviato alla porta Seriale @ 115200 bps.

4.5.2 PlatformIO

PlatformIO è un ecosistema open source per lo sviluppo embedded che si integra con VS Code e supporta numerose schede e framework.

La libreria [LVGL library](#) è disponibile direttamente nel registro delle librerie di PlatformIO e può essere aggiunta al progetto con pochi clic.

Si noti che è necessario scegliere una scheda sufficientemente potente per eseguire LVGL e la propria GUI. Vedere i [requisiti di LVGL](#).

Ad esempio, la famiglia ESP32 è un'ottima candidata per creare interfacce utente con LVGL. Le schede consigliate includono ESP32 (dual-core originale) ed ESP32-S3.

Ottenere la libreria LVGL

Il metodo consigliato per aggiungere driver LVGL e display è tramite l'interfaccia utente di PlatformIO Home in VS Code:

1. Aprire PIO Home cliccando sull'icona di PlatformIO nella toolbar a sinistra.
2. Andare su `Libraries`.
3. Nella barra di ricerca, digitare il nome della libreria (ad es. "lvgl").
4. Cliccare sulla libreria desiderata tra i risultati.
5. Premere `Add to Project`.
6. Selezionare il progetto target dall'elenco a discesa.

Questo aggiungerà automaticamente la libreria alla sezione `lib_deps` del file `platformio.ini` del progetto.

Si può anche modificare direttamente il file `platformio.ini` e aggiungere le librerie in `lib_deps`:

```
lib_deps =
    lvgl/lvgl
```

Si può anche specificare la versione della libreria da utilizzare. Vedere [platformio.ini configuration](#)

Se si preferisce l'installazione manuale, si può clonare il repository LVGL nella directory `lib/` del proprio progetto PlatformIO.

Configurare i driver

Per iniziare, l'opzione consigliata è quella di utilizzare la `LovyanGFX library` come driver TFT, poiché semplifica notevolmente i test. In PlatformIO, si aggiungeranno sia LVGL che LovyanGFX come dipendenze nel proprio file `platformio.ini`:

```
lib_deps =
    lvgl/lvgl
    lovyan03/LovyanGFX
```

Un'altra opzione popolare è la `TFT_eSPI library`, che supporta un'ampia gamma di display.

Inoltre, LVGL fornisce anche i propri driver display integrati utilizzabili direttamente nel proprio progetto. Consulta la sezione [supported drivers](#) per un elenco completo.

Configurazione del Display

Si dovrà creare un file di configurazione del display, come `my_display.hpp`, simile all'esempio in `LovyanGFX user_setting.ino`.

In PlatformIO, si può posizionare questo file nella cartella `include/` del proprio progetto (ad esempio, `include/my_display.hpp`).

Se si preferisce di non utilizzare LovyanGFX, si possono integrare anche altre librerie grafiche implementando una classe wrapper nello stesso modo in cui viene gestita LovyanGFX; `lv_lgfd_user.hpp` ne è un buon esempio.

Una volta pronto il file di configurazione, aggiornare `lv_conf.h` per includerlo, ad esempio:

```
/* Interface for Lovyan_GFX */
#define LV_USE_LOVYAN_GFX 1

#if LV_USE_LOVYAN_GFX
    #define LV_LGFD_USER_INCLUDE "my_display.hpp"
#endif /*LV_USE_LOVYAN_GFX*/
```

Quando si utilizza l'interfaccia LovyanGFX, è necessario aggiungere quanto segue al file `platformio.ini`:

```
build_flags =
    -I include ; the folder containing my_display.hpp
    -I .pio/libdeps/${PIOENV}/LovyanGFX/src ; so that PIO includes this when compiling LVGL
```

In alternativa, è possibile utilizzare la libreria `TFT_eSPI`. Per farla funzionare, configurare `TFT_eSPI` in base al tipo di display TFT modificando:

- `User_Setup.h`
- oppure selezionando una configurazione in `User_Setup_Select.h`

Entrambi i file si trovano nella cartella della libreria `TFT_eSPI` all'interno della directory `.pio/libdeps/<env>/TFT_eSPI/` di PlatformIO.

Con PlatformIO, è anche possibile sovrascrivere la configurazione utilizzando `build_flags` in `platformio.ini`, evitando di modificare direttamente i file della libreria. Ad esempio:

```
build_flags =
    -D USER_SETUP_LOADED=1
    -D ST7789_DRIVER=1
    -D TFT_WIDTH=240
    -D TFT_HEIGHT=240
    -D TFT_MOSI=23
    -D TFT_SCLK=18
    -D TFT_CS=5
    -D TFT_DC=16
    -D TFT_RST=17
```

Questo approccio mantiene la configurazione sotto controllo di versione.

Configurare LVGL

- Uso di `lv_conf.h`

Dopo aver installato LVGL, copiare `lv_conf_template.h` dal repository LVGL nel proprio progetto e rinominalo in `lv_conf.h`. Inserirlo nella cartella `include/` o `src/` e abilitare/disabilitare le funzionalità LVGL modificando le macro. Ad esempio:

```
#define LV_COLOR_DEPTH 16
#define LV_USE_LOG 1
```

- Uso di `platformio.ini`

In alternativa, si possono definire le macro di configurazione LVGL direttamente in `platformio.ini` sotto `build_flags`:

```
build_flags =
-D LV_CONF_SKIP
-D LV_COLOR_DEPTH=16
-D LV_USE_LOG=1
-D LV_FONT_MONTserrat_24=1
```

Questo evita di dover gestire un file `lv_conf.h` separato.

Inizializzare ed eseguire LVGL

In `main.cpp`, inizializzare LVGL e il driver video:

```
#include <Arduino.h>
#include <lvgl.h>

#define BUF_SIZE 320 * 50
uint8_t lv_buffer[BUF_SIZE];

/* Tick source, tell LVGL how much time (milliseconds) has passed */
static uint32_t my_tick(void)
{
    return millis();
}

void setup() {
    /* Initialize LVGL */
    lv_init();
    /* Set the tick callback */
    lv_tick_set_cb(my_tick);
    /* Initialize the display driver */
    lv_loyan_gfx_create(320, 480, lv_buffer, BUF_SIZE, true);

    lv_obj_t *label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Hello PlatformIO, I'm LVGL!");
    lv_obj_align(label, LV_ALIGN_CENTER, 0, 0 );
}

void loop() {
    lv_timer_handler(); // Update the UI-
    delay(5);
}
```

Usare gli esempi e le demo

LVGL include un'ampia gamma di esempi e applicazioni demo che si possono provare in PlatformIO. Questi sono inclusi nella libreria nelle cartelle `examples` e `demos`.

Per forzare PlatformIO a compilarli ed eseguirli, aggiungere quanto segue al proprio file `platformio.ini`:

```
build_src_filter =
+<>
; Force compile LVGL demos and examples, remove when working on your own project
+<./pio/libdeps/${PIOENV}/lvgl/demos>
+<./pio/libdeps/${PIOENV}/lvgl/examples>
```

Questo includerà tutti i `demos` e gli `examples` di LVGL nella build. Quando si è pronti a lavorare sul progetto, si devono rimuovere queste righe in modo che vengano compilati solo i sorgenti del progetto.

Come riferimento, il repository [lv_platformio](#) mostra come configurare ed eseguire LVGL su diversi target, tra cui ESP32, STM32 e un simulatore per PC (SDL2). Questo può essere molto utile per esplorare il comportamento di LVGL su diverse piattaforme.

Debug e logging

LVGL può visualizzare informazioni di debug in caso di problemi. In PlatformIO è possibile abilitare il logging modificando il file `lv_conf.h` (o aggiungendo i "define" in `platformio.ini`).

In `lv_conf.h`, si va alla sezione "Log settings" e si abilita il modulo log:

```
/* Log settings */
#define LV_USE_LOG 1 /* Enable/disable the log module */
#if LV_USE_LOG
    /* How important log should be added:
     * LV_LOG_LEVEL_TRACE A lot of logs to give detailed information
     * LV_LOG_LEVEL_INFO Log important events
     * LV_LOG_LEVEL_WARN Log if something unwanted happened but didn't cause a problem
     * LV_LOG_LEVEL_ERROR Only critical issues, when the system may fail
     * LV_LOG_LEVEL_NONE Do not log anything
    */
#define LV_LOG_LEVEL LV_LOG_LEVEL_WARN
#endif
```

Dopo aver abilitato il modulo log e impostato `LV_LOG_LEVEL` di conseguenza, LVGL invierà messaggi di log alla porta Seriale (per default a 115200 bps).

Se si preferisce non modificare `lv_conf.h`, si può ottenere lo stesso risultato aggiungendo flag di build in `platformio.ini`:

```
build_flags =
-D LV_USE_LOG=1
-D LV_LOG_LEVEL=LV_LOG_LEVEL_INFO
```

È quindi possibile stampare i log su seriale con una funzione di callback, ad esempio:

```
static void lv_log_print_g_cb(lv_log_level_t level, const char *buf)
{
    LV_UNUSED(level);
    Serial.write(buf);
}

void setup() {
    Serial.begin(115200);
    lv_log_register_print_cb(lv_log_print_g_cb); // Register print function for LVGL logs
}
```

4.5.3 Tasmota e Berry

Cos'è Tasmota?

Tasmota è un firmware open source ampiamente utilizzato per dispositivi basati su ESP8266 e ESP32. Supporta un'ampia varietà di dispositivi, sensori e integrazioni con servizi di domotica e cloud. Il firmware Tasmota viene scaricato più di 200.000 volte al mese e vanta una community attiva e in crescita.

Tasmota offre accesso a centinaia di dispositivi supportati, supporto completo di MQTT, HTTP(S), integrazione con i principali sistemi di domotica, una miriade di sensori, IR, RF, Zigbee, Bluetooth, AWS IoT, Azure IoT, Alexa e molto altro.

Cos'è Berry?

Berry è un linguaggio di scripting embedded a tipizzazione dinamica ultraleggero. È progettato per dispositivi embedded a basse prestazioni. L'interprete di Berry include un compilatore one-pass e una VM basata su registri; tutto il codice è scritto in ANSI C99. Berry offre una sintassi molto simile a Python e si ispira alla VM LUA. È completamente integrato in Tasmota.

Punti di forza di Berry

Berry offre i seguenti vantaggi:

- Leggero: Un interprete ben ottimizzato con risorse molto ridotte. Ideale per l'uso nei microprocessori.
- Veloce: compilatore bytecode ottimizzato a passaggio singolo e macchina virtuale basata su registri.
- Potente: supporta la programmazione imperativa, la programmazione orientata agli oggetti e la programmazione funzionale.
- Flessibile: Berry è uno script di tipo dinamico ed è progettato per l'integrazione nelle applicazioni. Può fornire una buona scalabilità dinamica per il sistema host.

- Semplice: sintassi semplice e naturale, supporto per la garbage collection e interfaccia FFI (foreign function interface) facile da usare.
- Risparmio di RAM: Con la costruzione di oggetti in fase di compilazione, la maggior parte degli oggetti costanti viene memorizzata in segmenti di dati di codice di sola lettura, quindi l'utilizzo di RAM dell'interprete è molto basso all'avvio.

Tutte le funzionalità sono descritte nel [Berry Reference Manual](#)

Perché LVGL + Tasmota + Berry?

Nel 2021, Tasmota ha aggiunto il supporto completo di LVGL per i dispositivi basati su ESP32. Ha inoltre introdotto il linguaggio di scripting Berry, un linguaggio compatto simile a Python e completamente integrato in Tasmota.

È ora disponibile una mappatura completa di LVGL nel linguaggio Berry, simile a quella di MicroPython. Consente di utilizzare oltre il 98% di tutte le funzionalità di LVGL. È anche possibile scrivere widget personalizzati in Berry.

Versioni supportate: LVGL v8.0.2, LodePNG v20201017, Freetype 2.10.4

Tasmota + Berry + LVGL possono essere utilizzati per:

- Interfaccia grafica per prototipazione rapida.
- Abbreviare il ciclo di modifica e messa a punto dell'interfaccia utente grafica.
- Modellare la GUI in modo più astratto definendo widget compositi riutilizzabili, sfruttando le funzionalità del linguaggio Berry come Ereditarietà, Chiusure, Gestione delle Eccezioni...
- Rendere LVGL accessibile a un pubblico più vasto. Non è necessario conoscere il C per creare una buona GUI su un sistema embedded.

È in fase di sviluppo anche un'interfaccia di livello superiore compatibile con [OpenHASP](#).

Che aspetto ha?

In breve: Simile a MicroPython, è molto simile all'API C, ma orientata agli oggetti per i componenti LVGL.

Immergiamoci subito in un esempio!

Un semplice esempio

```
lv.start()           # start LVGL
scr = lv.screen_active() # get default screen
btn = lv.btn(scr)    # create button
btn.center()
label = lv.label(btn) # create a label in the button
label.set_text("Button") # set a label to the button
```

Come si usa?

Si può iniziare in meno di 10 minuti su un M5Stack o un dispositivo equivalente in meno di 10 minuti con questo breve tutorial

Dove si possono trovare maggiori informazioni?

- Documentazione di Tasmota
- Documentazione di Berry
- Documentazione di Berry per Tasmota LVGL

4.6 Supporto delle Board

4.6.1 Supporto LVGL

4.6.2 Partner Supportato

4.6.3 Board

ICOP

TODO [da fare]

Toradex

Informazioni su Toradex

Toradex è specializzata nella produzione di soluzioni di elaborazione embedded di alta qualità basate su unità System-on-Module (SoM). Queste schede e moduli sono progettati per soddisfare le esigenze di applicazioni industriali e commerciali, offrendo affidabilità, longevità e scalabilità. Sfruttando framework standard di settore come lo Yocto Project, Toradex consente ai clienti di creare sistemi embedded personalizzati con facilità, garantendo compatibilità e flessibilità.

Portfolio Prodotti

Toradex offre un'ampia gamma di System-on-Module (SoM) e schede carrier progettate per diverse applicazioni e settori. Ecco una panoramica di base per aiutare a navigare nel portfolio.

Aquila

Aquila è un nuovo standard System on Module che espande il portfolio Toradex a nuovi livelli, offrendo prestazioni senza pari per carichi di lavoro CPU, machine learning e visione artificiale ad alta intensità di calcolo.

Per soddisfare i requisiti di intelligenza artificiale di nuova generazione in ambienti industriali e in altri ambienti difficili, Aquila introduce un connettore collaudato: il connettore board-to-board a 400 pin, che rende la scheda estremamente affidabile, soprattutto in questi ambienti.

Computer on Module Aquila:

- [Aquila iMX95](#)
- [Aquila AM69](#)

Schede Carrier Aquila:

- [Clover](#)
- [Aquila Development Board](#)

Apalis

Apalis è una famiglia scalabile di System on Module (SoM) / Computer on Module (CoM) che mira a fornire elevate prestazioni in un fattore di forma compatto. Apalis consente lo sviluppo di prodotti avanzati e robusti in tempi rapidi e con rischi ridotti. I moduli Apalis raggiungono le prestazioni più elevate nella gamma di prodotti Toradex e offrono il maggior numero di interfacce di qualsiasi altra linea di prodotti sui connettori MXM a 314 pin. I moduli Apalis sono adatti a un ampio numero di applicazioni in settori quali sanità, automazione industriale, robotica, smart city e molti altri. I moduli Apalis sono supportati dall'offerta software Toradex e dal ricco ecosistema di altri prodotti e servizi dell'azienda.

Computer on Module Apalis:

- [Apalis iMX8](#)
- [Apalis iMX6](#)
- [Apalis T30](#)

Schede Carrier Apalis:

- [Ixora](#)
- [Apalis Evaluation Board](#)

Colibri

La famiglia The Colibri Arm® è composta da System on Module (SoM) / Computer on Module (CoM) di dimensioni SODIMM. La famiglia offre moduli miniaturizzati basati su NXP® i.MX 8X, i.MX 6ULL, i.MX 7, i.MX 6, Vybrid e NVIDIA® Tegra SoCs. Tutti questi moduli sono compatibili a livello di pin, garantendo la scalabilità delle applicazioni sviluppate su di essi. La famiglia Colibri offre progetti ottimizzati in termini di costi e prestazioni. È inoltre disponibile un'ampia gamma di schede carrier e accessori compatibili con Colibri, come fotocamere e display, per un facile sviluppo di prodotti sulla famiglia di moduli Colibri Arm.

Computer on Module Colibri:

- [Colibri iMX8X](#)
- [Colibri T30](#)
- [Colibri T20](#)
- [Colibri iMX6](#)
- [Colibri iMX7](#)
- [Colibri iMXiMX6ULL8X](#)
- [Colibri VF61](#)
- [Colibri VF50](#)

Schede Carrier Colibri:

- [Colibri Evaluation Board](#)
- [Iris Carrier Board](#)
- [Viola Carrier Board](#)
- [Aster Carrier Board](#)

Verdin

Verdin è uno standard System on Module che amplia i fattori di forma di successo Colibri e Apalis. Offre un set di interfacce rivisitato, moderno e funzionale, puntando su facilità d'uso, robustezza e compatibilità con le esigenze future.

Verdin ha dimensioni simili a Colibri, ma con un numero di pin maggiore. Un connettore edge SODIMM DDR4 offre una connessione conveniente, altamente affidabile e resistente a urti e vibrazioni.

Computer on Module Verdin:

- [Verdin iMX8M Plus](#)
- [Verdin iMX8M Mini](#)
- [Verdin AM62](#)
- [i.MX 95 Verdin Evaluation Kit](#)
- [Verdin iMX95](#)

Schede Carrier Verdin:

- [Verdin Development Board con HDMI Adapter](#)
- [Dahlia Carrier Board con HDMI Adapter](#)
- [Yavia](#)

- Mallow Carrier Board
- Ivy Carrier Board

TorizonOS

TorizonOS è un sistema operativo basato su Linux sviluppato da Toradex, pensato appositamente per i sistemi embedded. È progettato per semplificare lo sviluppo, l'implementazione e la manutenzione delle applicazioni, in particolare in ambito industriale e IoT. Combinando la facilità d'uso con potenti funzionalità come la containerizzazione e gli aggiornamenti over-the-air (OTA), TorizonOS aiuta gli sviluppatori ad accelerare il time-to-market, garantendo al contempo prestazioni e sicurezza elevate per le applicazioni embedded.

Una guida di TorizonOS per sviluppare un'applicazione utilizzando LVGL è disponibile nella sezione [Torizon OS](#).

Ulteriori informazioni sono disponibili nella [documentazione di Torizon](#).

Esempi di Toradex

Sono disponibili repository pronti all'uso. Cliccare [qui](#) per consultarli.

Riverdi

Riverdi è specializzata nella produzione di display di alta qualità e delle relative schede. Offrono soluzioni che spaziano dai semplici pannelli display ai display intelligenti, con tutto ciò che sta nel mezzo.

- Display embedded STM32
- Display per Computer Single Board
- Display intelligenti EVE
- Display HDMI
- Evaluation Board
- Display RGB, LVDS, MIPI DSI LCD

Display embedded STM32

I display embedded STM32 sono dotati di un MCU STM32 integrato che consente di integrare un'intera applicazione nel dispositivo. LVGL è particolarmente adatto a queste schede.

lv_port_riverdi_stm32u5

lv_port_riverdi_stm32u5 è un porting del repository di LVGL che supporta tutti i Display Embedded Riverdi STM32 da 5 pollici.

- RVT50HQSNWC00-B
- RVT50HQSNWC00
- RVT50HQSFCA0
- RVT50HQSNWCA0
- RVT50HQSFWN00
- RVT50HQSNWN00

Supporta nativamente la GPU NeoChrom embedded grazie al supporto LVGL per [NeoChrom](#). NemaVG, un'estensione del supporto base di NeoChrom, è implementato da questa MCU.

L'MCU è dotato di una periferica driver LCD che LVGL utilizza per aggiornare il display. Consultare la documentazione del driver [LTDC](#) per maggiori informazioni e per personalizzarne l'utilizzo.

Display per Single-Board Computer

I Single-Board Computer Displays sono pronti per l'uso con le schede carrier Dahlia e Mallow [Toradex](#). Infatti, queste schede carrier sono compatibili con tutti i display Riverdi MIPI-DSI a 34 pin.

Altri prodotti

Gli **EVE Intelligent Display** sono dotati di un controller grafico IC Bridgetek EVE, in modo che il display possa essere controllato utilizzando comandi di disegno di alto livello su un'interfaccia a larghezza di banda inferiore rispetto a RGB, MIPI, ecc.

Gli **HDMI Displays** sono dotati di un'interfaccia HDMI per un'integrazione semplificata con PC o SBC. Questi display hanno comunque un profilo basso, quindi possono essere integrati facilmente in dispositivi personalizzati.

Le **Evaluation Boards** sono schede di supporto convenienti per far funzionare i pannelli di visualizzazione Riverdi prima di progettare o acquistare una scheda di supporto per un'applicazione di produzione.

Viewe

Viewe è specializzata in soluzioni di visualizzazione complete da oltre 10 anni, con una notevole capacità di ricerca e sviluppo software e hardware, produzione in fabbrica e assistenza commerciale. Viewe non offre solo [soluzioni complete per display e touch](#) ma anche [display intelligenti embedded](#) con produzioni standard e personalizzate per trasformare i vostri progetti in realtà.

Portfolio Prodotti:

- Display Intelligenti Embedded
 - [IOT_AIOT Smart Display](#)
 - [Uart Smart Display](#)
 - [HDMI Display_Raspberry Pi Display](#)
 - [Display Arduino](#)

Display TFT | OLED

- [Sunlight Readable Display](#)
- [AMOLED Displays](#)
- [Transflective TFT Displays](#)
- [Standard TFT Displays](#)
- [Bar Display/Stretched TFT Display](#)
- [Square TFT Display](#)
- [Round TFT Display](#)

IOT AIOT Smart Displays

Gli Smart Display Viewe IOT AIOT sono una serie di display che utilizzano microcontrollori ESP32 con interazioni multimodali integrate di GUI/voce/telecamera, edge computing ML/AI e connettività fluida. Interfacce UART/RS232/RS485/CAN e interfacce IO ridondanti sono incluse per facilitarne l'utilizzo prolungato. LVGL/Arduino/ESP-IDF sono particolarmente adatti a queste schede.

[lv_port_viewe_7_espifdf](#)

[lv_port_viewe_7_espifdf](#) è un repository di porte LVGL che supporta lo Smart Display Viewe ESP32 S3 da 7 pollici. Si basa su ESP-IDF e utilizza il microcontrollore ESP32-S3.

- [Viewe ESP32 S3 Smart Display](#)

lv_port_viewe_knob_15_espifdf

`lv_port_viewe_knob_15_espifdf` è un "port repository" LVGL che supporta il display touch Knob AMOLED Viewe. Si basa su ESP-IDF e utilizza il microcontrollore ESP32-S3.

- Viewe AMOLED Touch Knob Display

Display Intelligenti UART

I display intelligenti UART Viewe sono una serie di display che utilizzano microcontrollori ESP32 e STM32 con interfaccia UART/RS232/RS485/CAN e dispongono di interfaccia IO per la connessione alle periferiche. LVGL è particolarmente adatto a queste schede.

lv_port_viewe_7_espifdf

`lv_port_viewe_7_espifdf` è un repository di porte LVGL che supporta lo Smart Display Viewe ESP32 S3 da 7 pollici. Si basa su ESP-IDF e utilizza il microcontrollore ESP32-S3.

- Viewe ESP32 S3 Smart Display

4.7 Supporto ai Fornitori di Chip

4.7.1 Alif

Panoramica

Alif Semiconductor è un fornitore di chip di nuova generazione che produce chip disponibili in una varietà di configurazioni. I loro chip sono progettati per eccellere sotto diversi aspetti contemporaneamente. Offrono combinazioni di prestazioni, basso consumo energetico, sicurezza e funzionalità speciali come l'intelligenza artificiale. Molti dei loro chip hanno due core asimmetrici. Un core è in genere ad alte prestazioni, mentre l'altro ad alta efficienza. Alif offre sia microcontrollori che microprocessori.

LVGL su schede Alif

Questa è una guida per iniziare a usare LVGL su una scheda Alif. Descrive in dettaglio tutti i passaggi necessari per far funzionare il progetto di esempio LVGL `alif_m55-lvgl` su [Alif E7 Devkit Gen2](#); tuttavia, qualsiasi progetto basato sul [Alif VS Code Template](#) ha una procedura di installazione pressoché identica, quindi questa può essere utilizzata come guida generale per questi progetti. Esistono altri modi per compilare per le schede Alif, ad esempio con Zephyr RTOS. Vedere [i repository GitHub di Alif](#).

Questa guida è per Linux e Windows.

Questo progetto utilizza l'accelerazione di rendering 2D D/AVE con D/AVE 2D [draw unit](#) di LVGL.

Guida Passo-Passo

Installazione Visual Studio Code

Installare Visual Studio code. Esistono diversi modi per installarlo a seconda della piattaforma. [Vedere qui](#).

Nota

I passaggi rimanenti possono essere eseguiti **facoltativamente** all'interno di un container Docker. È possibile connettersi al container Docker per lo sviluppo da remoto con VS Code.

Dovrebbero essere create due porte seriali al momento della connessione della scheda Alif. Su Linux, saranno simili a `/dev/ttyACM0` e `/dev/ttyACM1`. Nel comando `docker run` utilizzato per creare il container di sviluppo, includere `--device /dev/ttyACM0 --device /dev/ttyACM1` per consentire al container di accedere a quelle porte in modo da poter eseguire il flashing da esso.

Installare l'estensione VS Code "Dev Containers". Selezionare il container da "Remote Explorer" nel pannello laterale sinistro.

Installare gli strumenti Prerequisiti

Assicurarsi che siano installati nel proprio ambiente. Le estensioni VS Code si basano sulla loro presenza.

- git
- curl
- unzip

Installare gli Alif SE Tool

Creare un account Alif e scaricare gli strumenti da [qui](#) sotto "Alif Security Toolkit". Estrarre il file. Il path in cui è stato estratto sarà necessario in seguito.

Su Linux, l'estrazione può essere eseguita eseguendo quanto segue:

```
cd
tar -xf Downloads/APFW0002-app-release-exec-linux-SE_FW_1.101.00_DEV.tar
ls
pwd
```

Tra i risultati di ls si dovrebbe vedere app-release-exec-linux. Questo, combinato con l'output di pwd, è il percorso che si dovrà utilizzare in seguito. Ovvero, /home/you/app-release-exec-linux.

Installare il Software J-Link (facoltativo)

Scaricare l'ultima versione stabile del software [J-Link Software](#). Il path di installazione sarà necessario in seguito.

Clonare il Progetto alif_m55-lvgl

```
git clone --recursive https://github.com/alifsemi/alif_m55-lvgl
```

Apertura di alif_m55-lvgl in VS Code

Aprire il repository clonato in VS Code. Affinché le estensioni di VS Code funzionino correttamente, si consiglia di aprire la cartella in VS Code anziché aprire una directory padrone contenente il progetto.

```
code alif_m55-lvgl
```

oppure si va su **File > Open Folder** in VS Code e si apre alif_m55-lvgl.

Se viene richiesto di installare automaticamente le estensioni consigliate, cliccare su "Install" per saltare il passaggio successivo.

Installazione delle Estensioni di VS Code Richieste

Installare le seguenti estensioni di VS Code dal pannello laterale "Extensions"

- Arm Tools Environment Manager
- Arm CMSIS Solution
- C/C++ Extension Pack
- Cortex-Debug (facoltativo necessario per il debug)

Activate Environment

Se non è avvenuto automaticamente, cliccare su "Arm Tools" nella barra in basso e poi su "Activate Environment" nell'elenco che appare. Verranno installati CMake, ninja-build, un compilatore GCC ARM e cmsis-toolbox.

Se si vede solo "Reactivate Environment", è probabile che sia già attivo.

Impostare i Path dei Tool Installati

Premere ctrl + shift + p. Digitare "preferences" e selezionare l'opzione "Preferences: Open User Settings (JSON)" tra le opzioni. Si aprirà un file `settings.json`. Nota: se si utilizza un container Docker, è meglio modificare il file JSON nella directory del progetto in `.vscode/settings.json`.

Si devono aggiungere alcune voci (almeno "`alif.setools.root`") al JSON che si vede.

Se il `settings.json` inizialmente appare così...

```
{
  "workbench.colorTheme": "Default Dark+",
  "editor.renderWhitespace": "all",
}
```

... allora dovrebbe apparire così:

```
{
  "workbench.colorTheme": "Default Dark+",
  "editor.renderWhitespace": "all",
  "alif.setools.root": "C:/alif-se-tools/app-release-exec",
  "cortex-debug.JLinkGDBServerPath": "C:/Program Files/SEGGER/JLink/JLinkGDBServerCL.exe"
}
```

Quanto sopra usa i path Windows come esempio. Un path Linux per i Tool Alif SE potrebbe essere simile a `/home/you/app-release-exec-linux`.

Configurare la Variante della Board

Aprire il file `board.h`.

Identificare la variante della scheda nell'elenco e impostare `BOARD_ALIF_DEVKIT_VARIANT` sul valore corretto. Potrebbe essere necessario impostare anche `BOARD_ILI9806E_PANEL_VARIANT` se il valore predefinito non corrisponde al proprio.

Impostare il Build Context, Compilare e Flashare

Accedere alla vista "Manage Solution" dall'estensione CMSIS Solution. Vi si può accedere cliccando sull'icona a forma di ingranaggio nella barra inferiore oppure andando al pannello CMSIS a sinistra e cliccando sull'ingranaggio in alto. Una volta aperto, si presenta come una scheda dell'editor grafico chiamata "Manage Solution" con un'icona a forma di ingranaggio.

Importante

In "Run and Debug" > "Run Configuration" (column) > "alif" (row), cliccare sul menù a discesa e selezionare "First time pack installation". Cliccare sull'icona di riproduzione nella parte superiore del pannello laterale sinistro di CMSIS. Nel terminale si dovranno vedere i pacchetti CMSIS in fase di installazione. Attendere il completamento.

Ora si può cliccare sull'icona del martello accanto all'icona di "play" per compilare il progetto. Verranno compilate alcune centinaia di file. Attendere il completamento.

Aprire il menù a discesa precedente e selezionare "Prepare program with Security Toolkit". Cliccare sull'icona di "play" [riproduzione]. Vengono preparati alcuni file come passaggio preliminare al flashing.

Aprire di nuovo il menù a discesa e selezionare "Program with Security Toolkit". Cliccare sull'icona di "play" [riproduzione]. Se è la prima volta, verrà chiesto di scegliere quale porta seriale utilizzare per flashare la scheda. Si può provare `/dev/ttyACM0`. Se è quella sbagliata, l'operazione fallirà e si dovrà aprire il menù a discesa e selezionare "Program with Security Toolkit (select COM port)" per sovrascrivere quella precedente che è stata salvata come default.

Il benchmark LVGL dovrebbe essere eseguito sulla scheda Alif al termine del flashing.

Ulteriori Informazioni

In caso di difficoltà durante la lettura di questa guida, fare riferimento a queste fonti Alif per passaggi più dettagliati.

- [Getting Started with VSCode CMSIS pack project](#)
- [VSCode Getting Started Template](#)

È possibile scaricare la "Alif Security Toolkit Quick Start Guide" da <https://alifsemi.com/support/software-tools/ensemble/>, presupponendo che sia stato creato un account, per imparare a utilizzare gli strumenti Alif SE per eseguire manipolazioni di basso livello sulla scheda.

I Core HP e HE e Build Ottimizzata

Nella vista "Manage Solution" illustrata nella guida, è possibile selezionare un target HP o un target HE. Questi si riferiscono ai due core distinti presenti nell'Alif E7. "HE" sta per "High Efficiency" mentre "HP" sta per "High Performance". Per ottenere le migliori prestazioni da un'applicazione LVGL, selezionare HP. Considerare HE quando il consumo energetico è un problema. Il vantaggio di avere core asimmetrici è che l'applicazione può teoricamente eseguire carichi di lavoro a bassa priorità in modo efficiente sul core HE e delegare carichi di lavoro critici e ad alta intensità di elaborazione al core HP.

È anche possibile scegliere un "Build Type". Per prestazioni ottimali, scegliere "release". Se si esegue il debug, scegliere "debug".

Per massimizzare il punteggio nel benchmark LVGL e massimizzare le prestazioni di un'applicazione LVGL in generale, deve essere selezionato il core HP e il tipo di build dev'essere release.

GPU Dave2D

4.7.2 Arm

Panoramica

Arm è un'azienda leader nella progettazione di semiconduttori e software, rinomata per aver creato i core dei microcontrollori (MCU) Cortex-M e quelli dei processori Cortex-A/R (MPU), che sono parte integrante di un'ampia gamma di dispositivi. Questi core sono il cuore di molti sistemi embedded e alimentano chip di giganti del settore come STMicroelectronics, NXP e Renesas. I progetti a basso consumo energetico di Arm sono utilizzati in miliardi di dispositivi in tutto il mondo, dai microcontrollori agli smartphone e ai server. Concedendo in licenza i propri progetti di processori, Arm consente a un ampio ecosistema di partner di sviluppare soluzioni personalizzate ottimizzate per prestazioni, consumo energetico e dimensioni. L'architettura di Arm è altamente compatibile con vari sistemi operativi e librerie software, tra cui LVGL, rendendola una scelta versatile per gli sviluppatori che creano interfacce utente grafiche efficienti e ad alte prestazioni.

Compilare LVGL per Arm

Non è richiesta alcuna azione specifica. Qualsiasi compilatore che supporti l'architettura Arm come target può essere utilizzato per compilare il codice sorgente di LVGL, inclusi GCC, LLVM e AC6.

È anche possibile cross-compilare LVGL per una MPU (invece di compilarlo sull'hardware target) o creare una libreria shared. Per ulteriori informazioni, consultare [CMake](#).

Introduzione a AC6

Poiché AC6 è una toolchain proprietaria, contiene molte ottimizzazioni specifiche, quindi è possibile aspettarsi le migliori prestazioni quando la si utilizza.

AC6 non è gratuita, ma offre una licenza community che può essere attivata come segue:

1. Scaricare e installare il compilatore AC6 dal [sito web di Arm](#).
2. Per registrare una licenza community, andare alla cartella bin del compilatore e, in un terminale, eseguire `armlm.exe activate -server https://mdk-preview.keil.arm.com -product KEMDK-COM0` (Su Linux, usare `./armlm`).

Supporto IDE

Non ci sono limitazioni sugli IDE supportati. LVGL funziona negli IDE di vari fornitori, tra cui Keil MDK di Arm, IAR, e2 studio di Renesas, MCUXpresso di NXP, CubeIDE di ST, nonché progetti custom make o CMake.

Arm2D e il set di istruzioni Helium

Arm Cortex-M55 e Cortex-M85 dispongono del set di istruzioni SIMD Helium. Tra le altre cose, questo può effettivamente velocizzare il rendering dell'interfaccia utente. [Arm2D](#) è una libreria gestita da Arm che sfrutta il set di istruzioni Helium.

Notare che GCC presenta alcuni problemi noti con le "Helium intrinsics". Si consiglia di utilizzare AC6 o LLVM quando si utilizza codice Helium.

Per aggiungere Arm2D al progetto, seguire questi passaggi:

1. Per sfruttarne la potenza, assicurarsi che `mcpu` sia impostato su `cortex-m85`, `cortex-m55`, o `cortex-m52` e aggiungere il flag `-fvectorize`. Per testare senza SIMD, si usa ad esempio `cortex-m85+nomve`.
2. Arm2D può essere scaricato da <https://github.com/ARM-software/Arm-2D>. Valutare la possibilità di utilizzare il branch `developing`, che contiene gli aggiornamenti più recenti.
3. Aggiungere `Arm-2D/Library/Include` ai path degli "include".
4. Copiare `Arm-2D/Library/Include/template/arm_2d_cfg.h` nella posizione che si preferisce per fornire la configurazione di default per Arm2D. La cartella che contiene `arm_2d_cfg.h` dev'essere aggiunta al path dell'include.
5. Il repository Arm2D contiene diversi esempi e template; tuttavia, assicurarsi che venga compilato solo `Arm-2D/Library/Source`.
6. Anche la libreria CMSIS DSP deve essere aggiunta al progetto. Si può utilizzare CMSIS-PACKS o aggiungerla manualmente.
7. Per prestazioni migliori, abilitare LTO (Link Time Optimization) e usare `-Omax` o `-Ofast`.
8. Arm2D tenta di leggere/scrivere più dati con una singola istruzione. Pertanto, è importante utilizzare la memoria più veloce (ad esempio, BSS o TCM) per il buffer di LVGL per evitare colli di bottiglia nella larghezza di banda della memoria.
9. Abilitare `LV_USE_DRAW_ARM2D_SYNC 1` e `LV_USE_DRAW_SW_ASM LV_DRAW_SW_ASM_HELIUM` in `lv_conf.h`.

Accelerazione Neon

Alcuni processori ARM Cortex-A e Cortex-R con architettura ARMv7 e tutti i processori ARM Cortex-A e Cortex-R con architettura ARMv8 supportano il set di istruzioni Neon SIMD <<https://www.arm.com/technologies/neon>>. LVGL ha un supporto integrato per migliorare le prestazioni del rendering software utilizzando le istruzioni Neon.

Supporto per l'Architettura

Sono supportate sia le architetture ARM a 32 bit che quelle a 64 bit. È sufficiente impostare `LV_USE_DRAW_SW_ASM` a `LV_DRAW_SW_ASM_NEON` in `lv_conf.h`.

Nota

Tutti i processori ARM Cortex-A e Cortex-R a 64 bit includono il supporto Neon come funzionalità obbligatoria a partire dalle specifiche dell'architettura ARMv8. Questo rende l'accelerazione Neon universalmente disponibile su tutte le piattaforme ARM a 64 bit, comprese le architetture ARM attuali e future.

GPU Arm-2D

Arm-2D non è una GPU, ma **un livello di astrazione per GPU 2D dedicato ai microcontrollori**. Supporta tutti i processori Cortex-M, dal Cortex-M0 al più recente Cortex-M85.

Arm-2D accelera LVGL9 con due modalità: **Modalità Sincrona** e **Modalità Asincrona**.

- Quando sono disponibili **Helium** e **ACI (Arm Custom Instruction)**, si consiglia di utilizzare la **Modalità Sincrona** per accelerare LVGL.
- Quando sono disponibili GPU 2D basate su Arm-2D, ad esempio **GPU 2D basate su DMAC-350**, si consiglia di utilizzare la **Modalità Asincrona** per accelerare LVGL.

Arm-2D è un progetto open source su GitHub. Per ulteriori informazioni, consultare: <https://github.com/ARM-software/Arm-2D>.

Come si Usa

In generale:

- è possibile impostare la macro `LV_USE_DRAW_ARM2D_SYNC` a 1 e `LV_DRAW_SW_ASM` a `LV_DRAW_SW_ASM_HELIUM` in `lv_conf.h` per abilitare l'accelerazione sincrona Arm-2D per LVGL.
- È possibile impostare la macro `LV_USE_DRAW_ARM2D_ASYNC` a 1 in `lv_conf.h` per abilitare l'accelerazione asincrona Arm-2D per LVGL.

Se si utilizza CMSIS-Pack per il deploy di LVGL. Non è necessario definire manualmente la macro `LV_USE_DRAW_ARM2D_SYNC`, ma `lv_conf_cmsis.h` controllerà l'ambiente e imposterà `LV_USE_DRAW_ARM2D_SYNC` di conseguenza.

Considerazioni Progettuali

Come accennato in precedenza, Arm-2D è un layer di astrazione per GPU 2D; pertanto, se non è disponibile un acceleratore o un set di istruzioni dedicato (come Helium o ACI) per Arm-2D, l'aumento delle prestazioni per LVGL è trascurabile (a volte addirittura peggiore) per i normali processori Cortex-M.

Consigliamo vivamente di abilitare l'accelerazione Arm-2D per LVGL quando:

- I processori target sono **Cortex-M55, Cortex-M52 e Cortex-M85**
- I processori target supportano **Helium**.
- Il fornitore del dispositivo fornisce un driver compatibile con ARM-2D per i propri acceleratori 2D proprietari e/o ACI (Arm Customized Instruction).
- Il dispositivo target contiene **DMAC-350**

Esempi

- Un progetto MDK basato su Cortex-M55 (supporta Helium), disponibile per l'emulazione PC.

API

4.7.3 Espressif

Panoramica

Informazioni

Espressif Systems è un produttore di chip "fabless" che realizza la serie ESP32 di "system on chip". Questi chip possono essere basati su architetture Xtensa o Risc-V e presentano il set comune di periferiche analogiche e digitali di un microcontrollore generico, combinato con un sottosistema radio in grado di gestire uno stack Bluetooth e/o WiFi.

Espressif Systems fornisce anche il suo kit di sviluppo software (SDK) completo, ovvero l'IDF, per consentire agli utenti di concentrarsi esclusivamente sulle proprie applicazioni. Tutti gli altri supporti di terze parti, come Arduino, Zephyr, NuttX e Micropython, sono basati sull'IDF o su alcuni dei suoi componenti.

Sviluppo di Applicazioni

LVGL è supportato dall'SDK di Espressif, ovvero l'IDF, come accennato in precedenza. Pertanto, la serie di chip ESP32 è supportata da diversi set di framework, definiti da Espressif "progetti di terze parti".

Questa guida illustrerà l'utilizzo di LVGL utilizzando il kit di sviluppo software Espressif-IDF. Sebbene ESP32 sia supportato su altri framework che includono anche LVGL, si consiglia all'utente di consultare le seguenti pagine:

- [Framework Arduino](#)
- [Platformio](#)
- [MicroPython](#)
- [NuttX RTOS](#)
- [Zephyr RTOS](#)

Nelle pagine sopra riportate, gli utenti potranno consultare le rispettive istruzioni per iniziare su queste piattaforme, dove ESP32 e altri chip sono astratti dal framework.

Progetti pronti all'uso

Per un rapido avvio con LVGL ed ESP32, LVGL gestisce un progetto demo compatibile con diverse schede ESP32 nel suo [Porting LV per ESP32 <https://github.com/lvgl/lv_esp_idf>](#).

Fare riferimento ai file README.md in tale repository per istruzioni rapide di compilazione e flashing.

Questi progetti demo utilizzano i Board Support Package (BSP) di Espressif. Ulteriori BSP ed esempi sono disponibili nel repository [esp-bsp](#).

Supporto LVGL per periferiche grafiche ESP32

Alcuni chip ESP32, come la famiglia ESP32P4, hanno il supporto integrato per la gestione del display tramite interfacce standard come RGB e MIPI. L'IDF Espressif (esp-idf) fornisce i driver necessari, lasciando all'utente la responsabilità di integrarli nel sottosistema di visualizzazione LVGL.

Espressif, tramite il suo sistema di gestione dei componenti, fornisce un componente di porting LVGL pronto all'uso, che rappresenta il metodo consigliato e preferibile per integrare i dispositivi di input e output dal chip ESP32 al sottosistema Display LVGL. Questo componente è trattato in [Aggiungere LVGL a un progetto ESP32 IDF](#)

Oltre al controller del display, alcuni chip ESP32 offrono periferiche di accelerazione hardware grafica come:

- [Supporto per la Direct Memory Access 2D \(DMA2D\)](#)
- [Supporto per il Pixel Processing Accelerator](#)

Aggiungere LVGL a un progetto ESP32 IDF

LVGL può essere utilizzato e configurato come componente standard [ESP-IDF](#).

Se non si ha familiarità con ESP-IDF, seguire le istruzioni nella [ESP-IDF Programming guide](#) per installare e configurare ESP-IDF sul computer.

Utilizzo di LVGL nel proprio progetto ESP-IDF

Il modo più semplice per integrare LVGL nel proprio progetto ESP-IDF è tramite il componente [esp_lvgl_port](#). Questo componente, utilizzato nei progetti demo menzionati sopra, fornisce funzioni di supporto per una facile installazione di LVGL e driver di display. Inoltre, può aggiungere il supporto per touch, encoder rotativi, pulsanti o ingressi USB HID. Semplifica il risparmio energetico, la rotazione dello schermo e altre funzionalità specifiche della piattaforma.

esp_lvgl_port supporta le versioni LVGL 8 e 9 ed è compatibile con ESP-IDF v4.4 e successive. Per aggiungerlo al progetto, si usa il seguente comando:

```
idf.py add-dependency "espressif/esp_lvgl_port^2.3.0"
```

Per default, esp_lvgl_port dipende dall'ultima versione stabile di LVGL, quindi non sono necessari passaggi aggiuntivi per i nuovi progetti. Se è richiesta una versione specifica di LVGL, la si deve specificare nel progetto per evitare aggiornamenti automatici. LVGL può essere utilizzato anche senza esp_lvgl_port, come descritto di seguito.

Ottenere LVGL

LVGL è distribuito tramite [ESP Registry](#), dove vengono caricate tutte le release LVGL. Se non si desidera utilizzare esp_lvgl_port, è possibile aggiungere LVGL component al progetto con il seguente comando:

```
idf.py add-dependency "lvgl/lvgl^9.*"
```

Modificare la parte ^9.* in base ai requisiti di versione LVGL. Ulteriori informazioni sulle specifiche di versione sono disponibili nella documentazione di [IDF Component Manager](#). Durante la build successiva, il componente LVGL verrà recuperato dal registro dei componenti e aggiunto al progetto.

Uso avanzato: Utilizzare LVGL come componente locale

Per lo sviluppo e il test di LVGL, potrebbe essere utile utilizzare LVGL come componente locale anziché dall'ESP Registry, che offre solo versioni rilasciate e non consente modifiche locali. Per farlo, clonare LVGL nel proprio progetto con il seguente comando:

```
git submodule add https://github.com/lvgl/lvgl.git components/lvgl
```

Nota

Tutti i componenti da \${project_dir}/components vengono aggiunti automaticamente alla build.

Integrazione del Display

Per un progetto LVGL di successo, è necessario un driver per il display e, facoltativamente, un driver per il touch. Espressif fornisce questi driver basati sul suo componente [esp_lcd](#).

- esp_lcd supporta nativamente alcuni [display di base](#)
- Altri display sono mantenuti nel repository [esp-bsp repository](#) e vengono caricati nell'ESP Registry
- I driver touch sono mantenuti nel repository [esp-bsp repository](#) e vengono caricati nell'ESP Registry

Questi componenti condividono un'API pubblica comune, semplificando la migrazione dei progetti tra diversi driver di display e touch.

Per aggiungere un driver di display o touch al progetto, utilizzare un comando come:

```
idf.py add-dependency "espressif/esp_lcd_gc9a01^2.0.0"
```

Configurazione

Per configurare LVGL, avviare il menù di configurazione con `idf.py menuconfig` nella directory principale del progetto. Accedere a `Component config` e poi a `LVGL configuration`.

Inoltre, l'utente può rendere permanenti, o di default, le impostazioni LVGL attuali per il progetto corrente. Tutto ciò che serve è creare un file nella root del progetto chiamato `sdkconfig.defaults` e spostare i simboli `CONFIG_LV_` in quel file.

È possibile creare file di configurazione predefiniti per ogni chip creando un file di configurazione di default che inizia con la variante del chip, ad esempio `sdkconfig.esp32p4` applicherà la configurazione predefinita solo per un progetto IDF ESP32-P4.

Avvio del componente LVGL

Una volta configurati il progetto IDF e il componente LVGL, tutto il processo di inizializzazione all'interno del codice sarà pronto per l'uso; tuttavia, l'utente dovrà avviare manualmente il sottosistema LVGL per IDF chiamando `bsp_display_start()`, o `lvgl_port_init()` se LVGL è stato configurato manualmente, ad esempio senza utilizzare il componente `esp_bsp`.

Dopo aver chiamato questa funzione, LVGL verrà eseguito in background; ovvero, a differenza dell'approccio consueto, non è necessario chiamare periodicamente `lv_timer_handler()`, questa funzione viene chiamata da un task in background gestito dall'IDF.

```
void app_main(void)
{
    bsp_display_start();
    bsp_display_backlight_on();

    bsp_display_lock(0);
    lv_demo_benchmark();
    bsp_display_unlock();
}
```

Nei casi in cui `esp_bsp` non viene utilizzato, è possibile richiamare direttamente la porta ESP-LVGL:

```
void app_main(void)
{
    const lvgl_port_cfg_t lvgl_cfg = ESP_LVGL_PORT_INIT_CONFIG();
    esp_err_t err = lvgl_port_init(&lvgl_cfg);

    lv_demo_benchmark();
}
```

Compilazione e Flashing

Compilare un progetto IDF che prevede l'utilizzo di LVGL è simile a qualsiasi altro progetto: utilizzando IDF tramite la riga di comando, l'utente può combinare diversi comandi in un unico prompt:

```
idf.py build flash monitor
```

Dopo il flashing, la console del monitor verrà avviata automaticamente.

Supporto per la Direct Memory Access 2D (DMA2D)

Panoramica

Alcuni chip di Espressif, come la famiglia ESP32-P4, sono dotati di una periferica che migliora la copia di dati 2D da una posizione all'altra, incluso l'output di questi dati 2D su altre periferiche.

Questa periferica è l'accesso diretto alla memoria 2D, o DMA2D; l'SDK Espressif, l>IDF, offre un driver completo per DMA2D che viene automaticamente abilitato sui chip supportati.

Uno dei suoi ruoli principali è quello di supportare il Pixel Processor Accelerator (PPA), utilizzato per copiare i dati dell'immagine sorgente sul client PPA desiderato, suddividendoli in blocchi di dimensioni fisse, chiamati "burst". Il DMA2D viene inoltre utilizzato per prelevare i blocchi di output dal client PPA e copiarli nel buffer di destinazione, ovvero nel frame buffer del display.

Ulteriori informazioni su PPA sono disponibili in: [Supporto per il PPA \(Pixel Processing Accelerator\)](#).

Interazione col DMA2D

LVGL non interagisce direttamente con DMA2D; il porting del driver del display di LVGL di Espressif utilizza DMA2D per copiare il buffer di disegno di destinazione in quello del display senza l'intervento della CPU. Sebbene questa opzione sia disponibile, l'utente è responsabile di abilitarla esplicitamente sul driver del display del porting del componente LVGL.

Per abilitarla, l'utente deve impostare `CONFIG_BSP_DISPLAY_LVGL_AVOID_TEAR` nel file `sdkconfig.defaults`, che indicherà al driver di utilizzare DMA2D per ottimizzare il trasferimento. Si noti che l'abilitazione di questa opzione sarà disponibile solo quando si utilizza la memoria PSRAM e la modalità doppio buffer, altrimenti verrà generato un errore del compilatore.

Se l'utente sta configurando il display manualmente, è possibile abilitare DMA2D in fase di esecuzione impostando il flag DMA2D a true:

```

/* Gets the default configuration for the MIPI display controller */
esp_lcd_dpi_panel_config_t dpi_config = EK79007_1024_600_PANEL_60HZ_CONFIG(LCD_COLOR_PIXEL_FORMAT_RGB888);

/* Explicitly set the DMA2D to assist the buffer transfer */
dpi_config.flags.use_dma2d = true;

ek79007_vendor_config_t vendor_config = {
    .miipi_config = {
        .dsi_bus = miipi_dsi_bus,
        .dpi_config = &dpi_config,
    },
};

esp_lcd_panel_dev_config_t lcd_dev_config = {
    .reset_gpio_num = UI_FIRMWARE_PIN_NUM_LCD_RST,
    .rgb_ele_order = LCD_RGB_ELEMENT_ORDER_RGB,
    .bits_per_pixel = 24,
    .vendor_config = &vendor_config,
};

/* Configures and enable the display */
ESP_ERROR_CHECK(esp_lcd_new_panel_ek79007(mipi_dbi_io, &lcd_dev_config, &mipi_dpi_panel));

```

Supporto per il PPA (Pixel Processing Accelerator)

Panoramica

Alcune serie di chip ESP32, come l'ESP32-P4, supportano l'hardware Pixel Processing Accelerator (PPA), in grado di accelerare le operazioni di riempimento e fusione delle immagini. Questa periferica funziona con l'hardware DMA-2D, responsabile dello spostamento dei buffer di input/output da/verso il motore di elaborazione PPA.

Dispositivi supportati

I target Espressif che supportano il PPA sono:

- Le serie ESP32-P4.

Utilizzo dell'unità di disegno PPA LVGL sul progetto ESP-IDF

LVGL supporta, a livello sperimentale, l'accelerazione del riempimento e della fusione delle immagini tramite PPA. L'utente può abilitarla nel proprio `sdkconfig.defaults` aggiungendo la seguente opzione per abilitare l'unità di disegno PPA in combinazione con il renderer software:

```
CONFIG_LV_USE_PPA=y
```

Salvare il file e poi ri-eseguire la build del progetto. Sarà sufficiente aggiungere il codice PPA e l'esecuzione inizierà automaticamente, quindi non sono necessari ulteriori passaggi dal punto di vista del codice utente.

Benchmarking

Eseguendo `lv_demo_benchmark` da LVGL, l'utente può confrontare le prestazioni con PPA abilitato rispetto al rendering software puro. Utilizzando l'ultima versione di LVGL e il porting del componente LVGL, si può osservare un risparmio medio del 30% del tempo di rendering per le attività di disegno come il riempimento di immagini e rettangoli. In alcuni casi, per il riempimento puro su multipli interi della dimensione del display, è possibile osservare un aumento della velocità fino a 9 volte quando il PPA è abilitato.

Limitazioni

Si noti che il PPA è a livello sperimentale e sono previsti alcuni miglioramenti delle prestazioni nelle attività di disegno relative alla copia o al riempimento di rettangoli, mentre per la fusione delle immagini, sebbene operativa, non si registrano miglioramenti significativi. La causa iniziale, secondo la sezione PPA del manuale di riferimento, è dovuta alla larghezza di banda della memoria DMA-2D.

Utilizzo delle funzionalità del PPA del componente LVGL di Espressif

Il porting del componente LVGL IDF di Espressif offre anche accelerazione hardware per le operazioni di visualizzazione: una volta che il rendering LVGL completa un'operazione e il buffer di disegno è già stato gestito dal driver del display, il componente Espressif può opzionalmente eseguire il mirroring e la rotazione dei dati renderizzati tramite l'hardware.

Il driver del display LVGL per Espressif legge il descrittore dei dati disegnati per verificare se rotazione e mirroring sono impostati come flag nelle proprietà dei dati; in caso positivo, prima di inviare i dati al display, il mirroring e la rotazione vengono eseguiti utilizzando il client di rotazione e mirroring del PPA.

Rispetto alla rotazione software pura, l'utilizzo del PPA per ruotare gli oggetti può far risparmiare in media fino al 40% del tempo di rendering, mantenendo la CPU inattiva per risparmiare energia o utilizzandola per eseguire altre attività. La rotazione del PPA è asincrona e, una volta avviata, segnalerà l'applicazione tramite l'interrupt dedicato.

Per abilitare tale comportamento sul driver del display, l'accelerazione del PPA deve essere impostata nel progetto `sdkconfig.defaults` tramite il simbolo `CONFIG_LVGL_PORT_ENABLE_PPA`; impostandola su "true" l'assistenza hardware per la rotazione e il mirroring delle immagini verrà automatizzata. Non è richiesto alcun codice aggiuntivo.

Si noti che tale opzione può essere impostata tramite menuconfig utilizzando il comando IDF standard per la configurazione: `idf.py menuconfig`.

API

`lv_draw_ppa.h`

`lv_draw_ppa_private.h`

Suggerimenti e trucchi

Migliorare la velocità di esecuzione di LVGL

In generale, il progetto IDF è configurato per ottimizzare l'immagine finale dell'applicazione in base alle sue dimensioni. Per alcune applicazioni LVGL, questo potrebbe non essere desiderato o potrebbe comportare una scarsa velocità di esecuzione.

In questo caso, è utile impostare alcune opzioni a livello di progetto IDF su `sdkconfig.defaults`, come ad esempio:

`CONFIG_COMPILER_OPTIMIZATION_PERF=`

Questa opzione compilerà l'applicazione dando priorità alle prestazioni, utilizzando istruzioni SIMD ove possibile. È possibile percepire un aumento fino al 30% della velocità di esecuzione complessiva.

È anche possibile velocizzare l'esecuzione del codice critico dell'LVGL, indicando al compilatore di posizionare queste sezioni nell'area IRAM dei chip ESP32 impostando la seguente opzione:

`CONFIG_LV_ATTRIBUTE_FAST_MEM_USE_IRAM=`

È anche possibile impostare la CPU affinché funzioni sempre alla massima velocità impostando l'opzione `CONFIG_ESP_DEFAULT_CPU_FREQ_MHZ_`. Il valore della frequenza varia da chip a chip, ad esempio le famiglie P4 supportano 360 MHz:

`CONFIG_ESP_DEFAULT_CPU_FREQ_MHZ_360`

E ESP32/ESP32-S3 supportano 240 MHz:

`CONFIG_ESP_DEFAULT_CPU_FREQ_MHZ_240`

Si noti che alcune di queste opzioni devono essere abilitate impostando le opzioni sperimentali IDF su true:

`CONFIG_IDF_EXPERIMENTAL_FEATURES=`

Configurazione della PSRAM sui dispositivi supportati da ESP32

Alcuni dei chip ESP32 di fascia alta dispongono di una memoria esterna sul proprio modulo, la PSRAM (Pseudo-Static Random Access Memory). In generale, sul chip sono presenti valori da 4 a 16 MB e LVGL può utilizzare una parte di questa memoria per:

- Copiare oggetti di sola lettura dalla Flash alla PSRAM per aumentare la velocità.
- Utilizzare la modalità diretta con doppio buffer anche su ESP32 che non dispone di un controller del display integrato.

In entrambi gli scenari, il risultato si tradurrà in un tempo inferiore per il trasferimento dei dati sul display, con conseguente aumento del frame rate. Per abilitare l'utilizzo della PSRAM, l'utente deve:

```
CONFIG_SPIRAM=y
CONFIG_SPIRAM_MODE_HEX=y
CONFIG_SPIRAM_USE=y
CONFIG_SPIRAM_ALLOW_BSS_SEG_EXTERNAL_MEMORY=y
CONFIG_SPIRAM_RODATA=y
```

Queste opzioni possono risiedere nel progetto IDF `sdkconfig.defaults`, l'ultima opzione.

L'applicazione va in "crash" quando si abilita PPA.

È possibile che un'applicazione inizi a bloccarsi perché l'utente ha abilitato l'opzione `CONFIG_LV_USE_PPA`. Il sintomo tipico è la comparsa di un messaggio sulla console del monitor che indica un errore quando esp32 richiama la funzione `esp_msync`.

Questo accade perché PPA accetta solo blocchi di dati allineati alla dimensione della linea L1 della cache, ovvero 64 byte. Sebbene l'unità di disegno PPA gestisca l'allineamento del buffer sorgente, anche l'area del buffer di disegno target deve essere allineata, altrimenti il trasferimento da PPA a quest'ultimo potrebbe non riuscire. Per evitare questo comportamento, è utile impostare `CONFIG_LV_DRAW_BUF_ALIGN` su un multiplo della dimensione della linea L1 della cache, ovvero impostarne il valore su 64 invece del 4 di default.

```
CONFIG_LV_DRAW_BUF_ALIGN=64
```

Il log del monitor EPS32-P4 segnala "buffer underrun" e diminuzioni del frame rate

Nei casi in cui la PSRAM è abilitata e viene utilizzato il PPA, è comune osservare un degrado del frame rate seguito da un messaggio nel log che segnala che il buffer di visualizzazione sarà "underrun". Questo comportamento si verifica perché, a seconda della versione di IDF, la PSRAM non è stata abilitata alla massima velocità supportata.

Per correggere questo comportamento, è sufficiente aggiungere a `sdkconfig.defaults` la seguente opzione:

```
CONFIG_SPIRAM_SPEED_200M=y
```

Abilitazione dei log LVGL sul progetto IDF

I log LVGL non sono abilitati di default; per farlo, si aggiungono le seguenti opzioni a `sdkconfig.defaults`:

```
CONFIG_LV_USE_LOG=y
CONFIG_LV_LOG_LEVEL_INFO=y
CONFIG_LV_LOG_PRINTF=y
```

Il sottosistema di logging di LVGL si basa sulla presenza di `printf` in ESP-IDF.

Utilizzo del file system in ESP-IDF

ESP-IDF utilizza le funzioni standard C per le operazioni sui file (`fopen`, `fread`) in tutte le sue API relative all'archiviazione. Ciò consente un'interoperabilità perfetta con LVGL quando si abilita la configurazione `LV_USE_FS_STDIO`. Il processo è descritto in dettaglio di seguito, utilizzando SPIFFS come dimostrazione.

- **Decidere quale sistema di archiviazione utilizzare**

ESP-IDF offre molti esempi pronti all'uso, come SPIFFS, SD Card e LittleFS.

- **Riconfigurare il progetto**

Il progetto di esempio dovrebbe essere esaminato per i dettagli, ma in generale le modifiche riguardano:

- Abilitazione del file system STDIO di LVGL nella configurazione

È possibile utilizzare menuconfig:

- * Component config → LVGL configuration → 3rd Party Libraries: enable File system on top of stdio API
- * Poi selezionare Set an upper cased letter on which the drive will accessible e imposta la su 65 (ASCII A)

- * È anche possibile impostare Default driver letter a 65 per ignorare il prefisso nei path dei file.
- Modifica della tabella delle partizioni

La configurazione esatta dipende dalle dimensioni della flash e dalle partizioni esistenti, ma il nuovo risultato finale dovrebbe essere simile a questo:

Tabella 3: Partition Table

nvs	data	nvs	0x9000	0x6000
phy_init	data	phy	0xf000	0x1000
factory	app	factory	0x10000	1400k
storage	data	spiffs		400k

i Nota

Se non si utilizza ancora un file `partition.csv` personalizzato, è possibile aggiungerlo tramite menuconfig (Partition Table → Partition Table → Custom partition table CSV).

- Applicare le modifiche al sistema di build

Alcuni file system ESP forniscono la generazione automatica da una cartella host tramite CMake. Le righe corrette devono essere copiate in `main/CMakeLists.txt`

i Nota

LittleFS ha dipendenze aggiuntive che devono essere inserite in `main/idf_component.yml`

• Preparare i file delle immagini

Il tool Python `LVGLImage.py` di LVGL può essere utilizzato per convertire le immagini in file binari di "pixel map". Supporta vari formati e compressioni.

Mentre librerie di terze parti (come `LodePNG` e `Tiny JPEG`) consentono di utilizzare file immagine senza conversione.

Dopo aver preparato i file, è necessario spostarli sul dispositivo di destinazione:

- Se attivato correttamente, un file system **SPIFFS** basato sulla cartella `spiffs_image` dovrebbe essere generato automaticamente e successivamente flashato sul dispositivo di destinazione.
- Un meccanismo simile per **LittleFS** usa la cartella `flash_data`, ma è disponibile solo per host Linux.
- Per la **SD Card**, è possibile utilizzare un file browser tradizionale.

• Invocare le chiamate API appropriate nel codice dell'applicazione

Le funzionalità principali richiedono solo poche righe. L'esempio seguente disegna anche l'immagine.

```
#include "esp_spiffs.h"

void lv_example_image_from_esp_fs(void) {
    esp_vfs_spiffs_conf_t conf = {
        .base_path = "/spiffs",
        .partition_label = NULL,
        .max_files = 5,
        .format_if_mount_failed = false
    };
    esp_err_t ret = esp_vfs_spiffs_register(&conf);

    if (ret != ESP_OK) {
        ESP_LOGE(TAG, "Failed to register SPIFFS filesystem");
        return;
    }

    lv_obj_t * obj = lv_image_create(lv_screen_active());
    lv_image_set_src(widget, "A:/spiffs/logo.bin");
    lv_obj_center(widget);
}
```

- **Build e flash**

Dopo aver chiamato `idf.py build flash`, l'immagine dovrebbe essere visualizzata sullo schermo.

1 Nota

Le modifiche apportate da `menuconfig` non vengono tracciate nel repository se il file `sdkconfig` viene aggiunto a `.gitignore`, che è l'impostazione di default per molti progetti ESP-IDF. Per rendere permanente la configurazione, aggiungere le seguenti righe a `sdkconfig.defaults`:

```
CONFIG_PARTITION_TABLE_CUSTOM=y
CONFIG_LV_USE_FS_STDIO=y
CONFIG_LV_FS_STDIO LETTER=65
CONFIG_LV_FS_DEFAULT_DRIVER LETTER=65
```

4.7.4 NXP

Panoramica

NXP ha integrato LVGL nei pacchetti "SDK MCUXpresso" per diversi dei microcontrollori come componente software opzionale, consentendo una facile valutazione e migrazione nella progettazione del prodotto. LVGL è una libreria grafica embedded gratuita e open source con funzionalità che consentono di creare GUI embedded con elementi grafici intuitivi, splendidi effetti visivi e un ridotto utilizzo di memoria. Il framework grafico completo include una varietà di widget da utilizzare nella creazione della GUI e supporta funzioni più avanzate come animazioni e anti-aliasing.

LVGL abilita la grafica nel tool gratuito "GUI Guider". È disponibile per l'uso con i microcontrollori generici e crossover di NXP, fornendo agli sviluppatori uno strumento per creare applicazioni GUI complete e di alta qualità con LVGL.

Creazione di un nuovo progetto con LVGL

Scaricare un [SDK per una board supportata](#) e iniziare a sviluppare la prossima applicazione GUI. È completamente configurato con LVGL (e con supporto PXP/VGLite/G2D se i moduli sono presenti), non è richiesto alcun ulteriore lavoro di integrazione.

Accelerazione HW per piattaforme NXP iMX RT

A seconda della piattaforma RT utilizzata, l'accelerazione può essere eseguita da NXP PXP (PiXel Pipeline) e/o dalla GPU Verisilicon tramite un'API denominata VGLite. Ogni acceleratore ha un proprio contesto che ne consente l'utilizzo sia singolarmente che simultaneamente (in modalità LVGL multithreading).

Accelerazione HW per piattaforme NXP iMX

Sulle piattaforme MPU, l'accelerazione può essere eseguita (indipendentemente dall'hardware) dalla libreria NXP G2D. Questo acceleratore ha un proprio contesto che ne consente l'utilizzo sia singolarmente che simultaneamente con la CPU (in modalità LVGL multithreading).

Acceleratore PXP

Configurazione di base:

- Selezionare il motore PXP NXP in "lv_conf.h": Impostare `LV_USE_PXP` a 1.
- Per utilizzare PXP come unità di disegno, selezionare in "lv_conf.h": Impostare `LV_USE_DRAW_PXP` a 1.
- Per utilizzare PXP per ruotare lo schermo, selezionare in "lv_conf.h": Impostare `LV_USE_ROTATE_PXP` a 1.
- Abilitare le asserzioni PXP in "lv_conf.h": Impostare `LV_USE_PXP_ASSERT` a 1. Esistono alcune asserzioni PXP che possono interrompere l'esecuzione del programma nel caso in cui esistono alcune asserzioni PXP che possono interrompere l'esecuzione del programma nel caso in cui `LV_ASSERT_HANDLER` sia impostato su `while(1);` ("Halt" per default). In caso contrario, verrà registrato solo un messaggio di errore tramite `LV_LOG_ERROR`.

- Se è definito il simbolo `SDK_OS_FREE_RTOS`, verrà utilizzata l'implementazione di FreeRTOS, altrimenti verrà incluso codice "bare-metal".

Inizializzazione di base:

L'inizializzazione del disegno PXP viene eseguita automaticamente in `lv_init()` una volta che il PXP è abilitato come unità di disegno o per ruotare lo schermo, non è richiesto alcun codice utente:

```
#if LV_USE_DRAW_PXP || LV_USE_ROTATE_PXP
    lv_draw_pxp_init();
#endif
```

Durante l'inizializzazione del PXP, verrà creata una nuova unità di disegno `lv_draw_pxp_unit_t` con le callback aggiuntive, se `LV_USE_DRAW_PXP` è impostato a 1:

```
lv_draw_pxp_unit_t * draw_pxp_unit = lv_draw_create_unit(sizeof(lv_draw_pxp_unit_t));
draw_pxp_unit->base_unit.evaluate_cb = _pxp_evaluate;
draw_pxp_unit->base_unit.dispatch_cb = _pxp_dispatch;
draw_pxp_unit->base_unit.delete_cb = _pxp_delete;
```

e verrà generato un thread aggiuntivo `_pxp_render_thread_cb()` per gestire i task di disegno supportati.

```
#if LV_USE_PXP_DRAW_THREAD
    lv_thread_init(&draw_pxp_unit->thread, "pxpdraw", LV_THREAD_PRIO_HIGH, _pxp_render_thread_cb, 2 * 1024, draw_pxp_unit);
#endif
```

Se `LV_USE_PXP_DRAW_THREAD` non è definito, non verrà creato alcun thread di disegno aggiuntivo e il task di disegno PXP verrà eseguito sullo stesso thread principale LVGL.

`_pxp_evaluate()` verrà chiamato dopo la creazione di ogni task e analizzerà se questa è supportata da PXP o meno. Se supportato, all'attività verranno assegnati un punteggio preferito e l'ID dell'unità di disegno. Uno score uguale a 100 è quello di default della CPU. Uno score più piccolo indica che PXP è in grado di disegnarlo più velocemente.

`_pxp_dispatch()` è la callback del dispatcher PXP, che accetta un task pronto per il disegno (con `DRAW_UNIT_ID_PXP` impostato) e la passa all'unità di disegno PXP per l'elaborazione.

`_pxp_delete()` ripulisce l'unità di disegno PXP.

Funzionalità supportate:

Diverse funzionalità di disegno in LVGL possono essere demandate al motore PXP. La CPU è disponibile per altre operazioni mentre PXP è in esecuzione. È necessario un RTOS per bloccare il thread di disegno LVGL e passare a un altro task o sospendere la CPU per risparmiare energia.

Le attività di disegno supportate sono disponibili in "src/draw/nxp/pxp/lv_draw_pxp.c":

```
switch(t->type) {
    case LV_DRAW_TASK_TYPE_FILL:
        lv_draw_pxp_fill(t, t->draw_dsc, &t->area);
        break;
    case LV_DRAW_TASK_TYPE_IMAGE:
        lv_draw_pxp_img(t, t->draw_dsc, &t->area);
        break;
    case LV_DRAW_TASK_TYPE_LAYER:
        lv_draw_pxp_layer(t, t->draw_dsc, &t->area);
        break;
    default:
        break;
}
```

Inoltre, la rotazione dello schermo può essere gestita da PXP:

- Riempire l'area con un colore (senza raggio, senza gradiente) + opacità opzionale.
- Blit [Block Image Transfer] dell'immagine sorgente RGB565/ARGB888/XRGB8888 sulla destinazione. RGB565/RGB888/ARGB888/XRGB8888 + opacità opzionale.
- Ricolorare l'immagine sorgente RGB565.
- Ridimensionare e ruotare (90, 180, 270 gradi) l'immagine sorgente RGB565.
- Fusione dei livelli (con gli stessi formati supportati per il blitting).
- Ruotare lo schermo (90, 180, 270 gradi).

Limitazioni note:

- PXP può ruotare i fotogrammi solo di angoli multipli di 90 gradi.
- La rotazione non è supportata per le immagini non allineate a blocchi di 16x16 pixel. PXP è impostato per elaborare blocchi 16x16 per ottimizzare il sistema in termini di larghezza di banda di memoria e tempo di elaborazione delle immagini. Il motore di output essenzialmente tronca tutti i pixel di output dopo che è stato scritto il numero desiderato di pixel. Quando si ruota un'immagine sorgente e l'output non è divisibile per la dimensione del blocco, i pixel errati potrebbero essere troncati e l'immagine di output finale potrebbe apparire spostata.
- La ricolorazione o la trasformazione di immagini con opacità o canale alfa non possono essere ottenute in una singola configurazione della pipeline PXP. Sarebbero necessari due o più passaggi.
- L'indirizzo del buffer deve essere allineato a 64 byte: impostare `LV_DRAW_BUF_ALIGN` a 64 in "lv_conf.h". Non è richiesto alcun allineamento di "stride": impostare `LV_DRAW_BUF_STRIDE_ALIGN` a 1 in "lv_conf.h".

Setup del progetto:

- Aggiungere i file sorgente relativi a PXP (e i relativi header, se disponibili) al progetto:
 - "src/draw/nxp/pxp/lv_draw_buf_pxp.c": callback del buffer di disegno
 - "src/draw/nxp/pxp/lv_draw_pxp_fill.c": area di riempimento
 - "src/draw/nxp/pxp/lv_draw_pxp_img.c": immagine blit (con ricolorazione o trasformazione opzionale)
 - "src/draw/nxp/pxp/lv_draw_pxp_layer.c": fusione dei livelli
 - "src/draw/nxp/pxp/lv_draw_pxp.c": inizializzazione dell'unità di disegno
 - "src/draw/nxp/pxp/lv_pxp_cfg.c": init,_deinit, run/wait del dispositivo PXP
 - "src/draw/nxp/pxp/lv_pxp_osa.c": astrazione del sistema operativo (FreeRTOS o "bare-metal")
 - "src/draw/nxp/pxp/lv_pxp_utils.c": helper delle funzioni
- Il codice relativo a PXP dipende da due driver forniti dall'SDK MCU. Questi driver devono essere aggiunti al progetto:
 - fsl_pxp.c: Driver PXP
 - fsl_cache.c: Funzioni di gestione della cache della CPU

Configurazione di default di PXP:

- L'implementazione dipende da più funzioni specifiche del sistema operativo. La struttura `pxp_cfg_t` con puntatori di callback viene utilizzata come parametro per la funzione `lv_pxp_init()`. Implementazione di default per FreeRTOS in lv_pxp_osa.c.
 - `pxp_interrupt_init()`: Inizializza l'interrupt PXP (configurazione HW, configurazione del sistema operativo)
 - `pxp_interrupt_deinit()`: De-inizializza l'interrupt PXP (configurazione HW, configurazione del sistema operativo)
 - `pxp_run()`: Avvia il processo PXP. Utilizza un meccanismo specifico del sistema operativo per bloccare il thread di disegno.
 - `pxp_wait()`: Attende il completamento di PXP.

Acceleratore VGLite

Il motore VGLite può gestire funzionalità di disegno aggiuntive in LVGL. La CPU è disponibile per altre operazioni mentre VGLite è in esecuzione. È necessario un RTOS per bloccare il thread di disegno LVGL e passare a un altro task o sospendere la CPU per risparmiare energia.

Configurazione di base:

- Selezionare il motore NXP VGLite in "lv_conf.h": Impostare `LV_USE_DRAW_VGLITE` a 1. Il simbolo `SDK_OS_FREE_RTOS` deve essere definito in modo che l'implementazione "osal" [OS Abstraction Layer] del driver FreeRTOS sia abilitata.
- Abilitare le asserzioni VGLite in "lv_conf.h": Impostare `LV_USE_VGLITE_ASSERT` a 1. Le asserzioni VGLite verificheranno il codice di stato dell'API del driver e, in caso di errore, potranno interrompere l'esecuzione del programma nel caso in cui `LV_ASSERT_HANDLER` sia impostato su `while(1);` ("Halt" per default). In caso contrario, verrà registrato solo un messaggio di errore tramite `LV_LOG_ERROR`.

Inizializzazione di base:

Inizializza la GPU VGLite prima di chiamare `lv_init()` specificando la larghezza/altezza della finestra di tassellatura. I valori di default per la larghezza e l'altezza della tassellatura e la dimensione del buffer dei comandi si trovano nel file SDK "vglite_support.h".

```
#if LV_USE_DRAW_VGLITE
#include "vg_lite.h"
#include "vglite_support.h"
#endif
...
#if LV_USE_DRAW_VGLITE
if(vg_lite_init(DEFAULT_VG_LITE_TW_WIDTH, DEFAULT_VG_LITE_TW_HEIGHT) != VG_LITE_SUCCESS)
{
    PRINTF("VGLite init error. STOP.");
    vg_lite_close();
    while (1)
        ;
}
if (vg_lite_set_command_buffer_size(VG_LITE_COMMAND_BUFFER_SIZE) != VG_LITE_SUCCESS)
{
    PRINTF("VGLite set command buffer. STOP.");
    vg_lite_close();
    while (1)
        ;
}
#endif
```

L'inizializzazione del disegno di VGLite viene eseguita automaticamente in `lv_init()` una volta abilitato VGLite, non è richiesto alcun codice utente:

```
#if LV_USE_DRAW_VGLITE
    lv_draw_vglite_init();
#endif
```

Durante l'inizializzazione di VGLite, verrà creata una nuova unità di disegno `lv_draw_vglite_unit_t` con le callback aggiuntive:

```
lv_draw_vglite_unit_t * draw_vglite_unit = lv_draw_create_unit(sizeof(lv_draw_vglite_unit_t));
draw_vglite_unit->base.unit.evaluate_cb = _vglite_evaluate;
draw_vglite_unit->base.unit.dispatch_cb = _vglite_dispatch;
draw_vglite_unit->base.unit.delete_cb = _vglite_delete;
```

e verrà generato un thread aggiuntivo `_vglite_render_thread_cb()` per gestire le attività di disegno supportate.

```
#if LV_USE_VGLITE_DRAW_THREAD
    lv_thread_init(&draw_vglite_unit->thread, "vglitedraw", LV_THREAD_PRIO_HIGH, _vglite_render_thread_cb, 2 * 1024, draw_vglite_unit);
#endif
```

Se `LV_USE_VGLITE_DRAW_THREAD` non è definito, non verrà creato alcun thread di disegno aggiuntivo e l'attività di disegno di VGLite verrà eseguita sullo stesso thread principale di LVGL.

`_vglite_evaluate()` verrà chiamato dopo la creazione di ogni task e analizzerà se questa è supportata da VGLite o meno. Se supportato, all'attività verranno assegnati un punteggio preferito e l'ID dell'unità di disegno. Uno score uguale a 100 è quello di default della CPU. Uno score più piccolo indica che VGLite è in grado di disegnarlo più velocemente.

`_vglite_dispatch()` è la callback del dispatcher VGLite, che accetta un task pronto per il disegno (con `DRAW_UNIT_ID_VGLITE` impostato) e la passa all'unità di disegno VGLite per l'elaborazione.

`_vglite_delete()` pulirà l'unità di disegno VGLite.

Configurazione avanzata:

- Abilitare la divisione blit VGLite in "lv_conf.h": Impostare `LV_USE_VGLITE_BLIT_SPLIT` a 1. Abilitando la soluzione alternativa per la divisione blit si attenuerà qualsiasi problema di degradazione della qualità su schermi con dimensioni superiori a 352 pixel.

```
#define VGLITE_BLIT_SPLIT_THR 352
```

- Per default, la soglia del "blit split" è impostata su 352. I blit [Block Image Transfer] con larghezza o altezza superiori a questo valore verranno eseguiti in più passaggi. Il valore deve essere un multiplo dell'allineamento "stride" in px. Per la maggior parte dei formati colore, l'allineamento è di 16 px (ad eccezione dei formati indice). La trasformazione non sarà più supportata con la divisione blit.
- Abilitare l'attività di disegno VGLite in modo sincrono in "lv_conf.h": Impostare `LV_USE_VGLITE_DRAW_ASYNC` a 1. È possibile mettere in coda più task di disegno e scaricarle una volta sulla GPU in base allo stato di inattività della GPU. Se la GPU è occupata, l'attività verrà messa in coda e il dispatcher VGLite richiederà una nuova attività disponibile. Se la GPU è inattiva, la coda con eventuali attività in sospeso verrà scaricata sulla GPU. Lo stato di completamento dell'attività di disegno verrà inviato al thread LVGL principale in modo asincrono.

Funzionalità supportate:

Diverse funzionalità di disegno in LVGL possono essere scaricate sul motore VGLite. La CPU è disponibile per altre operazioni mentre la GPU è in esecuzione. È necessario un RTOS per bloccare il thread di disegno LVGL e passare a un'altra attività o sospendere la CPU per risparmiare energia.

Le attività di disegno supportate sono disponibili in "src/draw/nxp/vglite/lv_draw_vglite.c":

```
switch(t->type) {
    case LV_DRAW_TASK_TYPE_LABEL:
        lv_draw_vglite_label(t, t->draw_dsc, &t->area);
        break;
    case LV_DRAW_TASK_TYPE_FILL:
        lv_draw_vglite_fill(t, t->draw_dsc, &t->area);
        break;
    case LV_DRAW_TASK_TYPE_BORDER:
        lv_draw_vglite_border(t, t->draw_dsc, &t->area);
        break;
    case LV_DRAW_TASK_TYPE_IMAGE:
        lv_draw_vglite_img(t, t->draw_dsc, &t->area);
        break;
    case LV_DRAW_TASK_TYPE_ARC:
        lv_draw_vglite_arc(t, t->draw_dsc, &t->area);
        break;
    case LV_DRAW_TASK_TYPE_LINE:
        lv_draw_vglite_line(t, t->draw_dsc);
        break;
    case LV_DRAW_TASK_TYPE_LAYER:
        lv_draw_vglite_layer(t, t->draw_dsc, &t->area);
        break;
    case LV_DRAW_TASK_TYPE_TRIANGLE:
        lv_draw_vglite_triangle(t, t->draw_dsc);
        break;
    default:
        break;
}
```

Tutte le operazioni seguenti possono essere eseguite anche con opacità opzionale.

- Riempire l'area con il colore (con raggio o gradiente).
- Eseguire il blit [Block Image Transfer] dell'immagine sorgente (qualsiasi formato da `_vglite_src_cf_supported()`) sulla destinazione (qualsiasi formato da `_vglite_dest_cf_supported()`).
- Ricolorare l'immagine sorgente.
- Ridimensionare e ruotare (90, 180, 270 gradi) l'immagine sorgente RGB565.
- Fusione dei livelli (con gli stessi formati supportati per il blitting).
- Disegnare lettere (lettere bitmap blit / font raster).
- Disegnare bordi completi (`LV_BORDER_SIDE_FULL`).
- Disegnare archi (con bordi arrotondati).
- Disegnare linee (con trattini o bordi arrotondati).
- Disegnare triangoli colorati (con gradiente).

Limitazioni note:

- Allineamento dell'immagine sorgente: Il requisito di allineamento dei byte per un pixel dipende dal formato specifico del pixel. Sia l'indirizzo del buffer che il passo del buffer devono essere allineati. Come regola generale, l'allineamento è impostato su 16 pixel. Questo fa sì che l'allineamento dell'indirizzo del buffer sia di 32 byte per RGB565 e di 64 byte per ARGB8888.
- Per la destinazione del pixel engine (PE), l'allineamento dovrebbe essere di 64 byte per tutti i layout di buffer a mosaico (4x4). Il pixel engine non ha requisiti di allineamento aggiuntivi per i layout di buffer lineari (VG_LITE_LINEAR).

Setup del progetto:

- Aggiungere i file sorgente correlati a VGLite (e gli header corrispondenti, se disponibili) al progetto:
 - "src/draw/nxp/vglite/lv_draw_buf_vglite.c": callback del buffer di disegno
 - "src/draw/nxp/vglite/lv_draw_vglite_arc.c": disegno dell'arco
 - "src/draw/nxp/vglite/lv_draw_vglite_border.c": disegno del bordo
 - "src/draw/nxp/vglite/lv_draw_vglite_fill.c": riempimento dell'area
 - "src/draw/nxp/vglite/lv_draw_vglite_img.c": immagine blit (con ricolorazione o trasformazione facoltativa)
 - "src/draw/nxp/vglite/lv_draw_vglite_label.c": disegno dell'etichetta
 - "src/draw/nxp/vglite/lv_draw_vglite_layer.c": fusione livelli
 - "src/draw/nxp/vglite/lv_draw_vglite_line.c": disegno linee
 - "src/draw/nxp/vglite/lv_draw_vglite_triangle.c": disegna triangoli
 - "src/draw/nxp/vglite/lv_draw_vglite.c": inizializzazione dell'unità di disegno
 - "src/draw/nxp/vglite/lv_vglite_buf.c": init/get buffer vglite
 - "src/draw/nxp/vglite/lv_vglite_matrix.c": imposta la matrice vglite
 - "src/draw/nxp/vglite/lv_vglite_path.c": crea i dati del percorso vglite
 - "src/draw/nxp/vglite/lv_vglite_utils.c": helper delle funzioni

Acceleratore G2D

Configurazione di base:

- Selezionare il motore NXP G2D in "lv_conf.h": Impostare `LV_USE_G2D` a 1.
- Per utilizzare G2D come unità di disegno, selezionare in "lv_conf.h": Impostare `LV_USE_DRAW_G2D` a 1.
- Abilitare le asserzioni G2D in "lv_conf.h": Impostare `LV_USE_G2D_ASSERT` a 1. Esistono alcune asserzioni G2D che possono interrompere l'esecuzione del programma nel caso in cui `LV_ASSERT_HANDLER` sia impostato su `while(1);` ("Halt" per default). In caso contrario, verrà registrato solo un messaggio di errore tramite `LV_LOG_ERROR`.

Inizializzazione di base:

L'inizializzazione del disegno G2D viene eseguita automaticamente in `lv_init()` una volta che G2D è abilitato come unità di disegno, non è richiesto alcun codice utente:

```
#if LV_USE_DRAW_G2D
    lv_draw_g2d_init();
#endif
```

Durante l'inizializzazione di G2D, verrà creata una nuova unità di disegno `lv_draw_g2d_unit_t` con le callback aggiuntive, se `LV_USE_DRAW_G2D` è impostato a 1:

```
lv_draw_g2d_unit_t * draw_g2d_unit = lv_draw_create_unit(sizeof(lv_draw_g2d_unit_t));
draw_g2d_unit->base_unit.evaluate_cb = _g2d_evaluate;
draw_g2d_unit->base_unit.dispatch_cb = _g2d_dispatch;
draw_g2d_unit->base_unit.delete_cb = _g2d_delete;
```

e verrà generato un thread aggiuntivo `_g2d_render_thread_cb()` per gestire le attività di disegno supportate.

```
#if LV_USE_G2D_DRAW_THREAD
    lv_thread_init(&draw_g2d_unit->thread, LV_THREAD_PRIO_HIGH, _g2d_render_thread_cb, 2 * 1024, draw_g2d_unit);
#endif
```

Se `LV_USE_G2D_DRAW_THREAD` non è definito, non verrà creato alcun thread di disegno aggiuntivo e l'attività di disegno G2D verrà eseguita sullo stesso thread principale LVGL.

`_g2d_evaluate()` verrà chiamato dopo la creazione di ogni task e analizzerà se questa è supportata da G2D o meno. Se supportato, all'attività verranno assegnati un punteggio preferito e l'ID dell'unità di disegno. Uno score uguale a 100 è quello di default della CPU. Uno score più piccolo indica che G2D è in grado di disegnarlo più velocemente.

`_g2d_dispatch()` è la callback del dispatcher G2D, che prenderà un'attività pronta per essere disegnata (con `DRAW_UNIT_ID_G2D` impostato) e la passerà all'unità di disegno G2D per l'elaborazione.

`_g2d_delete()` pulirà l'unità di disegno G2D.

Funzionalità supportate:

Diverse funzionalità di disegno in LVGL possono essere demandate al motore G2D. La CPU è disponibile per altre operazioni mentre G2D è in esecuzione. È necessario il sistema operativo Linux per bloccare il thread di disegno LVGL e passare a un altro task o sospendere la CPU per risparmiare energia.

Le attività di disegno supportate sono disponibili in "src/draw/nx/g2d/lv_draw_g2d.c":

```
switch(t->type) {
    case LV_DRAW_TASK_TYPE_FILL:
        lv_draw_g2d_fill(u, t->draw_dsc, &t->area);
        break;
    case LV_DRAW_TASK_TYPE_IMAGE:
        lv_draw_g2d_img(u, t->draw_dsc, &t->area);
        break;
    default:
        break;
}
```

- Riempire l'area con un colore (senza raggio, senza gradiente) + opacità opzionale.
- Blit [Block Image Transfer] dell'immagine sorgente ARGB8888 sulla destinazione. ARGB8888 + opacità opzionale.
- Ridimensionare l'immagine sorgente ARGB8888.

Limitazioni note:

- G2D/PXP può ruotare solo ad angoli di 90°.
- La rotazione non è supportata per le immagini non allineate a blocchi di 16x16 pixel. G2D/PXP è impostato per elaborare blocchi 16x16 per ottimizzare il sistema in termini di larghezza di banda di memoria e tempo di elaborazione delle immagini. Il motore di output essenzialmente tronca tutti i pixel di output dopo che è stato scritto il numero desiderato di pixel. Quando si ruota un'immagine sorgente e l'output non è divisibile per la dimensione del blocco, i pixel errati potrebbero essere troncati e l'immagine di output finale potrebbe apparire spostata.
- La ricolorazione o la trasformazione di immagini con opacità o canale alfa non possono essere ottenute in una singola configurazione della pipeline G2D/PXP. Sarebbero necessari due o più passaggi.
- L'indirizzo del buffer deve essere allineato a 64 byte: impostare `LV_DRAW_BUF_ALIGN` a 64 in "lv_conf.h". Non è richiesto alcun allineamento di "stride": impostare `LV_DRAW_BUF_STRIDE_ALIGN` a 1 in "lv_conf.h".

Setup del progetto:

- Aggiungere i file sorgente relativi a G2D (e i relativi header, se disponibili) al progetto:
 - "src/draw/nxp/g2d/lv_draw_buf_g2d.c": callback del buffer di disegno
 - "src/draw/nxp/g2d/lv_draw_g2d_fill.c": riempimento dell'area
 - "src/draw/nxp/g2d/lv_draw_g2d_img.c": immagine blot (con ricolorazione o trasformazione opzionale)
 - "src/draw/nxp/g2d/lv_draw_g2d.c": inizializzazione dell'unità di disegno
 - "src/draw/nxp/g2d/lv_draw_g2d_buf_map.c": mappa hash per i buffer g2d
 - "src/draw/nxp/g2d/lv_g2d_utils.c": helper delle funzioni

NXP eLCDIF

Panoramica

eLCDIF è una periferica presente su alcuni dispositivi NXP in grado di pilotare pannelli di display tramite l'interfaccia RGB, supporta diverse profondità di colore e, sui dispositivi compatibili con MIPI-DSI, il suo output può essere indirizzato all'interfaccia fisica del display MIPI. Il driver NXP eLCDIF di LVGL è responsabile del "bind" [collegamento] del driver di basso livello dell'SDK MCUX di NXP al sottosistema di visualizzazione LVGL.

Prerequisiti

- Questo driver si basa sulla presenza dell'SDK MCUX di NXP nello stesso progetto.
- Attivare il driver impostando `LV_USE_NXP_ELCDIF` a 1 nel file "`lv_conf.h`".

Utilizzo

Il driver LVGL per eLCDIF presuppone che la piattaforma abbia già configurato il driver di basso livello del display, impostato il pin-mux, i clock, ecc. Richiede inoltre che l'indirizzo base della periferica e la struttura di configurazione siano già impostati.

Il codice seguente illustra l'utilizzo del driver in modalità `LV_DISPLAY_RENDER_MODE_DIRECT`. Si noti che in questa modalità di funzionamento l'applicazione è responsabile dell'allocazione dello spazio dei framebuffer e del loro passaggio al display. Nell'esempio seguente `buffer1` e `buffer2` sono il buffer corrente e quello successivo che verranno copiati sullo schermo, scambiati a ogni operazione di flush (gestita internamente dal driver del display). Si noti inoltre che, in "direct mode", ogni buffer dovrebbe avere spazio sufficiente per contenere almeno le dimensioni dello schermo, ovvero l'altezza per la larghezza per i byte di un pixel (dipendente dall'applicazione o supportato dal display). Nel codice seguente, questa dimensione è rappresentata da `buf_size`.

```
elcdif_rgb_mode_config_t config;
ELCDIF_RgbModeGetDefaultConfig(&config);

lv_display_t * g_disp = lv_nxp_display_elcdif_create_direct(LCDIF, config, buffer1, buffer2, buf_size);
lv_display_set_default(g_disp);
```

Per utilizzare il driver in modalità `LV_DISPLAY_RENDER_MODE_PARTIAL`, è necessario allocare un buffer aggiuntivo, preferibilmente nella regione di memoria più veloce disponibile.

Lo swapping del buffer può essere attivato passando un secondo buffer della stessa dimensione al posto dell'argomento `NULL`. Si noti che in questo caso `BUF_SIZE` deve avere almeno uno spazio sufficiente a contenere dati pari a 1/10 delle dimensioni effettive del display.

```
#define BUF_SIZE (DISPLAY_HEIGHT * DISPLAY_WIDTH / 10 * 2) /*1/10 screen size for RGB565 format*/
static uint8_t partial_draw_buf[BUF_SIZE];
lv_display_t * g_disp = lv_nxp_display_elcdif_create_partial(LCDIF, config, partial_draw_buf, NULL, BUF_SIZE);
```

In fase di runtime, la funzione di gestione eventi del driver eLCDIF deve essere chiamata all'interno del gestore di interrupt eLCDIF. Questa funzione è responsabile della notifica al sottosistema di visualizzazione LVGL del completamento dell'operazione di flush:

```
void eLCDIF_IRQHandler(void)
{
    lv_nxp_display_elcdif_event_handler(g_disp);
}
```

NXP PXP GPU

API

lv_draw_pxp.h

GPU Generica VG-Lite

Si tratta di un'implementazione generica del backend di rendering VG-Lite, progettata per utilizzare l'API generica di VeriSilicon per gestire l'hardware GPU il più possibile.

Anche con diversi produttori di chip, purché utilizzino la stessa versione dell'API VG-Lite come backend di rendering, l'accelerazione del rendering LVGL può essere supportata senza la necessità di adattamenti LVGL.

Configurazione

1. Impostare `LV_USE_DRAW_VG_LITE` a 1 in `lv_conf.h` per abilitare il backend di rendering VG-Lite. Assicurarsi che l'hardware sia stato adattato all'API VG-Lite e che sia stato esposto il path assoluto a `vg_lite.h`, a cui lvgl può fare riferimento direttamente.
2. Per confermare il metodo di inizializzazione della GPU, ci sono due modi:
 - L'SDK richiama autonomamente la funzione di inizializzazione della GPU durante l'avvio del sistema e la GPU è disponibile all'avvio di LVGL; impostare `LV_VG_LITE_USE_GPU_INIT` a 0.
 - LVGL richiama attivamente la funzione di inizializzazione della GPU e l'SDK deve implementare la funzione pubblica `gpu_init()`. LVGL la chiamerà per completare l'inizializzazione hardware della GPU durante l'avvio; impostare `LV_VG_LITE_USE_GPU_INIT` a 1.
3. Impostare la configurazione `LV_VG_LITE_USE_ASSERT` per abilitare il controllo dei parametri di chiamata della GPU. A causa della complessità dei parametri utilizzati nelle chiamate della GPU, parametri errati possono causare un funzionamento anomalo dell'hardware della GPU, ad esempio dimenticando di aggiungere un simbolo di fine al path o non rispettando i requisiti di allineamento per il buffer stride. Per risolvere rapidamente tali problemi, è stato aggiunto un rigoroso controllo dei parametri prima di ogni chiamata a VG-Lite, che include la convalida del buffer stride e il controllo dell'invertibilità della matrice. Quando viene rilevato un parametro di errore, verrà eseguita un'asserzione per visualizzarlo, consentendo all'utente di apportare tempestivamente correzioni e ridurre il tempo sprecato nella simulazione hardware. Si noti che l'abilitazione di questo controllo ridurrà le prestazioni del runtime. Si consiglia di abilitarlo in modalità Debug e di disabilitarlo nella versione Release.
4. Impostare la configurazione `LV_VG_LITE_FLUSH_MAX_COUNT` per specificare il metodo di "flush". VG-Lite utilizza due set di buffer di comando per il rendering delle istruzioni e un utilizzo efficace di questo meccanismo può migliorare notevolmente l'efficienza di disegno. Attualmente sono supportati due metodi di buffering:
 - Impostare `LV_VG_LITE_FLUSH_MAX_COUNT` a zero (consigliato). Il backend di rendering otterrà lo stato di funzionamento della GPU ogni volta che scrive istruzioni di rendering nel buffer di comando.

Quando la GPU è inattiva, chiamerà immediatamente `vg_lite_flush` per notificare alla GPU di avviare il rendering e scambiare il buffer di comando. Quando la GPU è occupata, continuerà a riempire la cache del buffer di comando con le istruzioni di rendering. Il driver sottostante determinerà automaticamente se il buffer di comando è stato riempito. Quando sta per essere riempito, attenderà forzatamente la fine delle attività di disegno non completate e scambierà il buffer dei comandi. Questo metodo può migliorare efficacemente l'utilizzo della GPU, soprattutto in scenari in cui si esegue il rendering del testo, poiché il tempo di disegno della GPU e il tempo di preparazione dei dati della CPU sono molto simili, consentendo a CPU e GPU di funzionare in parallelo.
 - Impostare `LV_VG_LITE_FLUSH_MAX_COUNT` su un valore maggiore di zero, ad esempio 8. Dopo aver scritto 8 istruzioni di rendering nel buffer dei comandi, il backend di rendering chiamerà `vg_lite_flush` per notificare alla GPU di avviare il rendering e scambiare il buffer dei comandi.

5. Impostare la configurazione `LV_VG_LITE_USE_BOX_SHADOW` per utilizzare il rendering GPU per le ombreggiature. In effetti, l'hardware GPU non supporta effettivamente il rendering delle ombreggiature. Tuttavia, attraverso la sperimentazione, si è scoperto che un effetto ombra simile può essere ottenuto utilizzando più layer di bordi con diversi livelli di trasparenza. Si consiglia di abilitare questa configurazione in scenari in cui i requisiti di qualità delle ombre non sono elevati, in quanto può migliorare significativamente l'efficienza di rendering.
6. Impostare la configurazione `LV_VG_LITE_GRAD_CACHE_CNT` per specificare il numero di voci della cache del gradiente. Il disegno di gradienti include gradienti lineari e radiali. L'utilizzo di una cache può ridurre efficacemente il numero di volte in cui l'immagine del gradiente viene creata e migliorare l'efficienza del disegno. Ogni singolo gradiente consuma circa 4 KB di memoria della GPU. Se nell'interfaccia vengono utilizzati molti gradienti, si può provare ad aumentare il numero di voci nella cache dei gradienti. Se l'API VG-Lite restituisce l'errore `VG_LITE_OUT_OF_RESOURCES`, si può provare ad aumentare la dimensione del pool di memoria della GPU o a ridurre il numero di voci nella cache dei gradienti.
7. Impostare la configurazione `LV_VG_LITE_STROKE_CACHE_CNT` per specificare il numero di cache del path del tratto. Se i parametri del tratto non cambiano, i parametri del tratto generati in precedenza vengono recuperati automaticamente dalla cache per migliorare le prestazioni di rendering. La memoria occupata dal tratto è strettamente correlata alla lunghezza del percorso. Se l'API VG-Lite restituisce l'errore `VG_LITE_OUT_OF_RESOURCES`, si può provare ad aumentare la dimensione del pool di memoria della GPU o a ridurre il numero di voci nella cache dei gradienti.

NOTA: Il backend di rendering VG-Lite non supporta chiamate multi-thread, assicurarsi che `LV_USE_OS` sia sempre configurato come `LV_OS_NONE`.

Simulatore VG-Lite

LVGL integra un simulatore VG-Lite basato su ThorVG. Il suo scopo è semplificare il debug dell'adattamento VG-Lite e ridurre i tempi di debug e individuazione dei problemi sui dispositivi hardware. Per istruzioni dettagliate, vedere *Simulatore GPU VG-Lite*.

API

NXP G2D GPU

API

`lv_draw_g2d.h`

4.7.5 Renesas

Driver Nativi

Renesas è un partner ufficiale di LVGL. Pertanto, LVGL include il supporto nativo per `Dave2D` (la GPU di Renesas) e gestiamo anche progetti Renesas pronti all'uso.

Dave2D

Dave2D è in grado di accelerare la maggior parte delle operazioni di disegno di LVGL:

- Disegno di rettangoli, anche con gradienti
- Disegno, ridimensionamento e rotazione di immagini
- Disegno di lettere
- Disegno di triangoli
- Disegno di linee

Poiché Dave2D lavora in background, la CPU è libera per altre attività. In pratica, durante il rendering, Dave2D può ridurre l'utilizzo della CPU dalla metà a un terzo, a seconda dell'applicazione.

GLCDC

GLCDC è una periferica di uscita grafica multistadio disponibile in diversi MCU Renesas. È in grado di pilotare pannelli LCD tramite un'interfaccia RGB altamente configurabile.

Maggiori informazioni sono disponibili nella [pagina dei driver](#).

Driver MPU

Le MPU Renesas (ovvero la famiglia RZ/G) possono utilizzare questi driver MPU LVGL.

Wayland

L'utilizzo del driver Wayland è supportato sulle schede madri con un desktop Wayland, ovvero un compositore Wayland.

fbdev

Il driver video LVGL fbdev (frame buffer) può essere utilizzato quasi sempre.

RA Family

Schede supportate nella RA Family:

- **EK-RA8D1**
- **EK-RA8P1**
- **EK-RA6M3G**

Esecuzione del Progetto

- L'IDE ufficiale di Renesas si chiama e² studio. Essendo basato su Eclipse, funziona anche su Windows, Linux e Mac. I componenti della famiglia RA richiedono determinate versioni minime del Flexible Software Package (FSP). Per informazioni, consultare i singoli file README.md nei repository applicabili. Gli FSP possono essere scaricati [qui](#). Si noti che a partire dalla versione 6.0 di FSP, LVGL fa ora parte della sezione middleware dei componenti. Pertanto, quando si clonano le ultime versioni dei repository delle board che supportano LVGL, non è più necessario l'aggiornamento ricorsivo dei sottomoduli git.
- JLink viene utilizzato per il debug e può essere scaricato [qui](#).
- Clonare il repository pronto all'uso per la scheda selezionata, per versioni FSP precedenti alla 6.0:

```
git clone https://github.com/lvgl/lv_port_renesas_ek-ra8d1.git --recurse-submodules
```

Scaricare il file .zip da GitHub non funziona perché non scarica i sottomoduli.

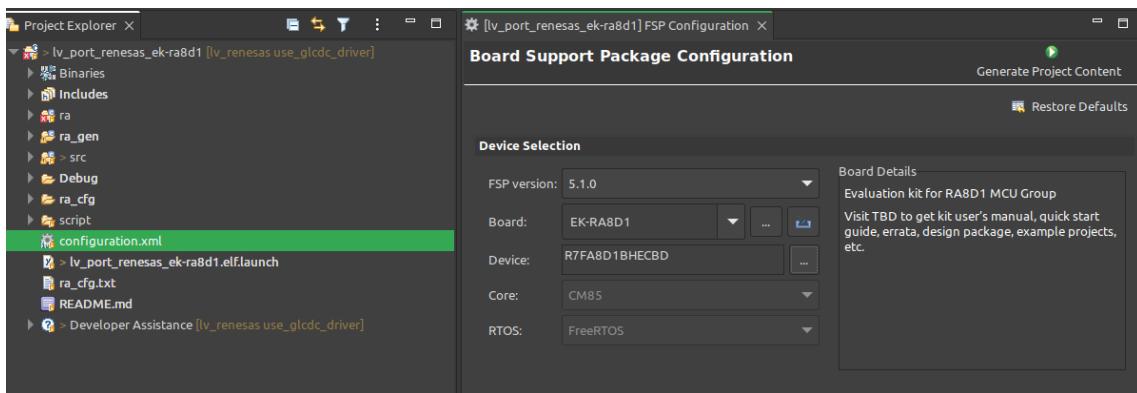
- Clonare il repository pronto all'uso per la scheda selezionata, per versioni FSP dalla 6.0 in poi:

```
git clone https://github.com/lvgl/lv_port_renesas_ek-ra8d1.git
```

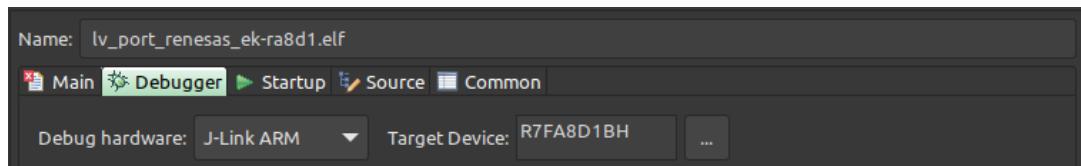
- Aprire e² studio, andare su File -> Import project e selezionare General / Existing projects into workspace.
- Selezionare la cartella clonata e premere Finish.
- Doppio-click su configuration.xml. Questo attiverà la Configuration Window.

Il Flexible Software Package (FSP) di Renesas include il supporto per i livelli BSP e HAL, esteso a diverse varianti di RTOS e altri stack middleware. I componenti saranno disponibili tramite la generazione di codice, incluso il punto di ingresso in "main.c".

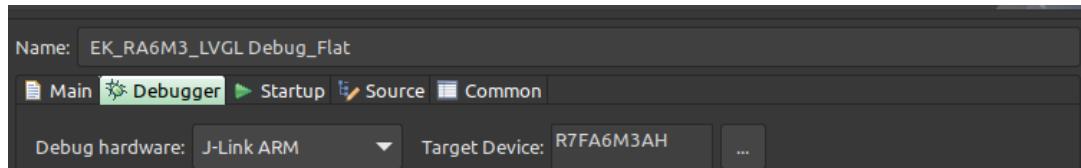
Premere Generate Project Content nell'angolo in alto a destra.



- Compilare il progetto premendo **Ctrl + Alt + B**
- Cliccare sul pulsante Debug (). Se viene richiesto **Debug Configurations**, sul tab **Debugger** selezionare J-Link ARM come **Debug hardware** e l'IC appropriato come **Target Device**:
 - R7FA8D1BH per EK-RA8D1



- R7FA6M3AH per EK-RA6M3G



i Nota

Sulle schede EK-RA8D1, il DIP switch 7 SW1 (al centro della board) deve essere impostato su ON, tutti gli altri su OFF.

Si noti inoltre che l'apertura di un progetto precedentemente creato su FSP prima della versione 6.0 aprirà una finestra di dialogo che chiederà all'utente se desidera migrare al nuovo FSP v6.0. La migrazione non interromperà il progetto.

Modificare il progetto

Aprire una demo

Il punto di ingresso del task principale è contenuto in `src/LVGL_thread_entry.c`.

È possibile disabilitare le demo LVGL (o semplicemente commentarle) e chiamare alcune funzioni `lv_example_.. .()`, oppure aggiungere codice personalizzato.

Configurazione

`src/lv_conf.h` (o `src/lv_conf_user.h`) contiene le impostazioni più importanti per LVGL. Ovvero:

- `LV_COLOR_DEPTH` per impostare la profondità di colore predefinita di LVGL
- `LV_MEM_SIZE` per impostare la RAM massima disponibile per LVGL

- LV_USE_DAVE2D per abilitare la GPU

I componenti hardware e software possono essere modificati visivamente utilizzando la Configuration Window.

A partire da e² Studio 2025-04.1 e FSP 6.0.0, LVGL è un componente gestito.

Supporto

In caso di problemi o domande, aprire una "issue" nel repository corrispondente.

RX Family

Schede supportate nella RX Family:

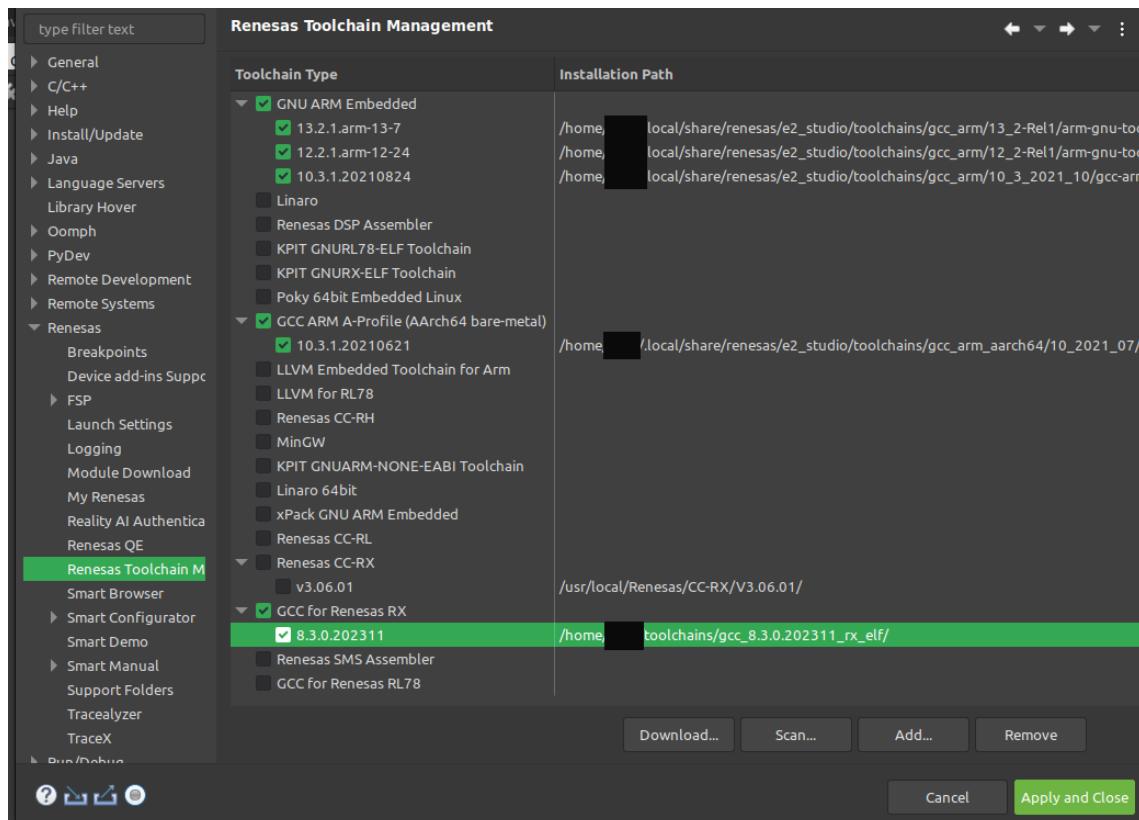
- RX72N Envision Kit

Esecuzione del Progetto

- L'IDE ufficiale di Renesas si chiama e² studio. Essendo basato su Eclipse, funziona anche su Windows, Linux e Mac. Può essere scaricato [qui](#).
- Scaricare e installare il driver necessario per il debugger
 - per Windows: [64 bit qui](#) e [32 bit qui](#)
 - per Linux: [qui](#)
- RX72 richiede un compilatore esterno per il core RXv3. Una versione gratuita e open source è disponibile [qui](#) dopo la registrazione.

Il compilatore deve essere attivato in e² studio:

- Si va su Help -> Add Renesas Toolchains
- Si preme il pulsante Add...
- Si seleziona la cartella di installazione della toolchain



- Clonare il repository già pronto lv_port_renesas_rx72n-envision-kit:

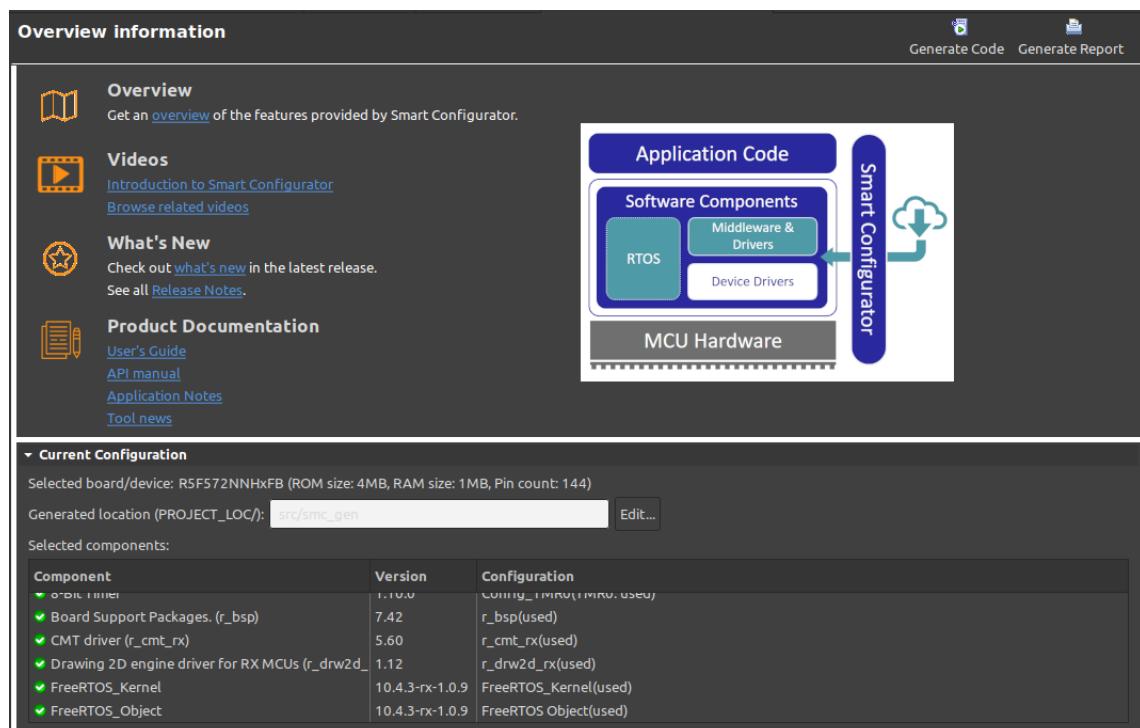
```
git clone https://github.com/lvgl/lv_port_renesas_rx72n-envision-kit.git --recurse-submodules
```

Scaricare il file .zip da GitHub non funziona perché non scarica i sottomoduli.

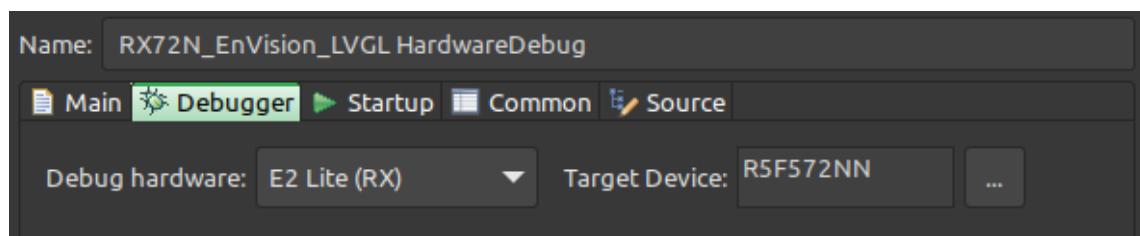
- Aprire e² studio, andare su File -> Import project e selezionare General / Existing projects into workspace
- Selezionare la cartella clonata e premere Finish.
- Doppio-click su RX72N_EnVision_LVGL.scfg per attivare la Configuration Window.

Lo Smart Configurator (SMC) di Renesas include il supporto per i layer BSP e HAL esteso a più varianti RTOS e altri stack middleware. I componenti saranno disponibili tramite la generazione di codice, incluso il punto di ingresso dell'applicazione.

Premere Generate Code nell'angolo in alto a destra.



- Compilare il progetto premendo Ctrl + Alt + B
- Cliccare sul pulsante Debug (). Se viene richiesto Debug Configurations, sul tab Debugger selezionare il E2 Lite come Debug hardware e R5F572NN come Target Device:



Nota

Assicurarsi che entrambi i canali del DIP switch SW1 (accanto a ECN1) siano OFF.

Modificare il progetto

Aprire una demo

Il punto di ingresso del task principale è contenuto in `src/LVGL_thread_entry.c`.

È possibile disabilitare le demo LVGL (o semplicemente commentarle) e chiamare alcune funzioni `lv_example_..()`, oppure aggiungere codice personalizzato.

Configurazione

`src/lv_conf.h` contiene le impostazioni più importanti per LVGL. Ovvero:

- `LV_COLOR_DEPTH` per impostare la profondità di colore predefinita di LVGL
- `LV_MEM_SIZE` per impostare la RAM massima disponibile per LVGL
- `LV_USE_DAVE2D` per abilitare la GPU

I componenti hardware e software possono essere modificati visivamente utilizzando la Configuration Window.

Supporto

In caso di problemi o domande, aprire una "issue" nel repository corrispondente.

RZ/G Family

Schede supportate nella RZ/G Family:

- **RZ/G2L-EVKIT**
- **RZ/G2UL-EVKIT**

Esecuzione del Progetto

- Le schede RZ/G sono MPU con supporto per Linux. I progetti vengono sviluppati utilizzando un SDK con toolchain CLI cross-compilatore in un ambiente PC Linux (WSL, Docker, ecc. possono essere utilizzati su Windows).
- La G2L ha un desktop Wayland e il progetto appare come una finestra Wayland. La G2UL non ha un desktop, quindi il progetto è a schermo intero e utilizza il driver `fbdev`.
- L'SDK attualmente utilizza LVGL v8.3, quindi questo progetto utilizza questa versione per rispecchiare la versione dell'SDK, anche se LVGL è linkato staticamente. È possibile provare a utilizzare versioni più recenti di LVGL. Consultare la [v8-to-v9 Migration Guide](#) per gli aspetti da considerare.
- Clonare il repository pronto all'uso per la board selezionata:

```
git clone https://github.com/lvgl/lv_port_renesas_rz-g2l-evkit --recurse-submodules
```

Scaricare il file `.zip` da GitHub non funziona perché non scarica i sottomoduli.

- Seguire le istruzioni nel file `README.md` del progetto per ottenere l'immagine della scheda SD e il programma di installazione della toolchain, compilare e caricare il progetto sulla scheda.
- Interrompere eventuali demo avviate automaticamente (su G2UL eseguire `systemctl stop demo-launcher` nel terminale).
- Esecuzione del progetto

```
./lvgl_demo_benchmark
```

Modificare il progetto

Aprire una demo

Il punto di ingresso è contenuto in `src/main.c`.

È possibile disabilitare le demo LVGL (`lv_demo_benchmark()`) (o semplicemente commentarle) e chiamare alcune funzioni `lv_example_...()`, oppure aggiungere codice personalizzato.

Configurazione

Modificare `lv_conf.h` e `lv_drv_conf.h` per configurare LVGL. L'immagine della board contiene LVGL e lv_drivers come librerie linkabili dinamicamente. Questo progetto compila LVGL staticamente per la personalizzazione e per il porting del benchmark LVGL v9 da LVGL v8.3.

Supporto

In caso di problemi o domande, aprire una "issue" nel repository corrispondente.

RZ/A Family

Schede supportate nella RZ/A Family:

- **RZ/A3M**

Esecuzione del Progetto

- Le schede RZ/A sono MPU con un focus su applicazioni "bare-metal" e RTOS. I progetti vengono creati utilizzando e² Studio IDE, disponibile per Windows, Mac e Linux.
- Clonare il repository pronto all'uso per la propria board:

```
git clone https://github.com/lvgl/lv_port_renesas-ek-rz_a3m --recurse-submodules
```

Scaricare il file .zip da GitHub non funziona perché non scarica i sottomoduli.

- Seguire le istruzioni nel file README.md del progetto per compilare e flashare il progetto sulla board.

Modificare il progetto

Aprire una demo

Il punto di ingresso è contenuto in `src/LVGL_thread_entry.c`.

È possibile disabilitare le demo LVGL (`lv_demo_benchmark()`) (o semplicemente commentarle) e chiamare alcune funzioni `lv_example_...()`, oppure aggiungere codice personalizzato.

Configurazione

Modificare `lv_conf.h` per configurare LVGL.

Verrà eseguita automaticamente qualsiasi demo abilitata in `lv_conf.h`. È possibile consultare `lv_conf.defaults` per un riepilogo delle configurazioni modificate rispetto a quelle di default.

Supporto

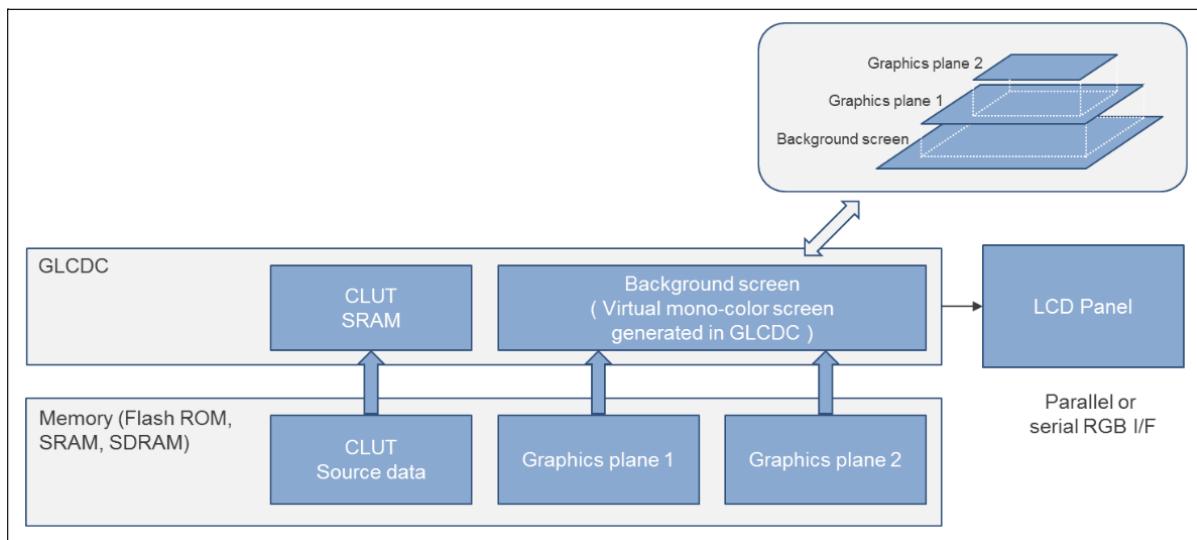
In caso di problemi o domande, aprire una "issue" nel repository corrispondente.

Board Supportate

Nome della Board	CPU	Memoria	Display	Board video	Link
EK-RA8D1					Repository delle demo
	480 MHz Arm Cortex-M85	1MB SDRAM interna 64MB SDRAM esterna 2MB flash interna 64MB octo-SPI flash esterna	4.5" 480x854 2-lane MIPI		
EK-RA8P1					Repository demo
	1 GHz Arm Cortex-M85	1872KB interna 128MB esterna 1MB flash interna	7" 1024x600 Parallel RGB888		
EK-RA6M3G					Repository demo
	120 MHz Arm Cortex-M4	640kB SRAM interna 2MB SRAM interna 32MB QSPI flash esterna	4.3" 480x272 Parallel RGB565		
RX72N Envision Kit					Repository demo
	240 MHz Renesas RXv3	1MB SRAM interna 4MB SRAM interna 32MB QSPI flash esterna	4.3" 480x272 Parallel RGB565		
RZ/G2L-EVKIT					Repository demo
	1.2 GHz Arm Cortex-A55	2GB DDR4 SDRAM	1920x1080 Micro-HDMI		
RZ/G2UL-EVK					Repository demo
	1.0 GHz Arm Cortex-A55	1GB DDR4 SDRAM	1280x800 Parallel RGB888 con Micro-HDMI		
RZ/A3M					Repository demo
	1.0 GHz Arm Cortex-A55	128MB DDR3L-SDRAM interna 128KB SRAM	4.3" 1280x720 MIPI		
4.7. Supporto ai Fornitori di Chip		128MB QSPI NAND flash esterna 32MB QSPI			212

Renesas GLCDC

Panoramica



GLCDC è una periferica di output grafico multistadio utilizzata nei microcontrollori Renesas. È progettata per generare automaticamente segnali di temporizzazione e dati per diversi pannelli LCD.

- Supporta pannelli LCD con interfaccia RGB (fino a 24 bit) e segnali di sincronizzazione (HSYNC, VSYNC e Data Enable opzionali)
- Supporta vari formati colore per i piani grafici in ingresso (RGB888, ARGB8888, RGB565, ARGB1555, ARGB4444, CLUT8, CLUT4, CLUT1)
- Supporta l'utilizzo della Color Look-Up Table (CLUT) per i piani grafici in ingresso (ARGB8888) con 512 word (32 bit/word)
- Supporta vari formati colore per l'output (RGB888, RGB666, RGB565, Serial RGB888)
- Può inserire due piani grafici sopra il piano di background e fonderli sullo schermo
- Genera un dot clock per il pannello. La sorgente di clock è selezionabile tra interna o esterna (LCD_EXTCLK).
- Supporta la regolazione della luminosità, del contrasto e la correzione gamma.
- Supporta gli interrupt GLCDC per gestire la commutazione del frame buffer o il rilevamento dell'underflow.

La configurazione di un progetto e l'ulteriore integrazione con l'ecosistema Renesas sono descritte in dettaglio nella [pagina Renesas](#). Consultare i seguenti repository per esempi pronti all'uso:

- [EK-RA8D1](#)
- [EK-RA6M3G](#)
- [RX72N Envision Kit](#)

Prerequisiti

- Questo driver si basa sul codice generato da e² studio. Omettere questo passaggio durante la configurazione del progetto causerà un errore di compilazione.
- Attivare il driver impostando `LV_USE_RENESAS_GLCD` a 1 nel file `lv_conf.h`.

Utilizzo

Non è necessario implementare funzioni specifiche per la piattaforma.

Il codice seguente illustra l'utilizzo del driver in modalità `LV_DISPLAY_RENDER_MODE_DIRECT`.

```
lv_display_t * disp = lv_renesas_glcdf_create();
lv_display_set_default(disp);
```

Per utilizzare il driver in modalità `LV_DISPLAY_RENDER_MODE_PARTIAL`, è necessario allocare un buffer aggiuntivo, preferibilmente nella regione di memoria più veloce disponibile.

Lo swapping del buffer può essere attivato passando un secondo buffer della stessa dimensione al posto dell'argomento NULL.

```
static lv_color_t partial_draw_buf[DISPLAY_HSIZE_INPUT0 * DISPLAY_VSIZE_INPUT0 / 10] BSP_PLACE_IN_SECTION(".sram") BSP_ALIGN_VARIABLE(1024);
lv_display_t * disp = lv_renesas_glcdf_partial_create(partial_draw_buf, NULL, sizeof(partial_draw_buf));
lv_display_set_default(disp);
```

Nota

La modalità parziale può essere attivata tramite la macro nel file `src/board_init.c` dei progetti demo.

Rotazione dello schermo

La rotazione dello schermo basata su software è supportata in modalità parziale. Utilizza l'API comune, non è richiesta alcuna configurazione aggiuntiva:

```
lv_display_set_rotation(lv_display_get_default(), LV_DISP_ROTATION_90);
/* OR */
lv_display_set_rotation(lv_display_get_default(), LV_DISP_ROTATION_180);
/* OR */
lv_display_set_rotation(lv_display_get_default(), LV_DISP_ROTATION_270);
```

Assicurarsi che l'heap sia sufficientemente grande, poiché verrà allocato un buffer con le stesse dimensioni del buffer parziale.

GPU Dave2D

4.7.6 STM32

Panoramica

Informazioni

STMicroelectronics è un fornitore di chip e produttore di schede per MCU e MPU ARM a 32 bit.

Sviluppo di Applicazioni

STM32CubeIDE è un IDE basato su Eclipse, generalmente utilizzato per lo sviluppo per STM32. Può configurare il progetto, compilarlo, flasharlo e debuggarlo. Integra LVGL in modo simile a qualsiasi altro IDE basato su Eclipse. I progetti che genera sono basati su STM32 HAL. STM32 HAL è gestito da ST ed è adatto a molte applicazioni. STM32CubeIDE può facoltativamente generare progetti che utilizzano FreeRTOS.

STM32CubeIDE può essere scaricato [da qui](#).

I progetti STM32 possono essere creati con STM32CubeIDE o con altri framework che non utilizzano STM32 HAL. Esistono molti modi per sviluppare per STM32, tra cui bare metal manuale, Zephyr RTOS, NuttX RTOS, ChibiOS e molti altri. Framework come Zephyr e NuttX spesso fanno sì che un progetto venga trattato come "un progetto Zephyr su STM32" piuttosto che come un "progetto STM32 che esegue Zephyr", quindi la loro documentazione è il punto di partenza. Vale la pena menzionare anche STM32CubeMX, uno strumento grafico per generare il codice boilerplate per un progetto STM32 HAL, da compilare al di fuori di STM32CubeIDE. Inoltre, STM32CubeCLT è un set di tool per integrare quelli proprietari di ST in altri IDE come Visual Studio Code.

Progetti Pronti-All'uso

LVGL gestisce alcuni progetti per schede STM32. Consultare [la pagina delle board](#) per le schede ST certificate da LVGL e con integrazione LVGL aggiornata, e consultare [tutti i repository](#) per altri repo di schede ST esistenti. Seguire i file README.md in quei repository per istruzioni specifiche su come iniziare a utilizzarli.

Supporto LVGL per Periferiche Grafiche STM32

LVGL offre un buon supporto per le periferiche di accelerazione hardware grafica di cui sono dotati alcuni modelli STM32. A volte queste funzionalità possono essere utilizzate contemporaneamente. Consultare le singole pagine a riguardo.

- *LTDC: controller del display.*
- *DMA2D: operazioni di memoria dati pixel asincrone.*
- *NeoChrom: GPU per il rendering di primitive grafiche 2D.*

Ciascuna pagina indica se/come possono essere utilizzate insieme.

Utilizzare LVGL in un progetto STM32 HAL

Creare il Progetto STM32 HAL

Se si dispone di una scheda STM32 con un display montato, è consigliabile inizializzare il progetto con STM32CubeIDE o STM32CubeMX individuando la scheda nel selettore delle schede. A volte le schede di terze parti dispongono di un progetto STM32Cube di esempio. Il componente comune di questi progetti è il file .ioc. Descrive tutte le configurazioni delle periferiche e le mappature dei pin. È molto più facile iniziare a utilizzare una board con un display se il display LCD è già configurato nel file .ioc. Se il display LCD della scheda è controllato dalla periferica LTDC (LCD-TFT Display Controller), è possibile utilizzare il *driver LVGL LTDC*.

Aggiungere LVGL al Progetto

- `git clone` o copiare LVGL nella directory del progetto.
- Nel riquadro **Project Explorer** di STM32CubeIDE : click destro sulla cartella LVGL e selezionare **Add/remove include path....**. Se questa opzione non viene visualizzata o non funziona, si possono controllare i percorsi di inclusione del progetto nel menu **Project -> Properties**, quindi andare in **C/C++ Build -> Settings -> Include paths** e assicurarsi che la directory LVGL sia elencata.

Ora che i file sorgente sono inclusi nel progetto, seguire le istruzioni per *aggiungere LVGL al progetto*, per creare il file `lv_conf.h` e inizializzare il display. Prima di inizializzare manualmente il display, verificare se il progetto utilizza la periferica LTDC (LCD-TFT Display Controller). In tal caso, si può semplicemente *usare il driver LTDC di LVGL*.

Utilizzo di LVGL con STM32 HAL

Dopo aver esaminato *Aggiungere LVGL al progetto*, si apprezzeranno le specifiche sull'utilizzo di LVGL con un progetto STM32 HAL illustrate qui.

tick

Si dovrà impostare la callback tick su `HAL_GetTick`.

```
lv_tick_set_cb(HAL_GetTick);
```

Nel ciclo del gestore del timer, si deve ritardare ritardare con `HAL_Delay`. Un buon valore per il ritardo è 2 ms. A causa del funzionamento di `HAL_Delay`, un ritardo di 1 in realtà ritarda solo per un intervallo compreso tra 0 e 1 ms.

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    lv_timer_handler();
    HAL_Delay(2);
    /* USER CODE END 3 */
}
```

Driver del Display

Prima di inizializzare manualmente il display, verificare se il progetto utilizza la periferica LTDC (LCD-TFT Display Controller). In tal caso, si può semplicemente *usare il driver LTDC di LVGL*.

Se il display è *uno per cui LVGL ha un driver*, lo si dovrebbe usare in modo da avere meno logica del driver da implementare nel progetto.

Consulta questa guida per creare un driver per display SPI.

RTOS

Abilitando FreeRTOS o CMSIS RTOS nel progetto HAL STM32, è possibile abilitare il supporto LVGL per il corrispondente RTOS e LVGL utilizzerà le primitive di creazione e sincronizzazione dei thread del rispettivo RTOS durante il rendering, a seconda dei renderer abilitati.

Impostare `LV_USE_OS` a `LV_OS_FREERTOS` o `LV_OS_CMSIS_RTOS2`.

Driver del display LTDC STM32

Alcuni STM32 dispongono di una periferica specializzata chiamata LTDC (LCD-TFT Display Controller) per il controllo dei display.

Modalità di utilizzo con LVGL

Il driver all'interno di LVGL è progettato per funzionare con una periferica LTDC già configurata. Si basa sull'HAL STM32 per rilevare informazioni sulla configurazione. Il formato colore del display LVGL creato corrisponderà al formato colore del layer LTDC. Utilizzare STM32CubeIDE o STM32CubeMX per generare il codice di inizializzazione LTDC. Si tratta di un driver basato su STM32 HAL e sulla possibilità di includere "main.h" dall'interno di LVGL per accedere alle definizioni STM32 HAL e ai prototipi di funzione.

Esistono diversi casi d'uso per il driver LVGL. Tutte le permutazioni delle opzioni seguenti sono ben supportate.

- buffer singolo o doppio
- modalità di rendering diretta o parziale
- OS o non OS
- flushing parallelo con DMA2D (solo per la modalità di rendering parziale)

Se il sistema operativo è abilitato, verrà utilizzata una primitiva di sincronizzazione per dare al thread la possibilità di cedere il passo ad altri thread mentre è bloccato, migliorando l'utilizzo della CPU. Vedere `LV_USE_OS` nel proprio `lv_conf.h`

Layer LTDC

Questo driver crea un display LVGL che si occupa solo di uno specifico layer della periferica LTDC, il che significa che è possibile creare due display LVGL LTDC che operano indipendentemente sui rispettivi layer.

Direct Render Mode

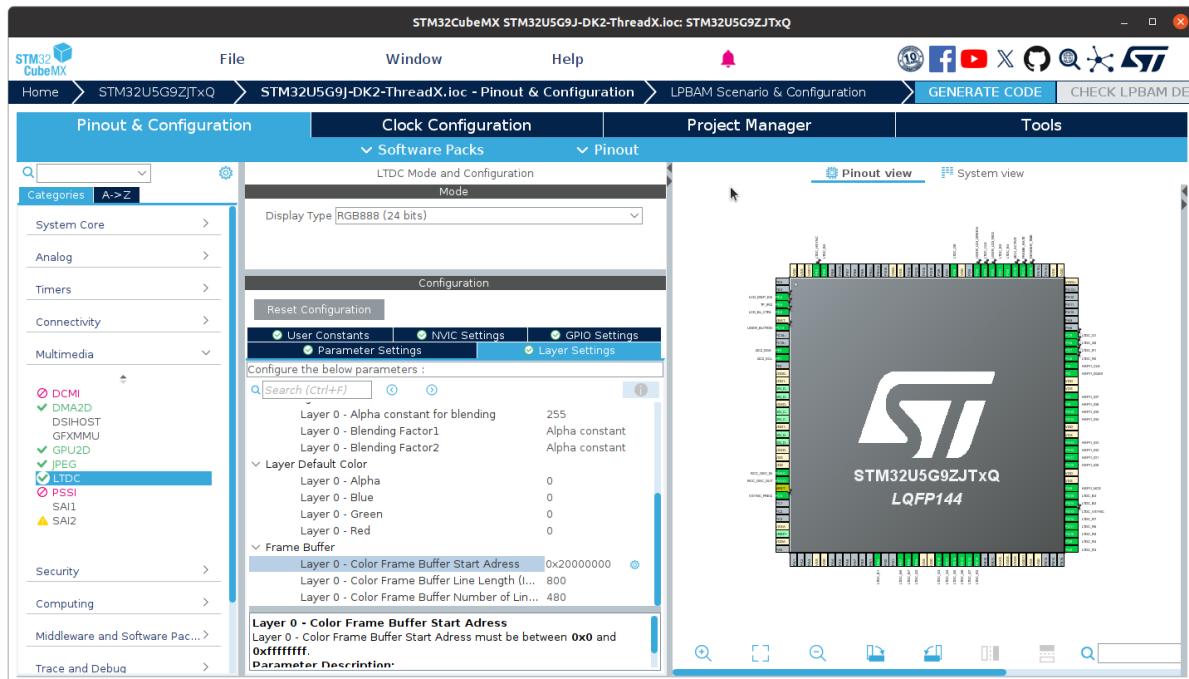
Per la modalità di rendering diretto, richiamare `lv_st_ltdc_create_direct()` in questo modo:

```
void * my_ltdc_framebuffer_address = (void *)0x20000000;
uint32_t my_ltdc_layer_index = 0; /* typically 0 or 1 */
lv_display_t * disp = lv_st_ltdc_create_direct(my_ltdc_framebuffer_address,
                                              optional_other_full_size_buffer,
                                              my_ltdc_layer_index);
```

`my_ltdc_framebuffer_address` è il framebuffer configurato per l'utilizzo da parte di LTDC. `optional_other_full_size_buffer` può essere un altro buffer delle stesse dimensioni del framebuffer di default per la modalità a doppio buffer, oppure `NULL` in caso contrario. `my_ltdc_layer_index` è l'indice del layer LTDC per cui creare il display.

Per ottenere i migliori risultati visivi, si consiglia di utilizzare `optional_other_full_size_buffer` se la memoria disponibile è sufficiente. La modalità a buffer singolo è quella da utilizzare se la memoria è molto scarsa. I chip dotati di cache dati della CPU presentano inevitabili artefatti visivi quando si utilizza la modalità diretta "single-buffered". Se la memoria disponibile è quasi sufficiente per la modalità diretta a doppio buffer, ma non del tutto, utilizzare la modalità di rendering parziale.

Per chiarire cosa sia esattamente `my_ltdc_framebuffer_address`, si tratta del valore di `pLayerCfg.FBStartAdress` quando il layer LTDC viene configurato utilizzando l'HAL STM32, che viene scritto nel registro CFBAR della periferica del layer LTDC.



Modalità di Rendering Parziale

Per la modalità di rendering parziale, invocare `lv_st_ltdc_create_partial()` in questo modo:

```
static uint8_t partial_buf[65536];
static uint8_t optional_partial_buf[65536];
uint32_t my_ltdc_layer_index = 0; /* typically 0 or 1 */
lv_display_t * disp = lv_st_ltdc_create_partial(partial_buf,
                                                optional_partial_buf,
                                                65536,
                                                my_ltdc_layer_index);
```

Il driver utilizzerà le informazioni nella configurazione del layer LTDC per trovare il framebuffer del layer e scaricarlo.

Fornire un secondo buffer parziale può migliorare l'utilizzo della CPU e aumentare le prestazioni rispetto a un singolo buffer se `LV_ST_LTDC_USE_DMA2D_FLUSH` è abilitato.

Script del Linker

È necessario assicurarsi che la memoria del framebuffer LTDC sia effettivamente riservata nello script del linker. Questo è un file che normalmente ha l'estensione `.ld`. Nell'esempio seguente, 1125K è specificato perché la profondità di colore è 24 (3 byte per pixel), la larghezza del display è 800, l'altezza del display è 480 e 1K significa 1024 byte. $3 \times 800 \times 480 \div 1024 = 1125$. La somma delle voci RAM (`FB_RAM + RAM`) dev'essere uguale alla RAM totale del dispositivo.

```
/* Memories definition */
MEMORY
{
    FB_RAM (xrw) : ORIGIN = 0x20000000, LENGTH = 1125K /* single 24bit 800x480 buffer */
    RAM (xrw)   : ORIGIN = 0x20119400, LENGTH = 1883K
    FLASH (rx)  : ORIGIN = 0x08000000, LENGTH = 4096K
}
```

Se il framebuffer non è riservato nello script del linker o non è impostato durante l'inizializzazione della board, è assolutamente possibile impostarlo in fase di esecuzione prima di creare il display LVGL.

```
/* like this */
HAL_LTDC_SetAddress(&hltdc, (uint32_t)my_ltdc_framebuffer_address, my_ltdc_layer_index);

/* and then... */

lv_display_t * disp;
disp = lv_st_ltdc_create_direct(my_ltdc_framebuffer_address,
                                optional_other_full_size_buffer,
                                my_ltdc_layer_index);

/* or */
disp = lv_st_ltdc_create_partial(partial_buf1,
                                 optional_partial_buf2,
                                 partial_buf_size,
                                 my_ltdc_layer_index);
```

Rotazione del Display

Il driver supporta la rotazione del display con `lv_display_set_rotation(disp, rotation)` dove `rotation` è una tra `LV_DISP_ROTATION_90`, `LV_DISP_ROTATION_180` o `LV_DISP_ROTATION_270`. La rotazione è inizialmente `LV_DISP_ROTATION_0`.

La rotazione viene eseguita tramite software e funziona solo se il display è stato creato utilizzando `lv_st_ltdc_create_partial()`. `LV_ST_LTDC_USE_DMA2D_FLUSH` non avrà alcun effetto se si utilizza la rotazione.

Interoperabilità con le unità di disegno DMA2D e NeoChrom

`LV_ST_LTDC_USE_DMA2D_FLUSH` può essere abilitato per utilizzare DMA2D per svuotare i buffer parziali in parallelo con altre attività LVGL, indipendentemente dal fatto che il sistema operativo sia abilitato o meno. Se la visualizzazione non è parziale, non è necessario abilitare questa opzione.

Non deve essere abilitato contemporaneamente a `LV_USE_DRAW_DMA2D`. Vedere [DMA2D support](#).

NeoChrom può essere abilitato contemporaneamente a LTDC. Non interferiranno in alcun modo tra loro.

NeoChrom

LVGL integra NeoChrom come [unità draw](#).

Progetti pronti all'uso con NeoChrom abilitato

`lv_port_riverdi_stm32u5` è un porting pronto-all'usoper il Display Embedded Riverdi STM32 da 5,0" (STM32U599NJH6Q o STM32U5A9NJH6Q) con NeoChrom abilitato. Seguire le istruzioni nel file `readme` per iniziare.

`lv_port_stm32u5g9j-dk2` è un "porting" pronto all'uso per il devkit STM32U5G9J-DK2 con display da 5,0" e GPU NeoChrom VG. Seguire le istruzioni nel file `readme` per iniziare.

Utilizzo e Configurazione

Abilitare il renderer impostando `LV_USE_NEMA_GFX` a 1 in `lv_conf.h`. Se si utilizza `LV_USE_NEMA_VG`, impostare `LV_NEMA_GFX_MAX_RESX` e `LV_NEMA_GFX_MAX_RESY` in base alle dimensioni del display che si utilizzerà, in modo che venga riservata sufficiente memoria statica per VG. Senza VG, il renderer software eseguirà più tipi di attività.

"libs/nema_gfx" contiene i binari precompilati per i driver GPU NeoChrom.

`lv_port_riverdi_stm32u5` è già configurato per linkare i binari "cortex_m33_revC" durante la compilazione e `lv_port_stm32u5g9j-dk2` configurato per linkare i binari "cortex_m33_NemaPVG" durante la compilazione.

"cortex_m33_revC" funziona su tutti i dispositivi STM32 m33 con un core NeoChrom, mentre "cortex_m33_NemaPVG" supporta inoltre l'accelerazione della tassellazione e della moltiplicazione di matrici di cui sono capaci i core STM32 U5F e U5G NeoChrom.

Con un diverso progetto STM32CubeIDE, è possibile configurare le librerie da linkare cliccando col pulsante destro del mouse sul progetto nella barra laterale "Project Explorer", selezionando "Properties", andando su "C/C++ Build", "Settings", "MCU G++ Linker" e quindi su "Libraries". Aggiungere una voce in "Libraries (-l)" che sia "nemagfx-float-abi-hard". Aggiungere una voce in "Library search path (-L)"

che sia un path per "libs/nema_gfx/lib/core/cortex_m33_revC/gcc" ad esempio "\${workspace_loc}/\${ProjName}/Middlewares/LVGL/lvgl/libs/nema_gfx/lib/core/cortex_m33_revC/gcc}". Aggiungere anche la directory "libs/nema_gfx/include" ai percorsi di ricerca degli include. In "MCU GCC Compiler", "Include paths", aggiungere una voce a "Include paths (-I)" che corrisponde al percorso di "libs/nema_gfx/include" ad esempio "\${workspace_loc}/\${ProjName}/Middlewares/LVGL/lvgl/libs/nema_gfx/include}". Cliccare su "Apply and Close".

1 Nota

```
cannot find -lnemagfx-float-abi-hard: No such file or directory
```

La versione della libreria per ST è stata aggiornata a revC quindi il componente del path cortex_m33 deve essere aggiornato a cortex_m33_revC in STM32CubeIDE.

È supportato *LV_COLOR_DEPTH* a 32 e 16 bit.

Al momento della stesura di questo articolo, il supporto per *LV_USE_OS* è sperimentale e non ancora funzionante in *lv_port_riverdi_stm32u5*

NeoChrom richiede una semplice implementazione HAL per allocare memoria e, facoltativamente, bloccare le risorse. È possibile utilizzare un'implementazione HAL personalizzata per la propria piattaforma oppure utilizzare una delle implementazioni fornite impostando *LV_USE_NEMA_HAL_CUSTOM*.

Grafica Vettoriale

Il driver NeoChrom VG in LVGL può eseguire il rendering di Widget di grafica vettoriale utilizzando il supporto hardware di NeoChrom VG per il "vector graphics drawing". È possibile visualizzare file SVG nella propria applicazione. Vedere gli esempi SVG per l'utilizzo.

Per utilizzare la grafica vettoriale con NeoChrom, è necessario abilitare le seguenti configurazioni in *lv_conf.h*.

```
LV_USE_NEMA_GFX 1
LV_USE_NEMA_VG 1
LV_USE_VECTOR_GRAPHIC 1
LV_USE_MATRIX 1
LV_USE_FLOAT 1
```

Per utilizzare il widget SVG, abilitare anche *LV_USE_SVG*.

Se è disponibile RAM, le prestazioni di SVG possono essere aumentate abilitando la cache delle immagini, *LV_CACHE_DEF_SIZE*. *LV_CACHE_DEF_SIZE* è la dimensione della cache in byte. Se è sufficientemente grande per i file SVG, memorizzerà nella cache i dati SVG decodificati in modo che non debbano essere analizzati ad ogni aggiornamento, riducendo significativamente i tempi di ridisegno SVG.

LV_USE_DEMO_VECTOR_GRAPHIC è una demo abilitabile e che disegna alcune forme grafiche vettoriali. I riempimenti a gradienti e con immagini non sono ancora supportati, così come le linee tratteggiate. Questi elementi non sono presenti nella demo quando viene eseguita con il driver NeoChrom.

Immagini TSC

Il convertitore TSC può essere scaricato qui.

Questo motore di rendering può disegnare immagini TSC (ThinkSilicon Compression). I formati colore TSC 4/6/6A/12/12A fanno parte di *lv_color_format_t*. Tutti gli altri motori di rendering ignoreranno le immagini con questi formati colore. Definire una variabile descrittore di immagine con il formato colore TSC corrispondente e la GPU sarà in grado di disegnarla direttamente. Non è necessario specificare Stride perché verrà calcolato dal motore di rendering.

```
const lv_image_dsc_t img_demo_widgets_avatar_tsc6a = {
    .header.cf = LV_COLOR_FORMAT_NEMA_TSC6A,
    .header.w = 144,
    .header.h = 144,
    .data = img_demo_widgets_avatar_tsc6a_map,
    .data_size = sizeof(img_demo_widgets_avatar_tsc6a_map),
};
```

Interoperabilità con il driver LTDC e la "Draw Unit" DMA2D

NeoChrom può essere abilitato contemporaneamente a LTDC. Non interferiranno in alcun modo tra loro.

NeoChrom e DMA2D possono essere abilitati contemporaneamente. Sono entrambe unità di disegno e accetteranno attività di disegno in modo indipendente.

API

`lv_draw_nema_gfx.h`

STM32 DMA2D

STM32 DMA2D è anche chiamato Chrom-ART. Qui verrà chiamato DMA2D.

Cos'è DMA2D?

Molti STM32 sono dotati di una periferica hardware dedicata per accelerare la manipolazione dei pixel. Funziona in modo asincrono con la CPU. È simile a un motore DMA da memoria a memoria e può anche essere utilizzato come tale, ma consente di specificare lo "stride" di origine e destinazione e di convertire i dati che sposta da un formato pixel a un altro. Può eseguire il "blend" dei primi piani con i background controllati da un valore alfa costante o da un canale pixel alfa.

LVGL integra DMA2D come *unità di disegno*. Durante il rendering, l'unità di disegno DMA2D di LVGL eseguirà i task che può eseguire. Quelli che non può eseguire verranno eseguiti in parallelo dall'unità di disegno software. I task di disegno che la "draw unit" DMA2D di LVGL può eseguire sono riempimenti e immagini. In generale, l'unità di disegno DMA2D può gestire riempimenti e immagini che utilizzano formati di colore standard, senza gradiente, senza trasformazioni e senza raggio.

Utilizzo

Per utilizzare DMA2D nel progetto, si imposta `LV_USE_DRAW_DMA2D` a 1 in `lv_conf.h`. Si dovrà specificare l'header che LVGL deve includere internamente per le definizioni DMA2D. Impostare `LV_DRAW_DMA2D_HAL_INCLUDE` all'header corrispondente. Ad esempio, se il modello STM32 è un STM32U5, il nome dell'header sarà probabilmente "`stm32u5xx_hal.h`". Se si sta usando un framework, assicurarsi che non sia in conflitto con LVGL sulla periferica DMA2D. LVGL utilizza l'accesso diretto ai registri di memoria per manipolare la periferica DMA2D. Non utilizza l'HAL STM32.

Non è richiesto altro per far funzionare DMA2D immediatamente, ma potrebbe essere opportuno fare alcune considerazioni.

Poiché DMA2D è implementato in LVGL come unità di disegno, può funzionare in parallelo con altre unità di disegno (in genere solo l'unità di disegno software). Per default, ogni volta che DMA2D sta lavorando a un task di disegno dal cui completamento LVGL dipende prima di continuare, LVGL si metterà in attesa del completamento di DMA2D. Per far andare la CPU in sospensione o pianificarla per altre attività RTOS mentre è in corso un trasferimento DMA2D, procedere come segue:

1. Deve essere presente un RTOS/OS. Impostare `LV_USE_OS` su uno dei valori supportati in `lv_conf_template.h` corrispondente all'RTOS in uso.
2. Impostare `LV_USE_DRAW_DMA2D_INTERRUPT` a 1 in `lv_conf.h`.
3. È necessario chiamare `lv_draw_dma2d_transfer_complete_interrupt_handler()` quando si riceve l'interrupt globale che segnala il completamento di un trasferimento DMA2D.

Interoperabilità con LTDC e NeoChrom

L'utilizzo di DMA2D può essere liberamente combinato con l'utilizzo di LTDC purché `LV_ST_LTDC_USE_DMA2D_FLUSH` non sia abilitato. LTDC utilizzerà la periferica DMA2D per il flushing, se abilitata.

NeoChrom e DMA2D possono essere abilitati contemporaneamente. Sono entrambe unità di disegno e accetteranno attività di disegno in modo indipendente.

API

lv_draw_dma2d.h

lv_draw_dma2d_private.h

Creazione di Driver per Display SPI per STM32

Driver del Display

Ecco come pilotare un display SPI RGB565 240x320 utilizzando STM32 HAL SPI. È possibile utilizzare la modalità diretta o parziale. In questo esempio è mostrata la modalità diretta a buffer singolo.

Nel codice di inizializzazione, si crea il display e si impostano i buffer e la funzione di callback del flush.

```
static uint8_t fb[240 * 320 * 2];
lv_display_t * disp = lv_display_create(240, 320);
lv_display_set_buffers(disp, fb, NULL, sizeof(fb), LV_DISPLAY_RENDER_MODE_DIRECT);
lv_display_set_flush_cb(disp, flush_cb);
lv_display_set_color_format(disp, LV_COLOR_FORMAT_RGB565); /* not needed if LV_COLOR_DEPTH is 16 */
```

Il codice HAL STM32 generato dovrebbe inizializzare la periferica SPI in base ai valori di configurazione in STM32CubeIDE o STM32CubeMX.

Se sono presenti comandi da inviare per inizializzare il display, inviarli prima di entrare nel ciclo del timer.

Ecco un esempio di implementazione della funzione di callback del flush .

```
extern SPI_HandleTypeDef hspil;

static void flush_cb(lv_display_t * disp, const lv_area_t * area, uint8_t * px_map)
{
    if(lv_display_flush_is_last(disp)) {
        /* Chip select (CS) pins are typically active-low. */
        HAL_GPIO_WritePin(LCD_CS_GPIO_Port, LCD_CS_Pin, GPIO_PIN_RESET);
        HAL_SPI_Transmit(&hspil, px_map, lv_area_get_size(area) * 2, HAL_MAX_DELAY);
        HAL_GPIO_WritePin(LCD_CS_GPIO_Port, LCD_CS_Pin, GPIO_PIN_SET);
    }
    lv_display_flush_ready(disp);
}
```

Le prestazioni possono essere migliorate utilizzando DMA con doppio buffering.

Touch indev

Nel codice di inizializzazione, crea l'indev [input device] del touchscreen.

```
lv_indev_t * indev = lv_indev_create();
lv_indev_set_type(indev, LV_INDEV_TYPE_POINTER);
lv_indev_set_read_cb(indev, touch_read);
```

Il codice HAL STM32 generato dovrebbe inizializzare la periferica I2C in base ai valori di configurazione in STM32CubeIDE o STM32CubeMX.

Ecco un esempio di implementazione di un driver touch che utilizza la comunicazione I2C e un pin di interrupt per segnalare i dati disponibili.

```
extern I2C_HandleTypeDef hi2c2;

static volatile bool do_sample_touch = false;
static lv_indev_state_t last_state = LV_INDEV_STATE_RELEASED;

/* the STM32 HAL calls this function for you */
void HAL_GPIO_EXTI_Falling_Callback(uint16_t GPIO_Pin)
{
    if (GPIO_Pin == TP IRQ_Pin)
    {
        do_sample_touch = true;
    }
}

static void touch_read(lv_indev_t * indev, lv_indev_data_t * data)
{
    NVIC_DisableIRQ(EXTI5_IRQn);
    if (do_sample_touch)
    {
        uint8_t touches = 0;
        uint8_t buf[6];
        const uint16_t STATUS_REG = 0x814E;
        const uint16_t TOUCH_POS_REG = 0x8150;
        uint8_t ZERO = 0;

        HAL_I2C_Mem_Read(&hi2c2, 0xBA, STATUS_REG, 2, buf, 1, HAL_MAX_DELAY);
        touches = (buf[0] & buf[1]);
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

    HAL_I2C_Mem_Write(&hi2c2, 0xBA, STATUS_REG, 2, &ZERO, 1, HAL_MAX_DELAY);

    do_sample_touch = false;

    if (touches > 0)
    {
        last_state = LV_INDEV_STATE_PRESSED;

        HAL_I2C_Mem_Read(&hi2c2, 0xBA, TOUCH_POS_REG, 2, buf, 4, HAL_MAX_DELAY);
        data->point.x = buf[0] + (buf[1] << 8);
        data->point.y = buf[2] + (buf[3] << 8);
    }
    else {
        last_state = LV_INDEV_STATE_RELEASED;
    }
}

NVIC_EnableIRQ(EXTI5 IRQn);

data->state = last_state;
}

```

4.8 Supporto del Controller del Display

4.8.1 EVE

EVE (FT81x)

Un driver framebuffer minimale per i controller smart display EVE FT81x. Funziona anche per BT81x.

Al momento è supportata la modalità di rendering parziale a buffer singolo. Questa potrebbe non supportare DSPI o QSPI.

Testato a 32 MHz su FT812 e a 23 MHz su BT817.

Utilizzo

```

#define FB_SIZE          800 * 2 * 50
#define MAX_TRANSFER_SIZE  FB_SIZE

static void spi_cb(lv_display_t * disp, lv_ft81x_spi_operation operation, void * data, uint32_t length)
{
    spi_device_handle_t spi = lv_ft81x_get_user_data(disp);
    switch(operation) {
        case LV_FT81X_SPI_OPERATION_CS_ASSERT:
            gpio_set_level(CS_PIN, 0);
            break;
        case LV_FT81X_SPI_OPERATION_CS_DEASSERT:
            gpio_set_level(CS_PIN, 1);
            esp_rom_delay_us(10); /* tiny delay in case a CS_ASSERT immediately follows */
            break;
        case LV_FT81X_SPI_OPERATION_SEND:
            spi_transaction_t trans = {0};
            while(length) {
                uint32_t sz = length < MAX_TRANSFER_SIZE ? length : MAX_TRANSFER_SIZE;
                trans.length = sz * 8;
                trans.rxlengt = 0;
                trans.tx_buffer = data;
                spi_device_polling_transmit(spi, &trans);
                length -= sz;
                data += sz;
            }
            break;
        case LV_FT81X_SPI_OPERATION_RECEIVE:
            spi_transaction_t trans = {0};
            trans.length = length * 8;
            trans.rxlengt = length * 8;
            trans.tx_buffer = data;
            spi_device_polling_transmit(spi, &trans);
            break;
    }
}

int main() {
    /* ... */

    // reset the ft81x
    gpio_set_level(PD_PIN, 0);
    vTaskDelay(6 / portTICK_PERIOD_MS);
    gpio_set_level(PD_PIN, 1);
    vTaskDelay(21 / portTICK_PERIOD_MS);

    /* ... */

    // taken from https://github.com/lvgl/lvgl_esp32_drivers/blob/9fed1cc47b5a45fec6bae08b55d2147d3b50260c/lvgl_tft/EVE_config.h
    // NHD-5.0-800x480FT-Cxx-xxx 800x480 5.0" Newhaven, resistive or capacitive, FT81x
    // EVE_NHD_50
    #define EVE_VSYNCO      (0L)
    #define EVE_VSYNC1      (3L)
    #define EVE_VOFFSET     (32L)
    #define EVE_VCYCLE      (525L)
    #define EVE_HSYNCO      (0L)
    #define EVE_HSYNC1      (48L)
    #define EVE_HOFFSET     (88L)
}

```

(continues on next page)

(continua dalla pagina precedente)

```

#define EVE_HCYCLE      (928L)
#define EVE_PCLKPOL    (0L)
#define EVE_SWIZZLE     (0L)
#define EVE_PCLK        (2L)
#define EVE_CSREAD      (1L)

lv_ft81x_parameters_t params = {
    .hor_res = 800,
    .ver_res = 480,

    .hcycle = EVE_HCYCLE,
    .hoffset = EVE_HOFFSET,
    .hsync0 = EVE_HSYNC0,
    .hsync1 = EVE_HSYNC1,
    .vcycle = EVE_VCYCLE,
    .voffset = EVE_VOFFSET,
    .vsync0 = EVE_VSYNC0,
    .vsync1 = EVE_VSYNC1,
    .swizzle = EVE_SWIZZLE,
    .pclkpol = EVE_PCLKPOL,
    .csread = EVE_CSREAD,
    .pclk = EVE_PCLK,

    .has_crystal = true,
    .is_bt81x = false
};

static uint8_t fb[FB_SIZE] __attribute__((aligned(4)));
lv_display_t * disp = lv_ft81x_create(&params, fb, FB_SIZE, spi_cb, spi);
/* ... */
}

```

Risoluzione dei problemi

Se il "backlight" [retroilluminazione] non si accende (o è troppo luminoso o troppo fioco), provare a modificare il valore di `PWM_DUTY_BACKLIGHT_ON` in `lv_ft81x.c`, che può variare a seconda della board.

Renderer GPU esterno EVE

EVE è un tipo di circuito integrato GPU esterno che accetta comandi di disegno di alto livello tramite SPI e invia la grafica renderizzata a un display tramite RGB parallelo.

Il vantaggio di utilizzare un chip EVE è che la responsabilità del rendering viene rimossa dalla MCU di controllo, che può quindi essere un componente con specifiche inferiori o dedicare più tempo di elaborazione ad altre attività. L'interfaccia SPI è più semplice da collegare rispetto all'RGB parallelo, soprattutto se il chip EVE è integrato in un display.

LVGL include un renderer per EVE. Le interfacce utente LVGL possono essere renderizzate da EVE e nella maggior parte dei casi sono praticamente indistinguibili dal renderer software.

Vedere anche il driver framebuffer [EVE \(FT81x\)](#). Controlla gli stessi chip EVE, ma è un'implementazione più semplice e autonoma che utilizza il rendering software e invia tutti i pixel tramite SPI, quindi è molto più lenta.

Limitazioni

- Formato, dimensione e limite di conteggio delle immagini.
- Formato, dimensione e limite di conteggio del font.
- Per ora sono supportati solo i font a 4 bpp.
- Il numero totale di attività renderizzate per aggiornamento ha un limite massimo.
- I layer non sono supportati.

Utilizzo

Parametri della Board

Trovare i parametri di visualizzazione e popolare una struct `lv_draw_eve_parameters_t` con essi.

Ecco un esempio di `lv_draw_eve_target_parameters.h` per Riverdi RVT50HQBNWC00-B che ha un BT817Q --- ovvero il chip EVE. Questi parametri sono stati presi dalla raccolta di configurazioni fornita dalla libreria EVE che supporta questo renderer. [Vedere qui](#). Controllare lì i parametri della propria scheda.

```
lv_draw_eve_parameters_t params = {
    .hor_res = 800,
    .ver_res = 480,
    .vsync0 = 0,
    .vsync1 = 4,
    .voffset = 8,
    .vcycle = 496,
    .hsync0 = 0,
    .hsync1 = 4,
    .hoffset = 8,
    .hcycle = 816,
    .pcclk = 3,
    .pcclkpol = 1,
    .swizzle = 0,
    .cspread = 0,
    .has_crystal = true,
    .has_gt911 = false,
    .backlight_freq = 4000,
    .backlight_pwm = 128,
};

};
```

Implementazione IO del chip EVE

L'utente deve implementare le funzionalità GPIO e SPI IO. Il renderer EVE LVGL chiamerà questa callback per eseguire la comunicazione SPI con il chip EVE.

```
static void op_cb(lv_display_t * disp, lv_draw_eve_operation_t operation,
                  void * data, uint32_t length)
{
    /* optional: get the `user_data` parameter you passed to `lv_draw_eve_display_create` */
    void * your_user_data = lv_draw_eve_display_get_user_data(disp);

    switch(operation) {
        case LV_DRAW_EVE_OPERATION_POWERDOWN_SET:
            /* Setting the pin low powers down the EVE chip */
            your_gpio_write(PD_N_PIN, 0);
            break;
        case LV_DRAW_EVE_OPERATION_POWERDOWN_CLEAR:
            /* setting the pin high powers on the EVE chip */
            your_gpio_write(PD_N_PIN, 1);
            break;
        case LV_DRAW_EVE_OPERATION_CS_ASSERT:
            /* Setting the pin low asserts the EVE chip SPI device */
            your_gpio_write(CS_N_PIN, 0);
            break;
        case LV_DRAW_EVE_OPERATION_CS_DEASSERT:
            /* setting the pin high de-asserts the EVE chip SPI device */
            your_gpio_write(CS_N_PIN, 1);
            break;
        case LV_DRAW_EVE_OPERATION_SPI_SEND:
            /* `data` is the data to send */
            your_spi_transmit(data, length);
            break;
        case LV_DRAW_EVE_OPERATION_SPI_RECEIVE:
            /* `data` is the destination for the data */
            your_spi_receive(data, length);
            break;
    }
}
```

Si dovrà anche inizializzare la periferica SPI e i pin GPIO.

22 MHz è stata la velocità SPI più elevata che ha funzionato durante i test con la scheda Riverdi e l'ESP32-S3. Si potrebbero non ottenere risultati con questa velocità, quindi si consiglia di convalidare con un valore SPI_SPEED di 10 (10 MHz) e di aumentarlo sperimentalmente durante i test.

Creazione di un display LVGL per EVE

Per creare il display LVGL per il renderer EVE, si chiama `lv_draw_eve_display_create(params, op_cb, your_user_data)` che restituisce il display creato. `your_user_data` può essere NULL. Dovrebbe essere chiamato dopo l'inizializzazione di GPIO e SPI. Si può scegliere di inizializzare l'IO la prima volta che viene chiamato `op_cb`.

Non sono necessari buffer per il renderer LVGL EVE perché nessun pixel viene scritto nei buffer del dispositivo che esegue LVGL. Quando è necessario disegnare qualcosa, una serie di comandi viene inviata a EVE.

Creazione di un Touch Indev

`lv_draw_eve_touch_create(disp)` crea un touch *Dispositivi di input (lv_indev)* per il display.

Potrebbe essere necessario configurare l'indirizzo I2C del controller touch connesso a EVE. Vedere la sezione [Accesso al Registro EVE](#) per maggiori informazioni sull'accesso ai registri.

Ecco un esempio di impostazione del registro REG_TOUCH_CONFIG su un chip EVE BT817q per un touch screen capacitivo con un controller con indirizzo I2C 0x15.

```

/*
15: 0: capacitive, 1: resistive      CAPACITIVE
14: host mode                      NO
13: reserved
12: ignore short circuit protection NO
11: low-power mode                 NO
10-4: 7-bit i2c address            0x15
3: reserved
2: suppress 300ms startup         NO
1-0: 2-bit sampling clocks val    use 1 (the reset default)
*/
lv_draw_eve_memwritel6(disp, LV_EVE_REG_TOUCH_CONFIG, 0x0151);

```

Rotazione del Display

La rotazione efficiente del display è pienamente supportata tramite `lv_display_set_rotation()`. La rotazione dell'input touch viene gestita di conseguenza.

Pre-Caricamento delle Risorse

Immagini e font vengono caricati nella RAM_G del chip EVE in base alle esigenze durante il rendering. Un'immagine o un glifo non verrà caricato finché non sarà necessario renderizzarlo. Una volta caricato, rimarrà nella RAM_G fino al reset. La prima volta che viene visualizzata una schermata, potrebbe verificarsi un ritardo notevole durante il caricamento di un'immagine o di un set di glifi per il primo utilizzo. Per mitigare questo ritardo, è disponibile una serie di funzioni per caricare le risorse in anticipo.

`lv_draw_eve_pre_upload_image()` viene utilizzato per caricare le immagini. `lv_draw_eve_pre_upload_font_range()` viene utilizzato per caricare un intervallo di caratteri. L'intervallo è fornito come primo e ultimo punto di codice Unicode nell'intervallo. Sia il valore iniziale che quello finale sono inclusi nell'intervallo. Può essere chiamato più volte con intervalli diversi. `lv_draw_eve_pre_upload_font_text()` viene utilizzato per caricare tutti i glifi necessari per il rendering di una stringa specifica (ASCII o UTF-8). Può essere chiamato più volte con stringhe diverse.

```

LV_IMAGE_DECLARE(asset_digital_lock);
lv_draw_eve_pre_upload_image(disp, &asset_digital_lock);

LV_FONT_DECLARE(lv_font_montserrat_48);
lv_draw_eve_pre_upload_font_text(disp, &lv_font_montserrat_48, "The current time is");
lv_draw_eve_pre_upload_font_range(disp, &lv_font_montserrat_48, '0', '9');
lv_draw_eve_pre_upload_font_range(disp, &lv_font_montserrat_48, ':', ';');

```

Formati di Risorse Supportati

I formati di colore immagine supportati sono RGB565, RGB565A8, ARGB8888 e L8. È importante notare che RGB565A8 e ARGB8888 vengono convertiti in ARGB4444 prima di essere caricati su EVE. Ciò implica che sul display verrà visualizzata una profondità di colore ridotta rispetto alla risorsa originale. Anche il tempo di caricamento iniziale della risorsa potrebbe essere più lungo.

L'unico formato di font supportato è 4 bpp. Se lo "stride" del font non è 1, verrà convertito in "stride 1" prima di essere caricato su EVE. Se il font ha già uno stride pari a 1, verrà caricato direttamente senza conversione, il che può ridurre i tempi di caricamento della risorsa.

Per generare un font con uno stride specifico (ad esempio 1), è necessario utilizzare [offline font converter](#) e specificare un argomento `stride` sulla riga di comando, ad esempio `--stride 1`. Il valore di default è "stride 0". Ciò significa che i bit vengono compressi uniformemente su più righe, ma EVE non può utilizzare font compressi su più righe.

Accesso al Registro EVE

Le funzioni `lv_draw_eve_memread8()`, `lv_draw_eve_memread16()`, `lv_draw_eve_memread32()`, `lv_draw_eve_memwrite8()`, `lv_draw_eve_memwrite16()` e `lv_draw_eve_memwrite32()` sono disponibili se necessario. Sono wrapper per `EVE_memRead8()`, ecc.

Le definizioni dei registri e altre enumerazioni EVE sono disponibili includendo `lvgl.h` sotto il namespace con prefisso `LV_EVE_`. Ad esempio, `REG_ID` è disponibile come `LV_EVE_REG_ID` e `EVE_ROM_CHIPID` è disponibile come `LV_EVE_EVE_ROM_CHIPID`, ecc.

Ulteriori Informazioni

- https://brtchip.com/wp-content/uploads/Support/Documentation/Programming_Guides/ICs/EVE/FT81X_Series_Programmer_Guide.pdf
- https://brtchip.com/wp-content/uploads/2024/06/BRT_AN_033_BT81X-Series-Programming-Guide.pdf

4.8.2 Driver generico per controller LCD compatibile con MIPI DCS

Panoramica

Da Wikipedia:

MIPI Alliance è un'alleanza commerciale globale che sviluppa specifiche tecniche per l'ecosistema mobile, in particolare per gli smartphone, ma include anche i settori influenzati dalla tecnologia mobile. MIPI è stata fondata nel 2003 da Arm, Intel, Nokia, Samsung, STMicroelectronics e Texas Instruments.

MIPI Alliance ha pubblicato una serie di specifiche relative ai dispositivi di visualizzazione, tra cui DBI (Display Bus Interface), DSI (Display Serial Interface) e DCS (Display Command Set). Di solito, quando si parla di un display compatibile con MIPI, si pensa a un dispositivo con interfaccia seriale DSI. Tuttavia, la specifica Display Bus Interface include una serie di altre interfacce legacy, come l'interfaccia seriale SPI o l'interfaccia parallela compatibile con i8080, che vengono spesso utilizzate per interfacciare i display LCD con microcontrollori di fascia bassa. Inoltre, la specifica DCS contiene un set di comandi standard, supportato da un gran numero di controller LCD TFT legacy, inclusi i popolari SOC Sitronix (ST7735, ST7789, ST7796) e Ilitek (ILI9341). Questi comandi forniscono un'interfaccia comune per configurare l'orientamento del display, la risoluzione del colore, varie modalità di alimentazione e forniscono l'accesso generico alla memoria video. Oltre a questo set di comandi standard, ogni chip controller LCD dispone di una serie di comandi specifici del fornitore per configurare i livelli del generatore di tensione, le temporizzazioni o le curve gamma.

Nota

È importante comprendere che questo driver LCD MIPI generico non è un driver hardware per display con interfaccia seriale DSI ("MIPI"). Invece, implementa il set di comandi MIPI DCS utilizzato in molti controller LCD con bus SPI o i8080 e fornisce un framework comune per i controller display specifici per chip.

Suggerimento

Sebbene si tratti di un driver generico, può essere utilizzato per supportare chip compatibili che non dispongono di un driver specifico.

Prerequisiti

Non ci sono prerequisiti.

Configurazione del driver

Abilitare il supporto del driver LCD MIPI generico in lv_conf.h, tramite la definizione del compilatore cmake o tramite KConfig.

```
#define LV_USE_GENERIC_MIPI 1
```

Nota

`LV_USE_GENERIC_MIPI` viene abilitato automaticamente quando viene abilitato un driver compatibile.

Utilizzo

È necessario implementare due funzioni dipendenti dalla piattaforma:

```
/* Send short command to the LCD. This function shall wait until the
 * transaction finishes. */
void my_lcd_send_cmd(lv_display_t * disp, const uint8_t * cmd, size_t cmd_size,
                      const uint8_t * param, size_t param_size)
{
    ...
}

/* Send large array of pixel data to the LCD controller. If necessary, this
 * function has to do the byte-swapping. This function can do the transfer in
 * the background. If so, the user must guarantee that it cannot be called while
 * another background transfer is still in progress. */
void my_lcd_send_color(lv_display_t * disp, const uint8_t * cmd, size_t cmd_size,
                       uint8_t * param, size_t param_size)
{
    ...
}
```

L'unica differenza tra le funzioni `my_lcd_send_cmd()` e `my_lcd_send_color()` è che `my_lcd_send_cmd()` viene utilizzata per inviare comandi brevi e ci si aspetta che completi la transazione quando ritorna (in altre parole, dovrebbe essere bloccante), mentre `my_lcd_send_color()` viene utilizzata solo per inviare dati di pixel e si consiglia di utilizzare DMA per trasmettere i dati in background. È possibile implementare anche metodi più sofisticati, come l'accodamento dei trasferimenti e la loro pianificazione in background.

Si noti che, sebbene il flushing del display sia gestito dal driver, è responsabilità dell'utente chiamare `lv_display_flush_ready()` al termine del trasferimento del colore. In caso di trasferimento DMA, questa chiamata viene solitamente effettuata in una routine di servizio di interrupt (ISR) che si attiva al termine del trasferimento.

Nota

Sebbene sia accettabile utilizzare un'implementazione bloccante anche per il trasferimento dei pixel, le prestazioni ne risentiranno.

Suggerimento

È necessario prestare attenzione a non inviare un comando mentre è in corso un trasferimento attivo in background. È responsabilità dell'utente implementare questa funzionalità interrogando l'hardware, interrogando una variabile globale (che viene reimpostata al termine del trasferimento) o utilizzando un semaforo o un altro meccanismo di blocco.

Si noti inoltre che il driver non gestisce l'allocazione del "draw buffer" [buffer di disegno], poiché anche questa potrebbe dipendere dalla piattaforma. Pertanto è necessario allocare i buffer e assegnarli all'oggetto di visualizzazione come di consueto chiamando `lv_display_set_buffers()`.

Il driver può essere utilizzato per creare più display. In una configurazione di questo tipo, le callback devono essere in grado di distinguere tra i display. In genere si implementa un set separato di callback per ciascun display. Si noti inoltre che l'utente deve occuparsi dell'arbitraggio del bus quando più dispositivi sono collegati ad esso.

Esempio

```
#include "src/drivers/display/st7789/lv_st7789.h"

#define LCD_H_RES      240
#define LCD_V_RES      320
#define LCD_BUF_LINES   60

lv_display_t * my_disp;
...

/* Initialize LCD I/O bus, reset LCD */
static int32_t my_lcd_io_init(void)
{
    ...
    return HAL_OK;
}

/* Send command to the LCD controller */
static void my_lcd_send_cmd(lv_display_t * disp, const uint8_t * cmd, size_t cmd_size,
                           const uint8_t * param, size_t param_size)
{
    MY_PIN_PREP_TO_SEND_CMD();
```

(continues on next page)

(continua dalla pagina precedente)

```

MY_PIN_LCD_CONTROLLER_CHIP_SELECT();
/* Any delay here needed to meet LCD controller requirements. */

/* Send command to LCD controller here. */

MY_PIN_PREP_TO_SEND_DATA();
/* Any delay here needed to meet LCD controller requirements. */

/* Send data to LCD controller here. */

MY_PIN_LCD_CONTROLLER_CHIP_DESELECT();
/* Any delay here needed to meet LCD controller requirements. */
}

/* Send pixel data to the LCD controller */
static void my_lcd_send_color(lv_display_t * disp, const uint8_t * cmd, size_t cmd_size,
                             uint8_t * param, size_t param_size)
{
    MY_PIN_PREP_TO_SEND_CMD();
    MY_PIN_LCD_CONTROLLER_CHIP_SELECT();
    /* Any delay here needed to meet LCD controller requirements. */

    /* Send command to LCD controller here. */

    MY_PIN_PREP_TO_SEND_DATA();
    /* Any delay here needed to meet LCD controller requirements. */

    /* Send data to LCD controller here in a manner efficient for large blocks of data. */

    /* If data transfer is done with DMA, these 2 steps are not made here, but
     * are instead made immediately after the DMA transfer completes. */
    MY_PIN_LCD_CONTROLLER_CHIP_DESELECT();
    lv_display_flush_ready(disp);
}

int main(int argc, char ** argv)
{
    ...

    /* Initialize LVGL */
    lv_init();

    /* Initialize LCD bus I/O */
    if (my_lcd_io_init() != 0)
        return;

    /* Create the LVGL display object and the LCD display driver */
    my_disp = lv_lcd_generic_mipi_create(LCD_H_RES, LCD_V_RES, LV_LCD_FLAG_NONE,
                                         my_lcd_send_cmd, my_lcd_send_color);

    /* Set display orientation to landscape */
    lv_display_set_rotation(my_disp, LV_DISPLAY_ROTATION_90);

    /* Configure draw buffers, etc. */
    uint8_t * buf1 = NULL;
    uint8_t * buf2 = NULL;

    uint32_t buf_size = LCD_H_RES * LCD_BUF_LINES
        * lv_color_format_get_size(lv_display_get_color_format(my_disp));

    buf1 = lv_malloc(buf_size);
    if(buf1 == NULL) {
        LV_LOG_ERROR("display draw buffer malloc failed");
        return;
    }

    /* Allocate secondary buffer if needed */
    ...

    lv_display_set_buffers(my_disp, buf1, buf2, buf_size, LV_DISPLAY_RENDER_MODE_PARTIAL);

    ui_init(my_disp);

    while(true) {
        ...

        /* Periodically call the lv_timer handler */
        lv_timer_handler();
    }
}

```

Argomenti avanzati

Creazione di flag

Il terzo argomento della funzione `lv_lcd_generic_mipi_create()` è un array di flag. Questo può essere utilizzato per configurare l'orientamento e l'ordinamento RGB del pannello se le impostazioni di default non funzionano. In particolare, il driver MIPI generico accetta i seguenti flag:

LV_LCD_FLAG_NONE
LV_LCD_FLAG_MIRROR_X
LV_LCD_FLAG_MIRROR_Y
LV_LCD_FLAG_BGR

È possibile passare più flag combinandoli tramite OR:

```

lv_display_t * disp;
disp = lv_lcd_generic_mipi_create(MY_DISPLAY_PIXEL_WIDTH,
                                 MY_DISPLAY_PIXEL_HEIGHT,
                                 LV_LCD_FLAG_MIRROR_X | LV_LCD_FLAG_BGR,

```

(continues on next page)

(continua dalla pagina precedente)

```
my_lcd_send_cmd,
my_lcd_send_color);
```

Elenchi di comandi personalizzati

Sebbene i driver specifici per chip facciano del loro meglio per inizializzare correttamente il controller LCD, è possibile che diversi pannelli TFT richiedano configurazioni diverse. In particolare, una corretta impostazione della gamma è fondamentale per una buona riproduzione dei colori. Purtroppo, trovare un buon set di parametri non è facile. Di solito il produttore del pannello fornisce un codice di esempio con le impostazioni di registro consigliate.

È possibile utilizzare la funzione `my_lcd_send_cmd()` per inviare un comando arbitrario al controller LCD. Tuttavia, per semplificare l'invio di un numero elevato di parametri, il driver MIPI generico supporta l'invio di un elenco di comandi personalizzati al controller. I comandi devono essere inseriti in un array di `'uint8_t'`:

```
static const uint8_t init_cmd_list[] = {
    <command 1>, <number of parameters>, <parameter 1>, ... <parameter N>,
    <command 2>, <number of parameters>, <parameter 1>, ... <parameter N>,
    ...
    LV_LCD_CMD_DELAY_MS, LV_LCD_CMD_EOF      /* Terminate list: this is required! */
};

...
lv_ld_generic_mipi_send_cmd_list(my_disp, init_cmd_list);
```

È possibile aggiungere un ritardo tra i comandi utilizzando lo pseudo-comando `LV_LCD_CMD_DELAY_MS`, seguito dal ritardo specificato in unità di 10 ms. Per terminare l'elenco dei comandi, è necessario utilizzare un ritardo con valore `LV_LCD_CMD_EOF`, come mostrato sopra.

Vedere un esempio concreto di invio di un elenco di comandi [qui](#).

4.8.3 Driver per controller LCD ILI9341

Panoramica

ILI9341 è un driver SOC a chip singolo da 262.144 colori per display a cristalli liquidi a-TFT con risoluzione di 240RGBx320 punti, composto da un driver sorgente a 720 canali, un driver di gate a 320 canali, 172.800 byte di GRAM per dati di visualizzazione grafica di 240RGBx320 punti e un circuito di alimentazione. ILI9341 supporta l'interfaccia MCU parallela con bus dati a 8/9/16/18 bit, l'interfaccia RGB con bus dati a 6/16/18 bit e l'interfaccia periferica seriale (SPI) a 3/4 linee.

Il `driver` del controller LCD ILI9341 è un driver indipendente dalla piattaforma, basato sul `driver generico MIPI`. Implementa l'inizializzazione del display, supporta la rotazione del display e implementa la callback di flush del display. L'utente deve implementare solo due funzioni specifiche della piattaforma per inviare un comando o dati pixel al controller tramite SPI o bus parallelo. In genere, queste vengono implementate richiamando le funzioni della libreria SDK appropriate sulla piattaforma specificata.

Prerequisiti

Non ci sono prerequisiti.

Configurazione del driver

Abilitare il supporto del driver ILI9341 in `lv_conf.h`, tramite la definizione del compilatore `cmake` o tramite `KConfig`.

```
#define LV_USE_ILI9341 1
```

Utilizzo

È necessario implementare due funzioni dipendenti dalla piattaforma:

```
/* Send short command to the LCD. This function shall wait until the transaction finishes. */
void my_lcd_send_cmd(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, const uint8_t *param, size_t param_size)
{
    ...
}

/* Send large array of pixel data to the LCD. If necessary, this function has to do the byte-swapping. This function can do the transfer in the background. */

```

(continues on next page)

(continua dalla pagina precedente)

```
void my_lcd_send_color(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, uint8_t *param, size_t param_size)
{
    ...
}
```

Per creare un display basato su ILI9341, utilizzare la funzione

```
/** 
 * Create an LCD display with ILI9341 driver
 * @param hor_res      horizontal resolution
 * @param ver_res      vertical resolution
 * @param flags        default configuration settings (mirror, RGB ordering, etc.)
 * @param send_cmd     platform-dependent function to send a command to the LCD controller (usually uses polling transfer)
 * @param send_color   platform-dependent function to send pixel data to the LCD controller (usually uses DMA transfer: must implement a 'ready' callback)
 * @return             pointer to the created display
 */
lv_display_t * lv_ilil9341_create(uint32_t hor_res, uint32_t ver_res, lv_lcd_flag_t flags,
                                    lv_ilil9341_send_cmd_cb_t send_cmd_cb, lv_ilil9341_send_color_cb_t send_color_cb);
```

Per ulteriori dettagli e un esempio funzionante, consultare la [documentazione del driver generico MIPI](#).

4.8.4 Driver del controller LCD ST7735

Panoramica

ST7735S è un controller/driver a chip singolo per LCD TFT-LCD grafico a 262K colori. È costituito da 396 circuiti di pilotaggio per linee "source" e 162 per linee di gate. Questo chip è in grado di connettersi direttamente a un microprocessore esterno e accetta interfacce SPI (Serial Peripheral Interface) e interfacce parallele a 8/9/16/18 bit. I dati del display possono essere memorizzati nella RAM integrata del display da 132 x 162 x 18 bit. Può eseguire operazioni di lettura/scrittura dei dati di visualizzazione nella RAM senza bisogno di un clock esterno, riducendo al minimo il consumo energetico. Inoltre, grazie ai circuiti di alimentazione integrati necessari per pilotare i cristalli liquidi, è possibile realizzare un sistema di visualizzazione con un numero inferiore di componenti.

Il [driver](#) del controller LCD ST7735 è un driver indipendente dalla piattaforma, basato sul [driver generico MIPI](#). Implementa l'inizializzazione del display, supporta la rotazione del display e implementa la callback di flush del display. L'utente deve implementare solo due funzioni specifiche della piattaforma per inviare un comando o dati pixel al controller tramite SPI o bus parallelo. In genere, queste vengono implementate richiamando le funzioni della libreria SDK appropriate sulla piattaforma specificata.

Prerequisiti

Non ci sono prerequisiti.

Configurazione del driver

Abilitare il supporto del driver ST7735 in `lv_conf.h`, tramite la definizione del compilatore `cmake` o tramite KConfig

```
#define LV_USE_ST7735 1
```

Utilizzo

È necessario implementare due funzioni dipendenti dalla piattaforma:

```
/* Send short command to the LCD. This function shall wait until the transaction finishes. */
void my_lcd_send_cmd(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, const uint8_t *param, size_t param_size)
{
    ...
}

/* Send large array of pixel data to the LCD. If necessary, this function has to do the byte-swapping. This function can do the transfer in the background. */
void my_lcd_send_color(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, uint8_t *param, size_t param_size)
{
    ...
}
```

Per creare un display basato su ST7735 utilizzare la funzione

```
/** 
 * Create an LCD display with ST7735 driver
 * @param hor_res      horizontal resolution
 * @param ver_res      vertical resolution
 * @param flags        default configuration settings (mirror, RGB ordering, etc.)
 * @param send_cmd     platform-dependent function to send a command to the LCD controller (usually uses polling transfer)
 * @param send_color   platform-dependent function to send pixel data to the LCD controller (usually uses DMA transfer: must implement a 'ready' callback)
 * @return             pointer to the created display
 */
lv_display_t * lv_st7735_create(uint32_t hor_res, uint32_t ver_res, lv_lcd_flag_t flags,
                                lv_st7735_send_cmd_cb_t send_cmd_cb, lv_st7735_send_color_cb_t send_color_cb);
```

(continues on next page)

(continua dalla pagina precedente)

```
/*
lv_display_t * lv_st7735_create(uint32_t hor_res, uint32_t ver_res, lv_lcd_flag_t flags,
                                lv_st7735_send_cmd_cb_t send_cmd_cb, lv_st7735_send_color_cb_t send_color_cb);
```

Per ulteriori dettagli e un esempio funzionante, consultare la [documentazione del driver generico MIPI](#).

4.8.5 Driver per Controller LCD ST7789

Panoramica

L'ST7789 è un controller/driver a chip singolo per TFT-LCD grafico a 262K colori. È costituito da 720 circuiti di pilotaggio per linee "source" e 320 per linee di gate. Questo chip è in grado di connettersi direttamente a un microprocessore esterno e accetta interfacce parallele a 8/9/16/18 bit. I dati del display possono essere memorizzati nella RAM integrata da 240x320x18 bit. Può eseguire operazioni di lettura/scrittura dei dati di visualizzazione nella RAM senza bisogno di un clock esterno, riducendo al minimo il consumo energetico. Inoltre, grazie al circuito di alimentazione integrato necessario per pilotare i cristalli liquidi, è possibile realizzare un sistema di visualizzazione con il minor numero di componenti.

Il [driver](#) del controller LCD ST7789 è un driver indipendente dalla piattaforma, basato sul [driver generico MIPI](#). Implementa l'inizializzazione del display, supporta la rotazione del display e implementa la callback di flush del display. L'utente deve implementare solo due funzioni specifiche della piattaforma per inviare un comando o dati pixel al controller tramite SPI o bus parallelo. In genere, queste vengono implementate richiamando le funzioni della libreria SDK appropriate sulla piattaforma specificata.

Prerequisiti

Non ci sono prerequisiti.

Configurazione del driver

Abilitare il supporto del driver ST7789 in lv_conf.h, tramite la definizione del compilatore cmake o tramite KConfig

```
#define LV_USE_ST7789 1
```

Utilizzo

È necessario implementare due funzioni dipendenti dalla piattaforma:

```
/* Send short command to the LCD. This function shall wait until the transaction finishes. */
void my_lcd_send_cmd(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, const uint8_t *param, size_t param_size)
{
    ...
}

/* Send large array of pixel data to the LCD. If necessary, this function has to do the byte-swapping. This function can do the transfer in the background. */
void my_lcd_send_color(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, uint8_t *param, size_t param_size)
{
    ...
}
```

Per creare un display basato su ST7789 utilizzare la funzione

```
/**
 * Create an LCD display with ST7789 driver
 * @param hor_res      horizontal resolution
 * @param ver_res      vertical resolution
 * @param flags        default configuration settings (mirror, RGB ordering, etc.)
 * @param send_cmd     platform-dependent function to send a command to the LCD controller (usually uses polling transfer)
 * @param send_color   platform-dependent function to send pixel data to the LCD controller (usually uses DMA transfer: must implement a 'ready' callback)
 * @return             pointer to the created display
 */
lv_display_t * lv_st7789_create(uint32_t hor_res, uint32_t ver_res, lv_lcd_flag_t flags,
                                lv_st7789_send_cmd_cb_t send_cmd_cb, lv_st7789_send_color_cb_t send_color_cb);
```

Per ulteriori dettagli e un esempio funzionante, consultare la [documentazione del driver generico MIPI](#).

4.8.6 Driver per Controller LCD ST7796

Panoramica

ST7796S è un controller/driver a chip singolo per display LCD TFT grafico a 262K colori. È costituito da 960 linee "source" e 480 linee di gate per il pilotaggio dei circuiti. ST7796S è in grado di connettersi direttamente a

un microprocessore esterno e accetta interfacce parallele a 8/9/16/18 bit, SPI; inoltre, fornisce anche un'interfaccia MIPI. I dati del display possono essere memorizzati nella RAM integrata da 320x480x18 bit. Può eseguire operazioni di lettura/scrittura della RAM dei dati del display senza clock esterno, riducendo al minimo il consumo energetico. Inoltre, grazie al circuito di alimentazione integrato necessario per pilotare i cristalli liquidi, è possibile realizzare un sistema di visualizzazione con un numero inferiore di componenti.

Il [driver](#) del controller LCD ST7796 è un driver indipendente dalla piattaforma, basato sul [driver generico MIPI](#). Implementa l'inizializzazione del display, supporta la rotazione del display e implementa la callback di flush del display. L'utente deve implementare solo due funzioni specifiche della piattaforma per inviare un comando o dati pixel al controller tramite SPI o bus parallelo. In genere, queste vengono implementate richiamando le funzioni della libreria SDK appropriate sulla piattaforma specificata.

Prerequisiti

Non ci sono prerequisiti.

Configurazione del driver

Abilitare il supporto del driver ST7796 in lv_conf.h, tramite la definizione del compilatore cmake o tramite KConfig

```
#define LV_USE_ST7796 1
```

Utilizzo

È necessario implementare due funzioni dipendenti dalla piattaforma:

```
/* Send short command to the LCD. This function shall wait until the transaction finishes. */
void my_lcd_send_cmd(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, const uint8_t *param, size_t param_size)
{
    ...
}

/* Send large array of pixel data to the LCD. If necessary, this function has to do the byte-swapping. This function can do the transfer in the background. */
void my_lcd_send_color(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, uint8_t *param, size_t param_size)
{
    ...
}
```

Per creare un display basato su ST7796 utilizzare la funzione

```
/**
 * Create an LCD display with ST7796 driver
 * @param hor_res      horizontal resolution
 * @param ver_res      vertical resolution
 * @param flags        default configuration settings (mirror, RGB ordering, etc.)
 * @param send_cmd     platform-dependent function to send a command to the LCD controller (usually uses polling transfer)
 * @param send_color   platform-dependent function to send pixel data to the LCD controller (usually uses DMA transfer: must implement a 'ready' callback)
 * @return             pointer to the created display
 */
lv_display_t * lv_st7796_create(uint32_t hor_res, uint32_t ver_res, lv_lcd_flag_t flags,
                                lv_st7796_send_cmd_cb_t send_cmd_cb, lv_st7796_send_color_cb_t send_color_cb);
```

Per ulteriori dettagli e un esempio funzionante, consultare la [documentazione del driver generico MIPI](#).

4.8.7 Driver per il Controller LCD NV3007

Panoramica

NV3007 è un driver a chip singolo per display a cristalli liquidi TFT a-Si a 262.144 colori con una risoluzione massima di 168RGBx428 punti. Contiene un sorgente di driver a 252 canali, un driver GIP a 24 canali utilizzato per il controllo a doppio gate, 161.784 byte di GRAM per i dati di visualizzazione grafica, un circuito di alimentazione interno di precisione che supporta la modalità di visualizzazione a colori, a 8 colori e in modalità sleep. NV3007 supporta l'interfaccia periferica seriale (SPI) a 3/4 linee e l'interfaccia periferica seriale quad (QSPI). L'area di visualizzazione può essere specificata nella GRAM interna tramite la funzione di indirizzo finestra.

Il [driver](#) è un driver indipendente dalla piattaforma, basato sul [driver generico MIPI](#). Implementa l'inizializzazione del display, supporta la rotazione del display e implementa la callback di flush del display. L'utente deve implementare solo due funzioni specifiche della piattaforma per inviare un comando o dati pixel al controller tramite SPI o bus parallelo. In genere, queste vengono implementate richiamando le funzioni della libreria SDK appropriate sulla piattaforma specificata.

Prerequisiti

Non ci sono prerequisiti.

Configurazione del driver

Abilitare il supporto del driver NV3007 in lv_conf.h, tramite la definizione del compilatore cmake o tramite KConfig.

```
#define LV_USE_NV3007 1
```

Utilizzo

È necessario implementare due funzioni dipendenti dalla piattaforma:

```
/* Send short command to the LCD. This function shall wait until the transaction finishes. */
void my_lcd_send_cmd(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, const uint8_t *param, size_t param_size)
{
    ...
}

/* Send large array of pixel data to the LCD. If necessary, this function has to do the byte-swapping. This function can do the transfer in the background. */
void my_lcd_send_color(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, uint8_t *param, size_t param_size)
{
    ...
}
```

Per creare un display basato su NV3007, si usa la funzione

```
/**
 * Create an LCD display with NV3007 driver
 * @param hor_res      horizontal resolution
 * @param ver_res      vertical resolution
 * @param flags        default configuration settings (mirror, RGB ordering, etc.)
 * @param send_cmd     platform-dependent function to send a command to the LCD controller (usually uses polling transfer)
 * @param send_color   platform-dependent function to send pixel data to the LCD controller (usually uses DMA transfer: must implement a 'ready' callback)
 * @return             pointer to the created display
 */
lv_display_t * lv_nv3007_create(uint32_t hor_res, uint32_t ver_res, lv_lcd_flag_t flags,
                                lv_nv3007_send_cmd_cb_t send_cmd_cb, lv_nv3007_send_color_cb_t send_color_cb);
```

Esempio Arduino

Ecco un semplice esempio di utilizzo del display NV3007 con il framework Arduino e l'interfaccia SPI:

```
#include <lvgl.h>
/* Platform-specific includes */
#include <SPI.h>

#define LCD_DC 21
#define LCD_CS 5
#define LCD_RST 4
#define LCD_SCK 18
#define LCD_MOSI 23
#define LCD_MISO -1
#define LCD_BL 15
#define SPI_CLK 40000000

#define BUFFER_SIZE 142 * 50
uint8_t buf[BUFFER_SIZE];

/* Define your platform-specific functions to send commands and data */
void my_lcd_send_cmd(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, const uint8_t *param, size_t param_size)
{
    SPI.beginTransaction(SPISettings(SPI_CLK, MSBFIRST, SPI_MODE0));
    /* Send command */
    digitalWrite(LCD_DC, LOW); /* command mode */
    digitalWrite(LCD_CS, LOW); /* CS low */
    SPI.transferBytes(cmd, NULL, cmd_size);

    /* Send parameters (if any) */
    if (param != NULL && param_size > 0)
    {
        digitalWrite(LCD_DC, HIGH); /* data mode */
        SPI.transferBytes(param, NULL, param_size);
    }
    digitalWrite(LCD_CS, HIGH); /* CS high */
    SPI.endTransaction();
}

void my_lcd_send_color(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, uint8_t *param, size_t param_size)
{
    SPI.beginTransaction(SPISettings(SPI_CLK, MSBFIRST, SPI_MODE0));
    digitalWrite(LCD_CS, LOW);

    /* Send the command first */
    digitalWrite(LCD_DC, LOW);
    SPI.transferBytes(cmd, NULL, cmd_size);

    /* Then send the pixel data */
    if (param && param_size > 0)
    {
        digitalWrite(LCD_DC, HIGH);
        SPI.transferBytes(param, NULL, param_size);
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

digitalWrite(LCD_CS, HIGH);
SPI.endTransaction();

/* Important: signal LVGL that we're done */
lv_display_flush_ready(display);

}

void setup()
{
    pinMode(LCD_BL, OUTPUT);
    digitalWrite(LCD_BL, HIGH); /* turn on backlight */

    pinMode(LCD_DC, OUTPUT);
    pinMode(LCD_CS, OUTPUT);
    pinMode(LCD_RST, OUTPUT);
    /* reset sequence */
    digitalWrite(LCD_RST, HIGH);
    delay(100);
    digitalWrite(LCD_RST, LOW);
    delay(120);
    digitalWrite(LCD_RST, HIGH);
    delay(120);

    SPI.begin(LCD_SCK, LCD_MISO, LCD_MOSI, LCD_CS); /* SCK, MISO, MOSI, SS */

    digitalWrite(LCD_CS, HIGH); /* disable device */
    delay(100); /* wait for device to stabilize */

    lv_init();

    lv_tick_set_cb(my_tick);

    /* Create NV3007 display */
    lv_display_t *display = lv_nv3007_create(142, 428, LV_DISPLAY_FLAG_NONE, my_lcd_send_cmd, my_lcd_send_color);
    lv_nv3007_set_gap(display, 0, 14);
    lv_display_set_rotation(display, LV_DISPLAY_ROTATION_270);
    lv_display_set_color_format(display, LV_COLOR_FORMAT_RGB565_SWAPPED);
    lv_display_set_buffers(display, buf, NULL, BUFFER_SIZE, LV_DISP_RENDER_MODE_PARTIAL);

    /* Create a simple label on the display */
    lv_obj_t *label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Hello NV3007!");
    lv_obj_align(label, LV_ALIGN_CENTER, 0, 0);
}

void loop()
{
    lv_task_handler();
    delay(5);
}

```

Per ulteriori dettagli e un esempio funzionante, consultare la [documentazione del driver generico MIPI](#).

4.9 Supporto del Sistema di Build

4.9.1 make

LVGL può essere facilmente integrato in qualsiasi progetto Makefile aggiungendo `include lvgl/lvgl.mk` al Makefile principale.

4.9.2 CMake

Panoramica

CMake è un generatore di sistemi di build multipiattaforma. Viene utilizzato per integrare facilmente un progetto/libreria in un altro progetto. Offre inoltre la possibilità di configurare la build con diverse opzioni, di abilitare o disabilitare componenti o di integrare l'esecuzione di script personalizzati durante la fase di configurazione/build.

LVGL include CMake in modo nativo, il che significa che è possibile utilizzarlo per configurare e compilare LVGL direttamente o integrarlo in una build CMake di livello superiore.

Questo progetto utilizza i preset CMake per garantire una build semplice. Maggiori informazioni su Cmake Presets qui: <https://cmake.org/cmake/help/latest/manual/cmake-presets.7.html>

Prerequisiti

È necessario installare

- CMake con GNU make o Ninja (per le build Linux). Aggiungere `ninja/make` al PATH!
- I prerequisiti sono elencati in `scripts/install-prerequisites.sh/bat`
- Un interprete Python3 per utilizzare KConfig.

Come compilare questo progetto con cmake

Compilazione da riga di comando

Il modo più semplice per compilare LVGL con cmake è usare le chiamate da riga di comando:

```
# Method 1
cd <lvgl_repo>
mkdir build
cd build
cmake ..           # Configure phase
cmake --build .    # Build phase

# Method 2
cd <lvgl_repo>
cmake -B build     # Configure phase
cmake --build build # build phase
```

Compilazione con i preset di CMake

Un altro modo per compilare questo progetto è utilizzare il file CMakePresets.json fornito o passare le opzioni tramite la riga di comando. Il file CMakePresets.json descrive alcune configurazioni di CMake e la fase di compilazione. È un modo per utilizzare rapidamente un set di opzioni predefinite di CMake.

Per ora, sono disponibili i seguenti preset di configurazione:

- **windows-base**: Una configurazione Windows, che utilizza VS MSVC. Usa `lv_conf.h` come sistema di configurazione.
- **windows-kconfig**: Una configurazione Windows, che utilizza VS MSVC. Utilizza Kconfig come sistema di configurazione.
- **linux-base**: Una configurazione Linux, che utilizza Ninja e GCC. Usa `lv_conf.h` come sistema di configurazione.
- **linux-kconfig**: Una configurazione Linux, che utilizza Ninja e GCC. Utilizza Kconfig come sistema di configurazione.

E questi sono i preset della build:

- **windows-base_dbg**: Build di Debug per Windows.
- **windows-base_rel**: Build di Release per Windows.
- **linux-base_dbg**: Build di Debug per Linux.
- **linux-base_rel**: Build di Release per Linux.

Ecco come compilare utilizzando i preset:

```
cmake --preset windows-base
cmake --build --preset windows-base_dbg
ctest --preset windows-base_dbg
```

Compilazione con IDE

Il metodo consigliato per utilizzare CMakePresets è un IDE compatibile con CMakePresets come

- VS 2022
- VS Code
- CLion

Basta caricare questo progetto nel proprio IDE e selezionare il preset desiderato.

Compilazione con la GUI di CMake

Aprire questo progetto con l'interfaccia grafica di CMake e selezionare il preset desiderato. Premendo il pulsante *generate*, CMake creerà file di soluzione (per VS) o file Ninja (per la build Ninja per Linux).

Sono disponibili i seguenti target.

- lvgl (la libreria vera e propria, obbligatoria)
- lvgl_thorvg (un'estensione per la grafica vettoriale, facoltativa)
- lvgl_examples (esempi di utilizzo, facoltativi)
- lvgl_demos (alcune demo, facoltative)

Tutti i target opzionali possono essere disabilitati impostando le variabili di cache appropriate. Se si utilizza cmake per installare lvgl, verranno create 3 cartelle.

- include/lvgl (contiene tutti gli header pubblici)
- bin (contiene tutti i binari (*.dll))
- lib (contiene tutti i file sorgente precompilati (*.lib))

Integrare LVGL nel progetto usando cmake

Il sistema cmake di LVGL è progettato per essere integrato in progetti di livello superiore. Per farlo, è sufficiente aggiungerlo al file CMakeLists.txt del progetto.

Questo frammento aggiunge LVGL e necessita di un file lv_conf.h presente accanto alla cartella lvgl:

```
add_subdirectory(lvgl)
```

Questo frammento imposta LVGL e gli indica quale file lv_conf.h utilizzare:

```
set(LV_BUILD_CONF_PATH path/to/my_lv_conf.h)
add_subdirectory(lvgl)
```

Questo frammento imposta LVGL e punta alla cartella in cui si trova lv_conf.h:

```
set(LV_BUILD_CONF_DIR path/to/directory)
add_subdirectory(lvgl)
```

Questo frammento aggiunge LVGL e specifica di utilizzare Kconfig come sistema di configurazione:

```
set(LV_BUILD_USE_KCONFIG ON)
add_subdirectory(lvgl)
```

Questo frammento aggiunge LVGL e specifica di utilizzare Kconfig come sistema di configurazione e di utilizzare un defconfig specifico:

```
set(LV_BUILD_USE_KCONFIG ON)
set(LV_BUILD_DEFCONFIG_PATH path/to/my_defconfig)
add_subdirectory(lvgl)
```

Per abilitare le demo e gli esempi, impostare queste opzioni:

```
set(CONFIG_LV_BUILD_EXAMPLES ON)
set(CONFIG_LV_BUILD_DEMOS ON)
add_subdirectory(lvgl)
```

Di seguito è riportato un elenco delle opzioni/variabili disponibili

Variabile/Opzione	Tipo	Descrizione
LV_BUILD_CONF_PATH	PATH	Consente di impostare un percorso personalizzato per <code>lv_conf.h</code>
LV_BUILD_CONF_DIR	PATH	Consente di impostare una directory contenente <code>lv_conf.h</code>
LV_BUILD_USE_KCONFIG	BOOLEAN	Quando impostato, KConfig viene utilizzato come sorgente della configurazione. Questa opzione è disabilitata per default.
LV_BUILD_DEFCONFIG_PATH	PATH	Specificare di utilizzare un file .defconfig al posto del file .config corrente in una configurazione di Kconfig.
LV_BUILD_LVGL_H_SYSTEM_INCLUDE	BOOLEAN	Abilitare se LVGL verrà installato sul sistema o se il sistema di build utilizza una sysroot. L'attivazione di questa opzione implica che le risorse generate dallo script di generazione dell'immagine includeranno <code>lvgl.h</code> come "include" di sistema. Ad esempio: <code>#include <lvgl.h></code> . Questa opzione è disabilitata per default.
LV_BUILD_LVGL_H_SIMPLE_INCLUDE	BOOLEAN	Se abilitata, le risorse includeranno <code>lvgl.h</code> come semplice "include"; questa opzione è abilitata di default.
LV_BUILD_SET_CONFIG_OPTS	BOOLEAN	Se abilitata, questa opzione esegue uno script che elabora la configurazione <code>lv_conf.h</code> /Kconfig utilizzando pcpp per generare le variabili CMake <code>CONFIG_LV_*</code> e <code>CONFIG_LV_BUILD_*</code> corrispondenti in base al contenuto di <code>lv_conf_internal.h</code> . Questo richiede python3 con venv e pip o l'accesso a un pcpp funzionante. Se si utilizza KConfig, questa opzione viene abilitata automaticamente.
CONFIG_LV_BUILD_DEMOS	BOOLEAN	Se abilitato, compila le demo
CONFIG_LV_BUILD_EXAMPLES	BOOLEAN	Se abilitato, compila gli esempi
CONFIG_LV_USE_THORVG_INTERNAL	BOOLEAN	Se abilitato, viene compilata la versione LVGL in-tree di ThorVG
CONFIG_LV_USE_PRIVATE_API	BOOLEAN	Se abilitato, vengono installati sul sistema gli header privati <code>*_private.h</code>

Nota

Se `LV_BUILD_SET_CONFIG_OPTS` o `LV_BUILD_USE_KCONFIG` sono abilitati, le opzioni/variabili che iniziano con il prefisso `CONFIG_*` vengono automaticamente impostate sui valori presenti in `lv_conf.h`

4.10 Binding

4.10.1 API di Output come Dati JSON

A partire dal 20 giugno 2024, LVGL viene fornito con uno script Python (`./scripts/gen_json/gen_json.py`) che legge i file header in LVGL e restituisce un formato JSON più intuitivo per l'API. Questo viene fatto in modo che i binding che generano codice automaticamente possano raccogliere facilmente le informazioni necessarie senza dover reinventare la ruota. Il formato JSON è stato scelto perché esistono librerie per la lettura di dati JSON in quasi tutti i linguaggi di programmazione.

Requisiti

- Python >= 3.10
- Pycparser >= 2.22: Libreria Python per leggere l'output del preprocessore C
- PyMSVC >= 0.4.0: Libreria Python per l'utilizzo del compilatore MSVC
- Un compilatore C: gcc per Linux, clang per OSX e MSVC per Windows
- Doxygen: utilizzato per leggere i commenti di Doxygen (la documentazione dell'API) dai file header.

Utilizzo

Opzioni della Riga di Comando

- `--output-path`: directory di output per il file JSON. Se non viene specificato, l'output verrà visualizzato su stdout.
- `--lvgl-config`: path per lv_conf.h (incluso il nome del file). Se non è impostato, verrà generato un file di configurazione con le opzioni LVGL più comuni attivate.
- `--develop`: lascia i file generati nella cartella temporanea.

Esempi

Utilizzo normale:

```
python ./scripts/gen_json/gen_json.py --output-path=json/output/directory --lvgl-config=path/to/lv_conf.h
```

Per eseguire un sottoprocesso dall'interno di uno script di generazione e leggere l'output da stdout:

```
python ./scripts/gen_json/gen_json.py --lvgl-config=path/to/lv_conf.h
```

Dati di Output

Il contenuto del file di output è un oggetto JSON di grandi dimensioni (`{...}`) con le seguenti coppie chiave/valore (queste sono le chiavi):

```
{
    "enums": [...],
    "functions": [...],
    "function_pointers": [...],
    "structures": [...],
    "unions": [...],
    "variables": [...],
    "typedefs": [...],
    "forward_decls": [...],
    "macros": [...]
}
```

Come si vede, il valore di ciascuno di questi elementi è un array. Gli elementi di ogni array sono oggetti JSON, ciascuno con una struttura univoca per il tipo indicato dal nome dell'elemento genitore (ad esempio "enum", "function", ecc.).

A ciascun oggetto è stata aggiunta una coppia chiave/valore (chiave = "json_type") per consentire il passaggio di un oggetto a una funzione generica e far sì che ciascun oggetto conosca il proprio tipo tramite questo campo. I possibili valori di "json_type" sono:

- "array": Il tipo dell'array viene utilizzato per identificare gli array.

Campi:

- "dim": numero di elementi nell'array
- "quals": array di qualificatori, ad esempio, "const"
- "type": Questo potrebbe essere disponibile o meno.
- "name": nome del tipo di dati

- "field": Questo tipo viene utilizzato per descrivere i campi in strutture e unioni. Viene utilizzato nell'array "fields" dei tipi "struct" "union" trattati di seguito.

Campi:

- "name": nome del campo
- "type": tipo di dato
- "json_type" trasporta il tipo di oggetto (ad esempio "enum", "function", ecc.) che identifica il gruppo di livello superiore da cui proviene
- "bitsize": Numero di bit per i campi bit, o null per i tipi di campo normali.
- "docstring": stringa contenente la documentazione estratta da Doxygen.

- "arg": Descrive un argomento di funzione.

Campi:

- "name": nome dell'argomento
- "type": tipo di dato
- "json_type" trasporta il tipo di oggetto (ad esempio "enum", "function", ecc.) che identifica il gruppo di livello superiore da cui proviene.
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "quals": array di tutti i qualificatori presenti, ad esempio "const"

- "forward_decl": Descrive una dichiarazione in avanti (forward). In LVGL ci sono strutture considerate private, ed è proprio ciò che descrivono.

Campi:

- "name": nome della dichiarazione forward
- "type": tipo di dato
- "json_type" trasporta il tipo di oggetto (ad esempio "enum", "function", ecc.) che identifica il gruppo di livello superiore da cui proviene.
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "quals": array di tutti i qualificatori presenti, ad esempio "const"

- "function_pointer": Descrive un puntatore a funzione. Questi vengono utilizzati per la registrazione di funzioni di callback in LVGL.

Campi:

- "name": nome del puntatore a funzione
- "type": tipo di ritorno della funzione
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "args": array degli oggetti "arg" descritti sopra
- "quals": array di tutti i qualificatori presenti, ad esempio "const"

- "variable": Descrive una variabile globale.

Campi:

- "name": nome della variabile
- "type": tipo di dato
- "json_type" trasporta il tipo di oggetto (ad esempio "enum", "function", ecc.) che identifica il gruppo di livello superiore da cui proviene.
- "docstring": stringa contenente la documentazione estratta da Doxygen.

- "quals": array di tutti i qualificatori presenti, ad esempio "const"
- "storage": array di tutti gli specificatori di classe di archiviazione presenti (ad esempio "auto", "static", "extern", ecc.)
- "special_type": Attualmente utilizzato solo per descrivere un argomento con ellissi (...) di una funzione.

Campi:

- "name": sempre "ellissi" (...)
- "primitive_type": Tipo di dati che non inizia con "lv_" e non termina con "_t". Confrontare con "lvgl_type". Include struct, union, tipi interi (ad esempio int, unsigned int), ecc.

Campi:

- "name": nome del tipo primitivo
- "enum": Enumerazioni C

Campi:

- "name": Se l'enumerazione è il risultato di un `typedef`, questo campo riporta il nome del tipo definito. Esempio: `lv_align_t`. (Non sempre disponibile).
- "type": tipo di enumeratori (sempre "int")
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "members": array di oggetti "enum_member"
- "enum_member": enumeratore (valore di enumerazione). Questo "json_type" si trova solo nell'array "members" di un oggetto "enum"

Campi:

- "name": nome dell'enumeratore
- "type": Se l'enumerazione è il risultato di un `typedef`, questo campo contiene il nome del tipo definito. Esempio: `lv_align_t`.
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "value": valore dell'enumeratore
- "lvgl_type": Tipo di dati definito in LVGL (inizializza con "lv_" e termina con "_t").

Campi:

- "name": nome del tipo
- "quals": array di tutti i qualificatori presenti, ad esempio "const"
- "struct": struct C

Campi:

- "name": nome della struct (tipo di dati se definito da `typedef`)
- "type": un oggetto "primitive_type" object {"name": "struct", "json_type": "primitive_type"}. (Vedere la definizione sopra).
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "fields": array di oggetti "field" (Vedere la definizione sopra).
- "quals": array di tutti i qualificatori presenti, ad esempio "const"
- "union": union C

Campi:

- "name": nome della union (tipo di dati se definito da `typedef`)

- "type": un oggetto "primitive_type" {"name": "union", "json_type": "primitive_type"}. (Vedere la definizione sopra).
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "fields": array di elementi "field".
- "quals": array di tutti i qualificatori presenti, ad esempio "const"
- "macro": macro C. Le informazioni che si possono raccogliere sulle macro sono limitate e nella maggior parte dei casi un binding richiederà che queste vengano aggiunte staticamente a un binding. Serve più per raccogliere le docstring che altro.

Campi:

- "name": nome della macro
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "ret_type": tipo di ritorno da una funzione. Questo verrà visualizzato solo nell'elemento "type" di un tipo "function".

Campi:

- "type": tipo di dato
- "json_type" trasporta il tipo di oggetto (ad esempio "enum", "function", ecc.) che identifica il gruppo di livello superiore da cui proviene.
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "function": funzione C

Campi:

- "name": nome della funzione
- "type": Un oggetto "ret_type". (Vedere la definizione sopra).
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "args": array of "arg" tipi json. (Vedere la definizione sopra).
- "stdlib_type": tipo C (il nome del tipo deriva da uno dei file header std*.h)

Campi:

- "name": nome del tipo
- "quals": array di tutti i qualificatori presenti, ad esempio "const"
- "unknown_type": Questo non dovrebbe essere visibile. In tal caso, è necessario apportare una modifica allo script. Aprire una segnalazione per farci sapere se si vede questo tipo.

Campi:

- "name": nome del tipo
- "quals": array di tutti i qualificatori presenti, ad esempio "const"
- "pointer": puntatore C

Campi:

- "type": tipo puntatore
- "json_type" trasporta il tipo di oggetto (ad esempio "enum", "function", ecc.) che identifica il gruppo di livello superiore da cui proviene.
- "quals": array di tutti i qualificatori presenti, ad esempio "const"
- "typedef": definizione del tipo C

Campi:

- "name": nome del tipo (ad esempio `lv_part_t`)
- "type": un oggetto "primitive_type" {"name": "uint32_t", "json_type": "stdlib_type"}. (Vedere la definizione sopra).
- "json_type" trasporta il tipo di oggetto (ad esempio "enum", "function", ecc.) che identifica il gruppo di livello superiore da cui proviene.
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "quals": array di tutti i qualificatori presenti, ad esempio "const"

Ecco un esempio abbreviato di come appare l'output.

```
{
  "enums": [
    {
      "name": "_lv_result_t",
      "type": {
        "name": "int",
        "json_type": "primitive_type"
      },
      "json_type": "enum",
      "docstring": "LVGL error codes.",
      "members": [
        {
          "name": "LV_RESULT_INVALID",
          "type": {
            "name": "_lv_result_t",
            "json_type": "lvgl_type"
          },
          "json_type": "enum_member",
          "docstring": "",
          "value": "0x0"
        },
        {
          "name": "LV_RESULT_OK",
          "type": {
            "name": "_lv_result_t",
            "json_type": "lvgl_type"
          },
          "json_type": "enum_member",
          "docstring": "",
          "value": "0x1"
        }
      ]
    }
  ],
  "functions": [
    {
      "name": "lv_version_info",
      "type": {
        "type": {
          "type": {
            "name": "char",
            "json_type": "primitive_type",
            "quals": [
              "const"
            ]
          },
          "json_type": "pointer",
          "quals": []
        },
        "json_type": "ret_type",
        "docstring": ""
      },
      "json_type": "function",
      "docstring": "",
      "args": [
        {
          "name": "null",
          "type": {
            "name": "void",
            "json_type": "primitive_type",
            "quals": []
          },
          "json_type": "arg",
          "docstring": "",
          "quals": []
        }
      ]
    }
  ],
  "function_pointers": [
    {
      "name": "lv_tif_walker",
      "type": {
        "type": {
          "name": "void",
          "json_type": "primitive_type",
          "quals": []
        },
        "json_type": "ret_type",
        "docstring": ""
      },
      "json_type": "function_pointer",
      "docstring": "",
      "args": [
        {
          "name": "ptr",
          "type": {
            "name": "void",
            "json_type": "primitive_type",
            "quals": []
          },
          "json_type": "arg",
          "docstring": "",
          "quals": []
        }
      ]
    }
  ]
}
```

(continues on next page)

(continua dalla pagina precedente)

```

        "json_type": "pointer",
        "quals": []
    },
    {
        "name": "size",
        "type": {
            "name": "size_t",
            "json_type": "stdlib_type",
            "quals": []
        },
        "json_type": "arg",
        "docstring": ""
    },
    {
        "name": "used",
        "type": {
            "name": "int",
            "json_type": "primitive_type",
            "quals": []
        },
        "json_type": "arg",
        "docstring": ""
    },
    {
        "name": "user",
        "type": {
            "type": {
                "name": "void",
                "json_type": "primitive_type",
                "quals": []
            },
            "json_type": "pointer",
            "quals": []
        },
        "json_type": "arg",
        "docstring": ""
    }
],
"quals": []
}
],
"structures": [
{
    "name": "_lv_grad_cache_t",
    "type": {
        "name": "struct",
        "json_type": "primitive_type"
    },
    "json_type": "struct",
    "docstring": null,
    "fields": [
        {
            "name": "color_map",
            "type": {
                "type": {
                    "name": "lv_color_t",
                    "json_type": "lvgl_type",
                    "quals": []
                },
                "json_type": "pointer",
                "quals": []
            },
            "json_type": "field",
            "bitsize": null,
            "docstring": ""
        },
        {
            "name": "opa_map",
            "type": {
                "type": {
                    "name": "lv_opa_t",
                    "json_type": "lvgl_type",
                    "quals": []
                },
                "json_type": "pointer",
                "quals": []
            },
            "json_type": "field",
            "bitsize": null,
            "docstring": ""
        },
        {
            "name": "size",
            "type": {
                "name": "uint32_t",
                "json_type": "stdlib_type",
                "quals": []
            },
            "json_type": "field",
            "bitsize": null,
            "docstring": ""
        }
    ]
},
{
    "name": "lv_global",
    "type": {
        "name": "lv_global_t",
        "json_type": "lvgl_type",
        "quals": []
    },
    "json_type": "variable",
    "docstring": "",
    "quals": [],
    "storage": [
        "extern"
    ]
}
],
"unions": [],
"variables": [
{
    "name": "lv_global",
    "type": {
        "name": "lv_global_t",
        "json_type": "lvgl_type",
        "quals": []
    },
    "json_type": "variable",
    "docstring": "",
    "quals": [],
    "storage": [
        "extern"
    ]
}
]
}
```

(continues on next page)

(continua dalla pagina precedente)

```

        ]
    },
    "typedefs": [
        {
            "name": "lv_pool_t",
            "type": {
                "type": {
                    "name": "void",
                    "json_type": "primitive_type",
                    "quals": []
                },
                "json_type": "pointer"
            },
            "json_type": "typedef",
            "docstring": "",
            "quals": []
        }
    ],
    "forward_decls": [
        {
            "name": "lv_fragment_managed_states_t",
            "type": {
                "name": "struct",
                "json_type": "primitive_type"
            },
            "json_type": "forward_decl",
            "docstring": "",
            "quals": []
        }
    ],
    "macros": [
        {
            "name": "ZERO_MEM_SENTINEL",
            "json_type": "macro",
            "docstring": ""
        }
    ]
}
}

```

4.10.2 Cpp

In progress: https://github.com/lvgl/lv_binding_cpp

4.10.3 JavaScript

Con `lv_binding_js` si può usare LVGL dall'interno di JavaScript.

Sfrutta il concetto di DOM virtuale di React per manipolare i componenti dell'interfaccia utente LVGL, offrendo agli utenti un'esperienza familiare simile a React.

Codice

Codice in Esecuzione su Dispositivo Reale

Funzionalità

- Supporta tutti i componenti LVGL nativi
- Supporta pienamente gli stili flex e grid LVGL
- Supporta la maggior parte degli stili LVGL, basta scrivere come CSS HTML5
- Supporta il caricamento dinamico delle immagini
- Supporta pienamente le animazioni LVGL

Demo

Vedere la cartella `demo`

Building

Di seguito sono riportate alcune note per sviluppatori su come compilare lvgljs sulla piattaforma nativa. Non si tratta di guide complete, ma includono note sulle librerie necessarie, i flag di compilazione, ecc.

lvgljs

- Note di build per dispositivi Linux embedded
- Note di build per SDL Simulator (Linux e macOS)

Bundle JS

- Note di build del bundle JS

Componenti

- View
- Image
- Button
- Text
- Input
- Textarea
- Switch
- Checkbox
- Dropdownlist
- ProgressBar
- Line
- Roller
- Keyboard
- Calendar
- Chart

Font

- Builtin-Symbol

Animazione

- Animazione

Stile

- position-size-layout
- boxing-model
- color
- flex
- grid
- font
- opacity
- display
- background
- scroll

- shadow
- recolor
- line
- transition
- transform

JSAPI

- network
- filesystem
- dimension

Ringraziamenti

lvgljs dipende dal seguente eccellente lavoro:

- [lvgl](#): Per creare splendide interfacce utente per qualsiasi MCU, MPU e tipo di display
- [QuickJS](#): Motore JavaScript
- [libuv](#): Layer di astrazione della piattaforma
- [curl](#): Client HTTP
- [txiki.js](#): Runtime di Tiny JavaScript

4.10.4 MicroPython

Cos'è MicroPython?

MicroPython è Python per microcontrollori. Utilizzando MicroPython, è possibile scrivere codice Python3 ed eseguirlo anche su un'architettura "bare-metal" con risorse limitate. Una delle sue potenti funzionalità è la possibilità di modificare il comportamento di un dispositivo modificando il codice Python su una memoria rimovibile (o interna), senza dover modificare il firmware del dispositivo.

Punti di Forza di MicroPython

Compatto

Si adatta e funziona in soli 256 kB di spazio di codice e 16 kB di RAM. Non è necessario alcun sistema operativo, anche se è possibile eseguirlo anche con un sistema operativo, volendo.

Compatibile

Cerca di essere il più compatibile possibile con Python standard (noto come CPython).

Versatile

Supporta molte architetture (x86, x86-64, ARM, ARM Thumb, Xtensa).

Interattivo

Non è necessario il ciclo di compilazione-flash-boot. Con REPL (prompt interattivo) è possibile digitare comandi ed eseguirli immediatamente, eseguire script, ecc.

Popolare

Sono supportate numerose piattaforme. La base utenti è in continua crescita. Fork degne di nota:

- [MicroPython](#)
- [CircuitPython](#)
- [MicroPython_ESP32_psRAM_LoBo](#)

Orientato agli Embedded

Fornito con moduli specifici per sistemi embedded, come il [modulo "machine"](#) per l'accesso all'hardware di basso livello (pin I/O, ADC, UART, SPI, I2C, RTC, timer ecc.)

Perché MicroPython + LVGL?

MicroPython non dispone di una buona libreria GUI nativa di alto livello. LVGL è una libreria GUI di alto livello Basata su [Componenti Object-Oriented](#) che è una candidata naturale per essere mappata in un linguaggio di livello superiore, come Python. LVGL è implementato in C e le sue API sono in C.

Ecco alcuni vantaggi dell'utilizzo di LVGL in MicroPython:

- Sviluppo di GUI in Python, un linguaggio di alto livello molto diffuso. Utilizzo di paradigmi come la Programmazione Object-Oriented.
- Di solito, lo sviluppo di GUI richiede più iterazioni per ottenere risultati ottimali. In C, ogni iterazione consiste in [Modifica del codice > Build > Flash > Run](#). In MicroPython è semplicemente [Modifica del codice > Run!](#) È anche possibile eseguire comandi in modo interattivo utilizzando [REPL](#) (il prompt interattivo)

MicroPython + LVGL potrebbe essere utilizzato per:

- Prototipazione rapida di GUI
- Ridurre il ciclo di modifica e messa a punto di una GUI
- Modellare una GUI in modo più astratto definendo widget compositi riutilizzabili, sfruttando le funzionalità del linguaggio Python come ereditarietà, chiusure, comprensione di liste, generatori, gestione delle eccezioni, interi a precisione arbitraria e altre.
- Rendere LVGL accessibile a un pubblico più vasto. Non è necessario conoscere il C per creare una buona GUI su un sistema embedded. Questo si integra perfettamente con la [visione di CircuitPython](#). CircuitPython è stato progettato pensando alla didattica, per facilitare l'approccio allo sviluppo embedded da parte di programmatore nuovi o inesperti.
- Creare strumenti per lavorare con LVGL a un livello superiore (ad esempio, un designer drag-and-drop).

Che aspetto ha?

È molto simile all'API C, ma orientata agli oggetti per i componenti LVGL.

Immergiamoci subito in un esempio!

Un semplice esempio

```
# Initialize
import display_driver
import lvgl as lv

# Create a button with a label
scr = lv.obj()
btn = lv.button(scr)
btn.align(lv.ALIGN.CENTER, 0, 0)
label = lv.label(btn)
label.set_text('Hello World!')
lv.screen_load(scr)
```

Come si Usa?

Simulatore Online

Per sperimentare con LVGL + MicroPython senza scaricare nulla, si può usare il simulatore online! È un simulatore LVGL + MicroPython completamente funzionante che funziona interamente nel browser e permette di modificare uno script Python ed eseguirlo.

Cliccare qui per sperimentare sul simulatore online

Molti *esempi LVGL* sono disponibili anche per MicroPython. Basta cliccare sul link!

Simulatore per PC

MicroPython è disponibile su numerose piattaforme. Una versione degna di nota è quella per "Unix", che consente di compilare ed eseguire MicroPython (+LVGL) su una macchina Linux. (Su una macchina Windows potrebbe essere necessario Virtual Box, WSL, MinGW, Cygwin, ecc.)

Cliccare qui per saperne di più sulla compilazione e l'esecuzione della versione Unix.

Piattaforme Embedded

In definitiva, l'obiettivo è eseguire il tutto su una piattaforma embedded. Sia MicroPython che LVGL possono essere utilizzati su numerose architetture embedded. **lv_micropython** è un fork di MicroPython+LVGL e attualmente supporta Linux, ESP32, STM32 e RP2. Può essere portato su qualsiasi altra piattaforma supportata da MicroPython.

- Sono necessari anche driver di input e display. È possibile utilizzare uno dei driver esistenti forniti con **lv_micropython** oppure creare driver di input/display personalizzati per l'hardware specifico.
- I driver possono essere implementati in C come modulo MicroPython o in Python puro.

lv_micropython contiene già i seguenti driver:

- Driver per display:
 - SDL su Linux
 - X11 su Linux
 - Specifici per ESP32:
 - * ILI9341
 - * ILI9488
 - * GC9A01
 - * ST7789
 - * ST7735
 - Generici (Python puro):
 - * ILI9341
 - * ST7789
 - * ST7735
- Driver di input:
 - SDL
 - X11
 - XPT2046
 - FT6X36
 - ESP32 ADC con touch resistivo

Dove si possono trovare maggiori informazioni?

- **lv_micropython README**
- **lv_binding_micropython README**
- Il forum **LVGL micropython** (Si è liberi di chiedere qualsiasi cosa!)
- Su MicroPython: [docs](#) e [forum](#)

- Post del Blog, un po' obsoleto.

Il Binding MicroPython è generato automaticamente!

- LVGL è un sottomodulo git all'interno di `lv_micropython` (LVGL è un sottomodulo git di `lv_binding_micropython` che a sua volta è un sottomodulo di `lv_micropython`).
- Durante la compilazione di `lv_micropython`, l'API C pubblica di LVGL viene analizzata e l'API MicroPython viene generata automaticamente. Ciò significa che `lv_micropython` fornisce l'API LVGL per **qualsiasi** versione di LVGL e generalmente non richiede modifiche al codice man mano che LVGL si evolve.

Convenzioni di codifica dell'API C di LVGL

Per un riepilogo delle convenzioni di codifica da seguire, consultare *Stile di Codifica*.

Gestione della Memoria

- Quando LVGL viene eseguito in MicroPython, tutte le allocazioni di memoria dinamica (`lv_malloc()`) vengono gestite dal gestore di memoria di MicroPython, che è **garbage-collected** (GC).
- Per evitare che il GC liberi la memoria prematuramente, tutta la RAM allocata dinamicamente deve essere raggiungibile dal GC.
- Il GC riconosce la maggior parte delle allocazioni, ad eccezione dei puntatori al Data Segment:
 - Puntatori che sono variabili globali
 - Puntatori che sono variabili globali statiche
 - Puntatori che sono variabili locali statiche

Tali puntatori devono essere definiti in modo specifico per renderli raggiungibili dal GC.

Identificare Il Problema

Un problema si verifica quando un puntatore di memoria allocata (valore di ritorno di `lv_malloc()`) viene memorizzato solo in una variabile puntatore **global**, **static global** o **static local** e non come parte di una **struct** precedentemente allocata o di un'altra variabile.

Risoluzione del Problema

- Sostituire la variabile locale/statica con (`LV_GLOBAL_DEFAULT() -> _var`)
- Includere `lv_global.h` nei file che usano `LV_GLOBAL_DEFAULT`
- Aggiungere `_var` a `lv_global_t` su `lv_global.h`

Esempio

Ulteriori Informazioni sulla Gestione della Memoria

- Nel README
- Nel Blog

Callback

In C, una callback è semplicemente un puntatore a funzione. In MicroPython, invece, dobbiamo registrare un *oggetto richiamabile MicroPython* per ogni callback. Pertanto, nel binding MicroPython dobbiamo registrare sia un puntatore a funzione che un oggetto MicroPython per ogni callback.

Abbiamo quindi definito una **convenzione di callback** per l'API C LVGL che prevede che gli header LVGL siano definiti in un certo modo. Le callback dichiarate secondo questa convenzione consentono al binding di registrare un

oggetto MicroPython accanto al puntatore a funzione quando si registra una callback e di accedere a tale oggetto quando la callback viene chiamata.

- L'idea di base è che abbiamo un campo `void * user_data` che viene utilizzato automaticamente dal binding MicroPython per salvare l'*oggetto richiamabile MicroPython* per una callback. Questo campo deve essere specificato durante la registrazione del puntatore alla funzione e fornito alla funzione di callback stessa.
- Sebbene chiamato "user_data", non è previsto che l'utente legga/scriva tale campo. Invece, il codice "collante" di MicroPython usa `user_data` per tenere traccia automaticamente dell'oggetto richiamabile di MicroPython. Il codice "collante" lo aggiorna quando la callback viene registrata e lo utilizza quando la callback viene chiamata per invocare una chiamata all'oggetto richiamabile originale.

Esistono alcune opzioni per definire una callback nell'API C LVGL:

- Opzione 1: `user_data` in una struct
 - Esiste una struttura che contiene un campo chiamato `void * user_data`
 - * Un puntatore a tale struttura viene fornito come **primo** argomento di una funzione di registrazione della callback.
 - * Un puntatore a tale struttura viene fornito come **primo** argomento della callback stessa.
- Opzione 2: `user_data` come argomento della funzione
 - Un parametro chiamato `void * user_data` viene fornito alla funzione di registrazione come **ultimo** argomento
 - * La callback stessa riceve `void *` come **ultimo** argomento
- Opzione 3: sia la callback che `user_data` sono campi di struct
 - L'API espone una struct con sia un membro puntatore a funzione che un membro `user_data`
 - * Il membro puntatore a funzione riceve la stessa struct come suo **primo** argomento

In pratica è anche possibile combinare queste opzioni, ad esempio fornire un puntatore a struct quando si registra una callback (opzione 1) e fornire l'argomento `user_data` quando si chiama la callback (opzione 2), **purché lo stesso user_data registrato venga passato alla callback quando viene chiamata**.

Esempi

- `lv_anim_t` contiene il campo `user_data`. `lv_anim_set_path_cb()` registra la callback `path_cb`. Sia `lv_anim_set_path_cb` che `lv_anim_path_cb_t` ricevono `lv_anim_t` come primo argomento
- Il campo `path_cb` può anche essere assegnato direttamente nel codice Python perché è un membro di `lv_anim_t` che contiene il campo `user_data`, e `lv_anim_path_cb_t` riceve `lv_anim_t` come primo argomento.
- `lv_imgfont_create()` registra `path_cb` e riceve `user_data` come ultimo argomento. Anche la callback `lv_imgfont_get_path_cb_t` riceve `user_data` come ultimo argomento.

Ulteriori Approfondimenti sulle Callback

- Nel Blog e nel README
- [v6.0] Callback conventions #1036
- Varie discussioni: [qui](#), [qui](#) e [qui](#)

4.10.5 PikaScript

Cos'è PikaScript?

PikaScript è un interprete Python progettato specificamente per microcontrollori e supporta un sottoinsieme della comune sintassi Python3.

È più leggero di MicroPython, richiedendo solo 32k di spazio per il codice e 4k di RAM, il che significa che può essere eseguito su stm32f103c8 (blue-pill) o persino stm32g030c8. D'altra parte, è possibile lasciare spazio prezioso per più materiale o aree di buffer più grandi.

È più semplice, pronto all'uso, funziona senza alcun porting o configurazione, non dipende dal sistema operativo o dal file system, offre un buon supporto per gli IDE più diffusi per piattaforme Windows come Keil, IAR, RT-Thread-Studio e, naturalmente, supporta le piattaforme di sviluppo Linux gcc.

È più intelligente, con un meccanismo esclusivo per i moduli C che consente di generare automaticamente i binding semplicemente scrivendo l'API per il modulo C in Python, senza dover affrontare il problema di dover scrivere manualmente macro o tabelle globali. D'altra parte, tutti i moduli C dispongono di suggerimenti intelligenti sofisticati, persino per i tipi degli argomenti.

Perché PikaScript + LVGL?

- PikaScript ora supporta le funzionalità principali di LVGL8 e queste API sono pienamente compatibili con MicroPython. Ciò significa che è possibile continuare a utilizzare il codice già scritto in MicroPython, ma occupando meno spazio di codice e RAM.
- Utilizzare suggerimenti dettagliati sul codice fino al tipo di parametro per una migliore esperienza di programmazione.
- Utilizzare un IDE più pratico, come i progetti di simulazione basati su Visual Studio.

Come si Presenta?

Ecco alcuni esempi di utilizzo di LVGL che PikaScript può già eseguire. Provengono principalmente dagli esempi della documentazione di LVGL.

LV_ARC

```
import pika_lvgl as lv
import PikaStdLib
mem = PikaStdLib.MemChecker()
# Create an Arc
arc = lv.arc(lv.screen_active())
arc.set_end_angle(200)
arc.set_size(150, 150)
arc.center()
print('mem used max: %0.2f kB' % (mem.getMax()))
print('mem used now: %0.2f kB' % (mem.getNow()))
```

LV_BAR

```
import pika_lvgl as lv
import PikaStdLib
mem = PikaStdLib.MemChecker()
bar1 = lv.bar(lv.screen_active())
bar1.set_size(200, 20)
bar1.center()
bar1.set_value(70, lv.ANIM.OFF)
print('mem used max: %0.2f kB' % (mem.getMax()))
print('mem used now: %0.2f kB' % (mem.getNow()))
```

LV_BTN

```
import pika_lvgl as lv
import PikaStdLib
mem = PikaStdLib.MemChecker()

def event_cb_1(evt):
    print('in evt1')
    print('mem used now: %0.2f kB' % (mem.getNow()))

def event_cb_2(evt):
    print('in evt2')
    print('mem used now: %0.2f kB' % (mem.getNow()))

btn1 = lv.btn(lv.screen_active())
btn1.align(lv.ALIGN.TOP_MID, 0, 10)
btn2 = lv.btn(lv.screen_active())
btn2.align(lv.ALIGN.TOP_MID, 0, 50)
```

(continues on next page)

(continua dalla pagina precedente)

```
btn1.add_event_cb(event_cb_1, lv.EVENT.CLICKED, 0)
btn2.add_event_cb(event_cb_2, lv.EVENT.CLICKED, 0)
print('mem used max: %0.2f kB' % (mem.getMax()))
print('mem used now: %0.2f kB' % (mem.getNow()))
```

LV_CHECKBOX

```
import pika_lvgl as lv
import PikaStdLib
mem = PikaStdLib.MemChecker()
cb = lv.checkbox(lv.screen_active())
cb.set_text("Apple")
cb.align(lv.ALIGN.TOP_LEFT, 0 ,0)
cb = lv.checkbox(lv.screen_active())
cb.set_text("Banana")
cb.add_state(lv.STATE.CHECKED)
cb.align(lv.ALIGN.TOP_LEFT, 0 ,30)
cb = lv.checkbox(lv.screen_active())
cb.set_text("Lemon")
cb.add_state(lv.STATE.DISABLED)
cb.align(lv.ALIGN.TOP_LEFT, 0 ,60)
cb = lv.checkbox(lv.screen_active())
cb.add_state(lv.STATE.CHECKED | lv.STATE.DISABLED)
cb.set_text("Melon")
cb.align(lv.ALIGN.TOP_LEFT, 0 ,90)
print('mem used max: %0.2f kB' % (mem.getMax()))
print('mem used now: %0.2f kB' % (mem.getNow()))
```

Come Funziona?

PikaScript dispone di un esclusivo strumento di binding intelligente per i moduli C.

Basta scrivere l'interfaccia Python in `pika_lvgl.pyi` (.pyi è un file di interfaccia Python).

```
# pika_lvgl.pyi
class arc(lv_obj):
    def set_end_angle(self, angle: int): ...
    def set_bg_angles(self, start: int, end: int): ...
    def set_angles(self, start: int, end: int): ...
```

Il precompilatore di PikaScript può poi associare automaticamente le seguenti funzioni C, semplicemente nominando le funzioni nel formato `modulo_classe_mетодо`, senza alcun lavoro aggiuntivo, e tutto il binding e la registrazione vengono eseguiti automaticamente.

```
/* pika_lvgl_arc.c */
void pika_lvgl_arc_set_end_angle(PikaObj* self, int angle) {
    lv_obj_t* lv_obj = obj_getPtr(self, "lv_obj");
    lv_arc_set_end_angle(lv_obj, angle);
}
void pika_lvgl_arc_set_bg_angles(PikaObj *self, int start, int end){
    lv_obj_t* lv_obj = obj_getPtr(self, "lv_obj");
    lv_arc_set_bg_angles(lv_obj, start, end);
}
void pika_lvgl_arc_set_angles(PikaObj *self, int start, int end){
    lv_obj_t* lv_obj = obj_getPtr(self, "lv_obj");
    lv_arc_set_angles(lv_obj, start, end);
}
```

Per utilizzare il modulo, basta `import pika_lvgl` e il precompilatore analizzerà automaticamente `main.py` e assocerà il modulo `pika_lvgl`.

```
$ ./rust-msc-latest-win10.exe
(pikascript) packages installed:
  pikascript-core==v1.10.0
  PikaStdLib==v1.10.0
  PikaStdDevice==v1.10.0
(pikascript) pika compiler:
  scanning main.py...
  binding pika_lvgl.pyi...
```

Il precompilatore è scritto in Rust, funziona su Windows e Linux ed è completamente open source.

Oltre ad associare i moduli C, il precompilatore compila gli script Python in bytecode nel PC, riducendo le dimensioni dello script e aumentandone la velocità.

Come si Usa?

Il repository di simulazione per Visual Studio è disponibile all'indirizzo https://github.com/pikasTech/lv_pikascript

CAPITOLO 5

Funzionalità Comuni dei Widget

I seguenti dettagli si applicano a tutti i tipi di widget.

5.1 Panoramica

Un Widget è l'**elemento costitutivo di base** dell'interfaccia utente LVGL.

Esempi di Widget: *Widget Base (e Screen), Pulsante, Etichetta, Immagine, Lista, Grafico e Area Testo*.

Vedere [Tutti i Widget](#) per visualizzare tutti i tipi di widget.

Le pagine seguenti presenteranno le funzionalità comuni a tutti i widget. Ad esempio, è possibile modificare le dimensioni e la posizione di qualsiasi widget. Possono avere stili, eventi, layout e molte altre funzionalità.

5.2 Convenzioni API

Nella maggior parte dei casi, le funzioni API dei widget LVGL sono strutturate come segue:

- `lv_ + <widget_name> + create(parent)`
- `lv_ + <widget_name> + set + <property>(widget, value)`
- `lv_ + <widget_name> + get + <property>(widget)`
- `lv_ + <widget_name> + add + <property>(widget)`

5.2.1 Attributi di Base

Tutti i tipi di widget condividono alcuni attributi di base:

- Posizione
- Dimensione
- Genitore
- Stili
- Callback degli eventi
- Flag come *Clickable*, *Scrollable*, ecc.
- Ecc.

È possibile impostare/ottenere questi attributi con le funzioni `lv_obj_set_...` e `lv_obj_get_....`. Ad esempio:

```
/* Set basic widget attributes */
lv_obj_set_size(btn1, 100, 50); /* Set a button's size */
lv_obj_set_pos(btn1, 20, 30); /* Set a button's position */
```

Per dettagli completi su posizione, dimensione, coordinate e layout, vedere [Posizione e Dimensione](#).

5.2.2 Attributi Specifici dei Widget

Anche i tipi di widget hanno attributi speciali. Ad esempio, uno slider ha:

- Valori minimi e massimi
- Valore corrente

Per questi attributi speciali, ogni tipo di widget può avere funzioni API univoche. Per esempio, per uno *Slider*:

```
/* Set slider-specific attributes */
lv_slider_set_range(slider1, 0, 100); /* Set the min and max values */
lv_slider_set_value(slider1, 40, LV_ANIM_ON); /* Set the current value (position) */
```

L'API dei Widget è descritta nella loro [Documentazione](#), ma è anche possibile consultare il rispettivo file header di ciascun Widget (ad esempio, `widgets/lv_slider.h`) per trovare un rapido riferimento ai prototipi di funzione con una breve documentazione su ciascuno.

5.2.3 Creazione di Widget

I Widget possono essere creati ed eliminati dinamicamente a runtime. Solo i widget attualmente creati (esistenti) consumano RAM.

Questo consente di creare uno Screen solo quando si clicca su un pulsante per aprirla e di eliminare gli Screen quando viene caricata una nuova schermata.

Le interfacce utente possono essere create in base all'ambiente corrente del dispositivo. Ad esempio, è possibile creare misuratori, grafici, barre e slider in base ai sensori attualmente collegati.

Ogni widget ha la sua funzione **create** con un prototipo come questo:

```
lv_obj_t * lv_<widget>_create(lv_obj_t * parent)
```

Le funzioni create hanno solo un parametro `parent` che specifica su quale widget creare quello nuovo.

Il valore restituito è un puntatore al widget creato di tipo `lv_obj_t *`.

5.2.4 Eliminazione del Widget

Esiste una funzione **delete** comune per tutti i tipi di widget. Elimina il widget e tutti i suoi elementi figlio.

```
void lv_obj_delete(lv_obj_t * widget);
```

`lv_obj_delete()` elimina il widget immediatamente. Se per qualsiasi motivo non è possibile eliminare immediatamente il widget, è possibile utilizzare `lv_obj_delete_async(widget)`, che eseguirà l'eliminazione alla successiva chiamata di `lv_timer_handler()`. Questo è utile, ad esempio, se si desidera eliminare l'elemento padre di un widget nel gestore `LV_EVENT_DELETE` del widget figlio. Una volta eliminato, la RAM occupata da un Widget viene liberata.

È possibile rimuovere tutti gli elementi figlio di un widget (ma non il widget stesso) utilizzando `lv_obj_clean(widget)`.

Si può usare `lv_obj_delete_delayed(widget, 1000)` per eliminare un widget dopo un certo periodo di tempo. Il ritardo è espresso in millisecondi.

Chiamando `lv_obj_null_on_delete(&widget)`, la variabile `lv_obj_t *` del widget verrà impostata su NULL quando il widget viene eliminato. Questo semplifica la verifica dell'esistenza o meno del widget.

Ecco un esempio che utilizza alcune delle funzioni sopra descritte:

```

static lv_obj_t * my_label; /* Static in the file so it stays valid */

/* Call it every 2000 ms */
void some_timer_callback(lv_timer_t * t)
{
    /* If the label is not created yet, create it and also delete it after 1000 ms */
    if(my_label == NULL) {
        my_label = lv_label_create(lv_screen_active());
        lv_obj_delete_delayed(my_label, 1000);
        lv_obj_null_on_delete(&my_label);
    }
    /* Move the label if it exists */
    else {
        lv_obj_set_x(my_label, lv_obj_get_x(my_label) + 1);
    }
}

```

5.3 Widget Tree

5.3.1 Genitori e Figli

LVGL memorizza i widget in una struttura padre-figlio (albero). L'elemento "root" [radice] è lo Screen, che non ha un elemento padre.

Quando viene creato un widget, un puntatore al suo elemento padre viene passato alla funzione "create".

LVGL fornisce molte funzioni utili per modificare l'albero dei widget e ottenere informazioni su di esso:

- `lv_obj_get_screen(widget)`: Ottiene lo "Screen" [schermata] di un widget (elemento padre "root").
- `lv_obj_get_parent(widget)`: Ottiene l'elemento padre corrente del widget.
- `lv_obj_set_parent(widget, new_parent)`: Sposta il widget in un nuovo elemento padre. Il widget diventerà il figlio più in alto (ultimo/più giovane) del nuovo genitore.
- `lv_obj_get_child(parent, idx)`: Restituisce un figlio specifico di un genitore. Alcuni esempi per `idx`:
 - 0: si ottiene il primo elemento figlio creato
 - 1: si ottiene il secondo elemento figlio creato
 - -1: si ottiene l'ultimo elemento figlio creato
- `lv_obj_get_child_by_type(parent, idx, &lv_slider_class)`: Simile a `lv_obj_get_child`, ma filtra gli elementi figlio con un tipo specificato.
- `lv_obj_find_by_name(parent, "text")`: Trova un widget con un nome specificato sotto un elemento padre. Non deve essere necessariamente un elemento figlio diretto.
- `lv_obj_get_child_by_name(parent, "container/button/text")`: Ottiene un widget navigando in un percorso utilizzando i nomi degli elementi padre (separati da /).
- `lv_obj_get_index(widget)`: Restituisce l'indice del widget nel suo elemento padre. Equivale al numero di elementi figlio maggiori nel widget padre.
- `lv_obj_move_to_index(widget, idx)`: Sposta il widget a un indice specificato. 0 è la posizione del widget figlio più vecchio, -1 è la posizione del widget figlio più giovane.
- `lv_obj_swap(widget1, widget2)`: Scambia le posizioni di due widget.

Per scorrere i widget figlio di un widget padre:

```

uint32_t i;
uint32_t count = lv_obj_get_child_count(parent);
for(i = 0; i < count; i++) {
    lv_obj_t * child = lv_obj_get_child(parent, i);
    /* Do something with `child` */
}

```

5.3.2 Nomi

Quando si crea un widget, il suo riferimento può essere memorizzato in una variabile puntatore `lv_obj_t*`. Per utilizzare questo widget in più punti del codice, la variabile può essere passata come parametro di funzione o resa globale. Tuttavia, questo approccio presenta alcuni svantaggi:

- L'utilizzo di variabili globali aggiunge nomi allo spazio dei nomi globale ed è quindi generalmente sconsigliato.

- Non è scalabile. Passare riferimenti a 20 widget come parametri di funzione non è l'ideale.
- Verificare se un widget esiste ancora o è stato eliminato richiede una logica aggiuntiva e può essere complicato.

Impostazione dei Nomi

Per risolvere questi problemi, LVGL introduce un potente sistema di denominazione dei widget abilitabile impostando `LV_USE_OBJ_NAME` in `lv_conf.h`.

È possibile assegnare un nome personalizzato utilizzando `lv_obj_set_name(obj, "name")` o `lv_obj_set_name_static(obj, "name")`. La variante "statica" richiede che il nome passato rimanga valido finché il widget esiste, poiché viene memorizzato solo il puntatore. In caso contrario, LVGL allocherà memoria per memorizzare una copia del nome.

Se un nome termina con `#`, LVGL lo sostituirà automaticamente con un indice basato sul numero di elementi simili con lo stesso nome di base. Se non viene specificato alcun nome, il valore predefinito è `<widget_type>_#`.

Di seguito è riportato un esempio che mostra come vengono risolti i nomi assegnati manualmente e automaticamente:

- Contenitore principale `lv_obj` chiamato "cont": "cont"
 - Contenitore `lv_obj` chiamato "header": "header"
 - * `lv_label` senza nome: "lv_label_0"
 - * `lv_label` chiamato "title": "title"
 - * `lv_label` senza nome: "lv_label_1" (È la terza etichetta, ma i widget con nomi personalizzati non vengono conteggiati)
 - Contenitore `lv_obj` chiamato "buttons":
 - * `lv_button` senza nome: "lv_button_0"
 - * `lv_button` chiamato "second_button": "second_button"
 - * `lv_button` senza nome: "lv_button_1"
 - * `lv_button` chiamato `lv_button_#`: "lv_button_2"
 - * `lv_button` chiamato `mybtn_#`: "mybtn_0"
 - * `lv_button` senza nome: "lv_button_2"
 - * `lv_button` chiamato `mybtn_#`: "mybtn_1"
 - * `lv_button` chiamato `mybtn_#`: "mybtn_2"
 - * `lv_button` chiamato `mybtn_#`: "mybtn_3"

Ricerca dei Widget

I widget si possono ricercare per nome in due modi:

1. Ottenere un figlio diretto per nome* utilizzando `lv_obj_get_child_by_name(parent, "child_name")`. Esempio: `lv_obj_get_child_by_name(header, "title")`

5.4 Screen

Gli Screen sono semplicemente dei Widget creati con `NULL` passato come elemento padre nella funzione "create". In quanto tali, costituiscono la "root" [radice] di un "Widget Tree" [Albero dei Widget].

In genere, a questo scopo viene utilizzato il "Base Widget", poiché possiede tutte le funzionalità necessarie alla maggior parte delle schermate. Ma è possibile utilizzare anche un Widget *Image* (`lv_image`) per creare un'immagine come sfondo per il "Widget Tree".

5.4.1 Creazione di Screen

Gli Screen vengono creati in questo modo:

```
lv_obj_t * scr1 = lv_obj_create(NULL);
```

Gli Screen vengono creati sul *Display di Default*. Poiché in genere è presente un solo display, le schermate vengono create automaticamente.

La risoluzione dello schermo corrisponde sempre alla risoluzione del display. Funzioni come `lv_obj_set_pos()` e `lv_obj_set_size()` non possono essere utilizzate sugli Screen.

Gli Screen creati possono essere eliminati con `lv_obj_delete(scr)`, ma non si deve eliminare lo *Lo Screen Attivo*.

5.4.2 Lo Screen Attivo

Sebbene a ogni oggetto *Display (lv_display)* possa essere associato un numero qualsiasi di Widget Screen, solo uno di questi Screen è considerato "Attivo" in un dato momento. Quello Screen è definito lo "Screen Attivo" del Display. Per questo motivo, su un display verrà visualizzato solo uno Screen e i relativi Widget figli alla volta.

Quando ogni oggetto *Display (lv_display)* è stato creato, è stato creato uno Screen di default con esso e impostato come "Screen Attivo".

Per ottenere un puntatore allo "Screen Attivo", si chiama `lv_screen_active()`.

Per impostare uno Screen come "Screen Attivo", si chiama `lv_screen_load()` o `lv_screen_load_anim()`.

5.4.3 Caricamento degli Screen

Per rendere visibile uno Screen, è necessario "caricarlo".

Il modo più semplice per caricare uno Screen è usare `lv_screen_load(scr1)`. Questo imposta `scr1` come Screen Attivo.

Usando `lv_screen_load_anim(scr, transition_type, time, delay, auto_del)`, il caricamento dello Screen può essere eseguito

- con una transizione opzionale
- se `auto_del` è `true`, lo Screen precedente viene automaticamente eliminata al termine di qualsiasi animazione di transizione.

Esistono i seguenti tipi di transizione:

- `LV_SCREEN_LOAD_ANIM_NONE`: Cambia immediatamente dopo `delay` millisecondi
- `LV_SCREEN_LOAD_ANIM_OVER_LEFT`, `LV_SCREEN_LOAD_ANIM_OVER_RIGHT`, `LV_SCREEN_LOAD_ANIM_OVER_TOP` e `LV_SCREEN_LOAD_ANIM_OVER_BOTTOM`: Spostano il nuovo Screen sopra quello corrente verso la direzione indicata
- `LV_SCREEN_LOAD_ANIM_OUT_LEFT`, `LV_SCREEN_LOAD_ANIM_OUT_RIGHT`, `LV_SCREEN_LOAD_ANIM_OUT_TOP` e `LV_SCREEN_LOAD_ANIM_OUT_BOTTOM`: Spostano il vecchio Screen sopra quello corrente nella direzione indicata
- `LV_SCREEN_LOAD_ANIM_MOVE_LEFT`, `LV_SCREEN_LOAD_ANIM_MOVE_RIGHT`, `LV_SCREEN_LOAD_ANIM_MOVE_TOP` e `LV_SCREEN_LOAD_ANIM_MOVE_BOTTOM`: Spostano sia lo Screen corrente che quello nuovo nella direzione indicata
- `LV_SCREEN_LOAD_ANIM_FADE_IN` e `LV_SCREEN_LOAD_ANIM_FADE_OUT`: Dissolvenza del nuovo Screen su quello precedente, o viceversa.

Il nuovo Screen diventerà attivo (restituito da `lv_screen_active()`) quando l'animazione inizia dopo il tempo `delay`.

Tutti gli eventi dei dispositivi di input (ad esempio touch, tasti, ecc.) sono disabilitati durante l'animazione dello Screen.

5.4.4 API

lv_display.h

5.5 Posizione e Dimensione

Similmente a molte altre parti di LVGL, il concetto di impostazione delle coordinate è ispirato ai CSS. LVGL non implementa completamente i CSS, ma ne fornisce un sottoinsieme comparabile (a volte con piccole modifiche).

In breve, questo significa:

- Se la posizione di un elemento padre cambia, anche gli elementi figlio si sposteranno di conseguenza.
- Se un elemento figlio è parzialmente o completamente esterno al suo elemento padre, le parti esterne non saranno visibili.
- Le coordinate impostate esplicitamente vengono memorizzate negli stili (posizione, dimensione, layout, ecc.).
- Supporta `min-width`, `max-width`, `min-height` e `max-height`.
- Supporta le unità di misura pixel, percentuale e content.
- $x = 0$, $y = 0$ indica l'angolo in alto a sinistra dell'elemento padre, più il "padding" [spaziatura] interno sinistro/superiore e la larghezza del bordo dell'elemento padre.
- `width` e `height` rappresentano le dimensioni complete. L'area del contenuto è più piccola a causa della spaziatura interna e della larghezza del bordo.
- È supportato un sottoinsieme di layout Flexbox e Grid.

In pratica, posizione, dimensione e allineamento sono tutte proprietà di stile.

Per semplicità, sono disponibili funzioni API dedicate per impostare posizione e dimensione. Tuttavia, per mantenere l'API di LVGL più snella, le funzioni relative alle coordinate meno comunemente utilizzate sono disponibili solo tramite le proprietà di stile.

Ecco come impostare posizione e dimensione direttamente:

```
lv_obj_set_x(widget, 10);
lv_obj_set_width(widget, 150);
```

Ottimamente tramite stili:

```
/* Using a style sheet */
static lv_style_t style_main;
lv_style_init(&style_main);
lv_style_set_width(&style_main, 100);

lv_obj_t * btn = lv_button_create(lv_screen_active());
lv_obj_add_style(btn, &style_main, LV_PART_MAIN);

/* Using a local style to set the width in the PRESSED state */
lv_obj_set_style_width(btn, 150, LV_PART_PRESSED);
```

Vantaggi dell'utilizzo degli stili:

- Applicare facilmente larghezza/altezza/etc. a più widget. Ad esempio, impostare tutti i cursori a 100x10 px.
- Controllo centralizzato sui valori (modifica in un unico punto).
- I valori possono essere sovrascritti selettivamente. Ad esempio, uno stile imposta un widget a 100x50, un altro sovrascrive solo la larghezza.
- La posizione o le dimensioni del widget possono cambiare in base allo stato. Ad esempio, 100 px di larghezza in `LV_STATE_DEFAULT` ma 120 px in `LV_STATE_PRESSED`.
- Abilitare le transizioni di stile per animare le variazioni di dimensione/posizione tra gli stati.

5.5.1 Note Importanti

Questa sezione descrive casi speciali in cui il comportamento di LVGL potrebbe essere inaspettato.

Calcolo Posticipato delle Coordinate

LVGL **non** ricalcola le coordinate immediatamente dopo una modifica. I widget vengono invece contrassegnati come "sporchi". LVGL ricalcola le posizioni, le dimensioni e i layout dei widget "sporchi" appena prima di ridisegnare lo schermo.

Quindi, se si modifica la posizione o le dimensioni di un widget e poi lo si legge immediatamente, è necessario chiamare prima `lv_obj_update_layout()` per assicurarsi che i valori siano aggiornati.

Nota: Poiché il layout e le dimensioni possono dipendere dall'oggetto padre, `lv_obj_update_layout()` ricalcola le coordinate per **tutti** i widget sullo schermo dell'oggetto specificato.

La Rimozione degli Stili fa Scomparire le Coordinate

Le coordinate vengono memorizzate come proprietà di stile in modo automatico. Quindi, se si utilizza:

```
lv_obj_set_x(widget, 20);
```

LVGL salva `x = 20` nello stile locale del widget.

Normalmente questo non è importante per l'utente, ma se si rimuovono gli stili di un widget, potrebbero essere rimosse anche le sue coordinate:

```
/* The size of widget1 will reset to default */
lv_obj_set_size(widget1, 200, 100);
lv_obj_remove_style_all(widget1);

/* widget2 will have 200x100 size in the end */
lv_obj_remove_style_all(widget2);
lv_obj_set_size(widget2, 200, 100);
```

5.5.2 Modello di Boxing

LVGL segue il modello CSS border-box:

<https://developer.mozilla.org/en-US/docs/Web/CSS/box-sizing>

Il "box" di un widget include:

bounding box

Larghezza/altezza totale del widget.

border width

Spessore del bordo.

padding

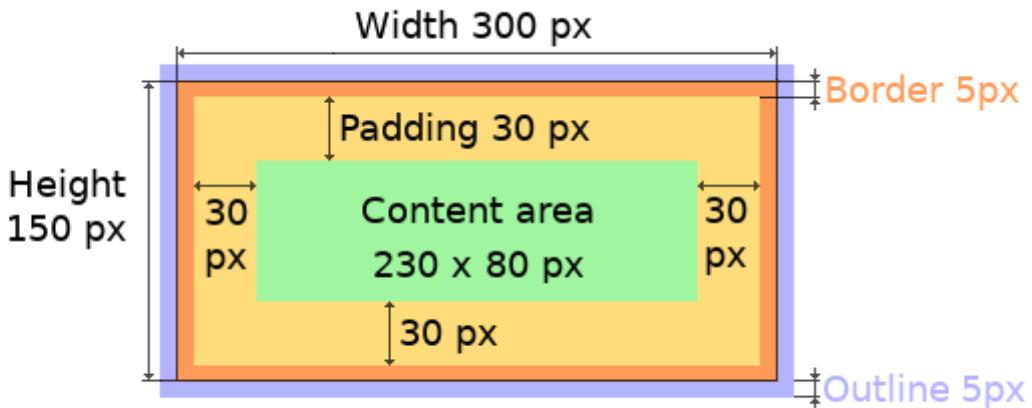
Spazio tra il bordo e il contenuto/gli elementi figlio.

margin

Spazio esterno al widget (utilizzato solo in alcuni layout).

content

Area del contenuto ("bounding box" meno bordo e padding interno).



Il bordo viene disegnato **all'interno** del "bounding box" [riquadro di delimitazione] e la spaziatura interna viene applicata **all'interno** del bordo. Il contorno (se presente) viene disegnato **all'esterno** del riquadro di delimitazione.

Utilizzo:

```
lv_obj_set_content_width(widget, 50);
```

Questo imposta la larghezza del *contenuto* e la larghezza effettiva del widget includerà il padding interno e il bordo.

Per recuperare informazioni sulle dimensioni:

```
int32_t w = lv_obj_get_width(widget);
int32_t h = lv_obj_get_height(widget);
int32_t content_w = lv_obj_get_content_width(widget);
int32_t content_h = lv_obj_get_content_height(widget);
```

5.5.3 Posizionamento Statico

Il modo più semplice per posizionare un widget è tramite valori in pixel:

```
lv_obj_set_x(widget, 10); /* From the left */
lv_obj_set_y(widget, 20); /* From the top */
lv_obj_set_pos(widget, 10, 20); /* Set both */
```

Questi valori sono relativi all'angolo in alto a sinistra del genitore, inclusi padding interno e bordo del genitore.

5.5.4 Posizionamento Dinamico

Allineare al Genitore

È possibile posizionare un widget rispetto a qualsiasi lato o angolo del genitore:

```
lv_obj_align(widget, LV_ALIGN_TOP_RIGHT, -10, 5);
```

Oppure:

```
lv_obj_set_align(widget, LV_ALIGN_BOTTOM_MID);
```

Punti di ancoraggio disponibili:

- *LV_ALIGN_TOP_LEFT*
- *LV_ALIGN_TOP_MID*
- *LV_ALIGN_TOP_RIGHT*
- *LV_ALIGN_BOTTOM_LEFT*
- *LV_ALIGN_BOTTOM_MID*
- *LV_ALIGN_BOTTOM_RIGHT*

- *LV_ALIGN_LEFT_MID*
- *LV_ALIGN_RIGHT_MID*
- *LV_ALIGN_CENTER*

Usare `lv_obj_center(widget)` come scorciatoia per centrare un widget.

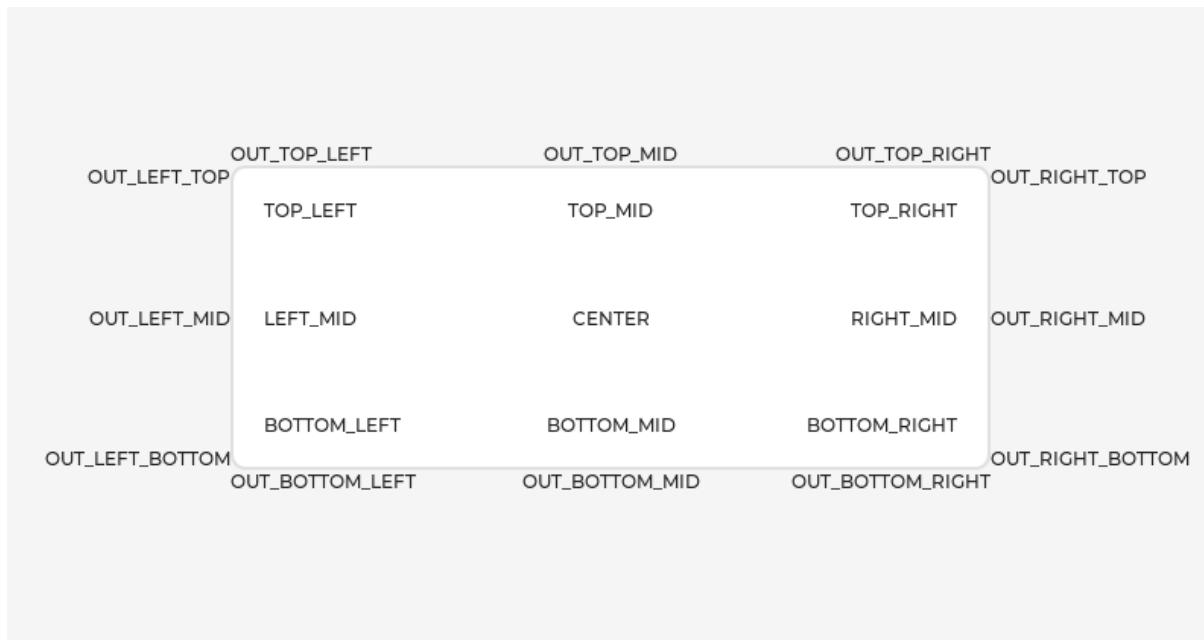
Allineare a un Altro Widget

Allineare a un widget di riferimento diverso:

```
lv_obj_align_to(label, btn, LV_ALIGN_OUT_TOP_MID, 0, -10);
```

Questo posiziona `label` 10 px sopra `btn`.

A differenza di `lv_obj_align`, questo allineamento non è persistente: riposizionare `btn` non aggiornerà `label`.



Layout

I layout `Flex` e `Grid` possono posizionare automaticamente i widget in righe, colonne o griglie 2D. Sono completamente responsive.

Percentuale

È possibile utilizzare le percentuali per il posizionamento dinamico:

```
lv_obj_set_x(widget, lv_pct(50));
```

100% indica l'area del contenuto del genitore (la dimensione escludendo padding e bordo).

5.5.5 Dimensionamento Statico

Impostare una dimensione fissa in pixel:

```
lv_obj_set_size(widget, 200, 100);
```

5.5.6 Dimensionamento Dinamico

Percentuale

Impostare larghezza o altezza come percentuale dell'area del contenuto del genitore:

```
lv_obj_set_height(widget, lv_pct(100));
```

Funziona anche in modo responsive, il che significa che le dimensioni del widget verranno aggiornate automaticamente al variare delle dimensioni del genitore.

Contenuto

Usare `LV_SIZE_CONTENT` per ridimensionare il widget in base ai suoi figli:

```
lv_obj_set_size(cont, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
lv_obj_t * label = lv_label_create(cont);
lv_label_set_text(label, "Some text");
```

Ignorato per i widget nascosti (`LV_OBJ_FLAG_HIDDEN`) o "floating" (`LV_OBJ_FLAG_FLOATING`).

Layout

I layout Flex e Grid possono anche allungare i widget per controllarne sia la posizione che le dimensioni.

5.5.7 Dimensioni Minime e Massime

LVGL supporta `min-width`, `max-width`, `min-height` e `max-height`.

Utilizzarli per impostare limiti quando si utilizzano `LV_SIZE_CONTENT` o dimensioni percentuali:

```
lv_style_set_max_height(&style, 200);
lv_obj_add_style(widget, &style, LV_STATE_DEFAULT);
```

È anche possibile utilizzare le percentuali:

```
lv_style_set_max_height(&style, lv_pct(50));
```

5.5.8 Traslazione della Posizione

Per spostare visivamente un widget dalla sua posizione attuale senza modificarne le coordinate di base:

```
lv_style_set_translate_y(&style_pressed, -20);
```

Utile per gli effetti del passaggio o della pressione del mouse.

La traslazione basata sulla percentuale è relativa alle dimensioni del widget (non a quelle del widget padre).

La traslazione delle coordinate si applica dopo il layout e influisce sulle barre di scorrimento e su `LV_SIZE_CONTENT`.

5.5.9 Trasformare la Dimensione

È anche possibile aumentare visivamente le dimensioni del widget senza influire sul layout:

```
lv_style_set_transform_width(&style_pressed, 10);
lv_style_set_transform_height(&style_pressed, 10);
```

A differenza di `translate`, questo è un effetto solo visivo e **non** influisce sul layout, sul comportamento della barra di scorrimento o sul ridimensionamento del contenitore.

5.5.10 Unità Indipendente dai DPI

Se si utilizza solo un numero intero come unità di misura per le dimensioni o la posizione, si intende il numero di pixel fisici sul display. Ciò significa anche che se si cambia il display con uno nuovo con le stesse dimensioni ma una densità di pixel maggiore, tutto apparirà più piccolo.

Utilizzando `lv_dpx(n)`, i valori vengono ridimensionati in base ai DPI del display per rendere il risultato indipendente dalla densità di pixel.

Il DPI di riferimento è 160, su cui `lv_dpx(100)` restituisce 100. Se il DPI fosse solo 80, `lv_dpx(100)` restituirebbe 50, ovvero solo 50 pixel.

Il DPI predefinito dei display è impostato da `LV_DPI_DEF` in `lv_conf.h`, ma può essere modificato con:

`lv_display_set_dpi(disp, new_dpi)`

`lv_dpx` può essere utilizzato ovunque sia necessario un valore in pixel (`x, y, width, height, radius, padding`, ecc.).

Si noti che `lv_dpx` viene valutato una sola volta, quindi se il DPI cambia in seguito, il valore non verrà aggiornato automaticamente.

Con l'aiuto di `lv_dpx`, è possibile ridimensionare solo i valori in pixel. Le immagini e i font devono comunque essere modificati manualmente per adattarsi al nuovo DPI.

5.6 Parti e Stati

5.6.1 Parti

I widget sono costituiti da più parti. Ad esempio, un *Widget Base* ha le parti "main" e "scrollbar", mentre uno *Slider* ha le parti "main", "indicator" e "knob". Le parti sono simili agli *pseudo-elements* in CSS.

In LVGL sono presenti le seguenti parti predefinite:

- `LV_PART_MAIN`: Un rettangolo simile a uno sfondo.
- `LV_PART_SCROLLBAR`: La/le barra/e di scorrimento.
- `LV_PART_INDICATOR`: L'indicatore, ad esempio per cursori, barre, interruttori o la casella di spunta di una checkbox.
- `LV_PART_KNOB`: Una maniglia utilizzata per regolare un valore.
- `LV_PART_SELECTED`: Indica l'opzione o la sezione attualmente selezionata.
- `LV_PART_ITEMS`: Utilizzato se il widget ha più elementi simili (ad esempio, celle di tabella).
- `LV_PART_CURSOR`: Indica una posizione specifica, ad esempio il cursore in un'area di testo o in un grafico.
- `LV_PART_CUSTOM_FIRST`: È possibile aggiungere parti personalizzate a partire da qui.

Lo scopo principale delle parti è consentire di definire lo stile dei singoli "componenti" di un widget. Sono descritte più dettagliatamente nella sezione *Panoramica sugli Stili*.

5.6.2 Stati

Un widget può trovarsi in una combinazione dei seguenti stati:

- `LV_STATE_DEFAULT`: Stato normale, rilasciato.
- `LV_STATE_CHECKED`: Stato attivato o "checked" [selezionato].
- `LV_STATE_FOCUSED`: Attivato tramite tastiera, encoder o cliccato tramite touchpad/mouse.
- `LV_STATE_FOCUS_KEY`: Attivato tramite tastiera o encoder, ma non tramite touchpad/mouse.
- `LV_STATE_EDITED`: In fase di modifica tramite encoder.
- `LV_STATE_HOVERED`: Sorvolato dal mouse (attualmente non supportato).
- `LV_STATE_PRESSED`: Premuto.
- `LV_STATE_SCROLLED`: "Scrolled".
- `LV_STATE_DISABLED`: Stato disabilitato.
- `LV_STATE_USER_1`: Stato personalizzato.

- *LV_STATE_USER_2*: Stato personalizzato.
- *LV_STATE_USER_3*: Stato personalizzato.
- *LV_STATE_USER_4*: Stato personalizzato.

Gli stati vengono solitamente modificati automaticamente dalla libreria quando l'utente interagisce con un widget (ad esempio, premendo, rilasciando, mettendo nel "focus"). Tuttavia, gli stati possono anche essere modificati manualmente. Per impostare o cancellare un determinato stato (lasciando inalterati gli altri stati), utilizzare:

- *lv_obj_add_state(widget, LV_STATE_...)*
- *lv_obj_remove_state(widget, LV_STATE_...)*

In entrambi i casi, è possibile eseguire l'operazione OR bit per bit su più valori di stato. Ad esempio:
lv_obj_add_state(widget, LV_STATE_PRESSED | LV_STATE_CHECKED)

Per saperne di più sugli stati, consultare la sezione correlata in *Panoramica sugli Stili*.

5.7 I Layer

Quando il termine "layer" viene utilizzato nella documentazione LVGL, può riferirsi a diversi aspetti:

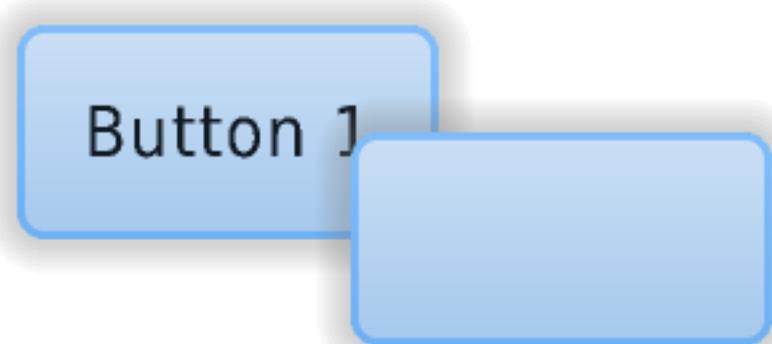
1. per i Widget, *Ordine di Creazione* crea una stratificazione naturale di Widget;
2. nel contesto del "pixel rendering" (disegno), ci sono *Layer di Disegno*;
3. i *Layer dello Screen* permanenti fanno parte di ciascun oggetto *Display* (*lv_display*).

Il punto 1 è trattato di seguito. I punti 2 e 3 sono trattati rispettivamente in *Layer di Disegno* e *Layer dello Screen*.

5.7.1 Ordine di Creazione

Per default, LVGL disegna i nuovi Widget sopra i vecchi Widget.

Ad esempio, supponiamo di aggiungere un pulsante a un Widget padre denominato button1 e poi un altro pulsante denominato button2. Quindi button1 (insieme ai suoi Widget figli) sarà sullo sfondo e potrà essere coperto da button2 e dai suoi figli.



```
/* Create a screen */
lv_obj_t * scr = lv_obj_create(NULL);
lv_screen_load(scr); /* Load the screen */

/* Create 2 buttons */
lv_obj_t * btl1 = lv_button_create(scr); /* Create the first button on the screen */
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_pos(btn1, 60, 40);           /* Set the position of the first button */

lv_obj_t * btn2 = lv_button_create(scr); /* Create the second button on the screen */
lv_obj_set_pos(btn2, 180, 80);          /* Set the position of the second button */

/* Add labels to the buttons */
lv_obj_t * label1 = lv_label_create(btn1); /* Create a label on the first button */
lv_label_set_text(label1, "Button 1");    /* Set the text of the label */

lv_obj_t * label2 = lv_label_create(btn2); /* Create a label on the second button */
lv_label_set_text(label2, "Button 2");    /* Set the text of the label */

/* Delete the second label */
lv_obj_delete(label2);

```

Modifica dell'Ordine

Esistono tre modi esplicativi per portare un Widget in primo piano:

- Si usa `lv_obj_move_to_index(widget, idx)` per spostare un widget a un indice specificato nell'ordine dei figli.
 - 0: background [sfondo]
 - child_num - 1: primo piano
 - < 0: conta dall'alto, per andare avanti (verso l'alto): `lv_obj_move_to_index(widget, lv_obj_get_index(widget) - 1)`
- Si usa `lv_obj_swap(widget1, widget2)` per scambiare la posizione relativa del layer di due widget.
- Quando si usa `lv_obj_set_parent(widget, new_parent)`, widget sarà in primo piano rispetto a new_parent.

5.7.2 API

`lv_obj_tree.h`

5.8 Stili

5.8.1 Panoramica sugli Stili

Gli stili vengono utilizzati per impostare l'aspetto dei Widget. Gli stili in LVGL sono fortemente ispirati ai CSS. Il concetto, in breve, è che uno stile è una variabile `lv_style_t` che può contenere proprietà come lo spessore del bordo, il font, il colore del testo e così via. È simile a una `class` in CSS.

- Gli stili possono essere assegnati ai Widget per modificarne l'aspetto. Al momento dell'assegnazione, è possibile specificare la parte di destinazione (`pseudo-element` in CSS) e lo stato di destinazione (`pseudo-class` in CSS). Ad esempio, è possibile aggiungere `style_blue` alla manopola di uno slider quando è premuto.
- Lo stesso stile può essere utilizzato da un numero qualsiasi di widget.
- Gli stili sono sovrapponibili, il che significa che è possibile assegnare più stili a un widget e ogni stile può avere proprietà diverse. Pertanto, non tutte le proprietà devono essere specificate in uno stile. LVGL cercherà una proprietà finché non verrà definita da uno stile oppure utilizzerà un valore predefinito se non è specificato da nessuno degli stili. Ad esempio, `style_btn` può generare un pulsante grigio predefinito e `style_btn_red` può aggiungere solo `background-color:red` per sovrascrivere il colore di sfondo.
- Lo stile aggiunto più di recente ha la precedenza. Ciò significa che, se una proprietà è specificata in due stili, verrà utilizzato lo stile più recente nel widget.
- Alcune proprietà (ad esempio, il colore del testo) possono essere ereditate da uno o più elementi padre se non sono specificate in un Widget.
- I Widget possono anche avere `stili locali` con precedenza maggiore rispetto agli stili "normali".
- A differenza di CSS (dove le `pseudo-classes` descrivono stati diversi, ad esempio `:focus`), in LVGL una proprietà viene assegnata a un determinato stato.
- Le transizioni possono essere applicate quando il Widget cambia stato.

Per l'elenco completo delle proprietà di stile, cliccare [qui](#).

Stati

I Widget possono trovarsi nella seguente combinazione di stati:

- `LV_STATE_DEFAULT`: (0x0000) Stato normale, rilasciato
- `LV_STATE_CHECKED`: (0x0001) Stato attivato o selezionato
- `LV_STATE_FOCUSED`: (0x0002) Focus tramite tastiera, encoder o clic tramite touchpad/mouse
- `LV_STATE_FOCUS_KEY`: (0x0004) Focus tramite tastiera o encoder ma non tramite touchpad/mouse
- `LV_STATE_EDITED`: (0x0008) Modifica tramite encoder
- `LV_STATE_HOVERED`: (0x0010) Sovrapposto dal mouse
- `LV_STATE_PRESSED`: (0x0020) Premuto
- `LV_STATE_SCROLLED`: (0x0040) In scorrimento
- `LV_STATE_DISABLED`: (0x0080) Stato disabilitato
- `LV_STATE_USER_1`: (0x1000) Stato personalizzato
- `LV_STATE_USER_2`: (0x2000) Stato personalizzato
- `LV_STATE_USER_3`: (0x4000) Stato personalizzato
- `LV_STATE_USER_4`: (0x8000) Stato personalizzato

Un Widget può trovarsi in una combinazione di stati, ad esempio essere attivo e premuto contemporaneamente. Questo è rappresentato come `LV_STATE_FOCUSED | LV_STATE_PRESSED`.

È possibile aggiungere uno stile a qualsiasi stato o combinazione di stati. Ad esempio, impostando un colore di sfondo diverso per gli stati di default e premuto. Se una proprietà non è definita in uno stato, verrà utilizzata la proprietà dello stato che corrisponde meglio. In genere, questo significa che viene utilizzata la proprietà con `LV_STATE_DEFAULT`. Se la proprietà non è impostata nemmeno per lo stato di default, verrà utilizzato il valore di default. (Vedere più avanti).

Poiché `LV_PART_MAIN` e `LV_STATE_DEFAULT` hanno entrambi valore zero, si può semplicemente passare 0 come argomento `selector` invece di `LV_PART_MAIN | LV_STATE_DEFAULT` come scorciatoia quando si aggiungono stili a un oggetto.

Cosa significa "proprietà dello stato più corrispondente"?

Gli stati hanno una precedenza, indicata dal loro valore (vedere l'elenco sopra). Un valore più alto significa una precedenza più alta. Per determinare quale proprietà dello stato utilizzare, prendiamo un esempio. Si immagini che il colore di sfondo sia definito in questo modo:

- `LV_STATE_DEFAULT`: bianco
- `LV_STATE_PRESSED`: grigio
- `LV_STATE_FOCUSED`: rosso

1. Inizialmente il Widget è nello stato di default, quindi è un caso semplice: la proprietà è perfettamente definita nello stato corrente del Widget, ovvero bianco.
2. Quando si preme il Widget, ci sono 2 proprietà correlate: il default con bianco (il default è relativo a tutti gli stati) e "pressed" con grigio. Lo stato "pressed" ha una precedenza 0x0020, che è superiore alla precedenza 0x0000 dello stato di default, quindi verrà utilizzato il colore grigio.
3. Quando il widget è attivo, accade la stessa cosa di quando si trova nello stato "pressed" e verrà utilizzato il colore rosso. (Lo stato "focused" ha una precedenza maggiore rispetto allo stato di default).
4. Quando il Widget ha il focus ed è premuto, funzionerebbero sia il grigio che il rosso, ma lo stato pressed ha una precedenza maggiore rispetto allo stato focused, quindi verrà utilizzato il colore grigio.

5. È possibile impostare, ad esempio, il colore rosa per `LV_STATE_PRESSED | LV_STATE_FOCUSED`. In questo caso, questo stato combinato ha una precedenza $0x0020 + 0x0002 = 0x0022$, che è superiore alla precedenza dello stato pressed, quindi verrà utilizzato il colore rosa.
6. Quando il Widget è nello stato checked, non esiste alcuna proprietà per impostare il colore di background per questo stato. Pertanto, in mancanza di un'opzione migliore, il Widget rimane bianco per la proprietà dello stato di default.

Alcune note pratiche:

- La precedenza (valore) degli stati è piuttosto intuitiva ed è qualcosa che l'utente si aspetterebbe naturalmente. Esempio: se un Widget ha il focus, l'utente vorrà comunque vedere se è premuto, quindi lo stato "pressed" ha una precedenza maggiore. Se lo stato con focus avesse una precedenza maggiore, sovrascriverebbe il colore di "pressed".
- Per impostare una proprietà per tutti gli stati (ad esempio, il colore di sfondo rosso), è sufficiente impostarla sullo stato di default. Se il widget non riesce a trovare una proprietà per il suo stato attuale, tornerà alla proprietà dello stato di default.
- Utilizzare gli stati con OR per descrivere le proprietà in casi complessi (ad esempio pressed + checked + focused).
- Potrebbe essere una buona idea utilizzare elementi di stile diversi per stati diversi. Ad esempio, trovare colori di sfondo per gli stati released, pressed, checked + pressed, focused, focused + pressed, focused + pressed + checked, ecc. è piuttosto difficile. In alternativa, ad esempio, utilizzare il colore di sfondo per gli stati pressed e checked e indicare lo stato focused con un colore del bordo diverso.

Stili a Cascata

Non è necessario impostare tutte le proprietà in un unico stile. È possibile aggiungere più stili a un Widget e fare in modo che quest'ultimo modifichi o estenda l'aspetto. Ad esempio, creare uno stile generico per i pulsanti grigi e crearne uno nuovo per i pulsanti rossi, in cui viene impostato solo il nuovo colore di sfondo.

Questo è molto simile a quello dei CSS, quando le classi utilizzate sono elencate come `<div class=".btn .btn-red">`.

Gli stili aggiunti in seguito hanno la precedenza su quelli impostati in precedenza. Quindi, nell'esempio del pulsante grigio/rosso sopra, lo stile del pulsante normale dovrebbe essere aggiunto per primo e quello rosso per secondo. Tuttavia, la precedenza degli stati viene comunque presa in considerazione. Esaminiamo il seguente caso:

- lo stile base del pulsante definisce il colore grigio scuro per lo stato di default e il colore grigio chiaro per lo stato pressed.
- lo stile del pulsante rosso definisce il colore di sfondo come rosso solo nello stato di default.

In questo caso, quando il pulsante viene rilasciato (è nello stato di default), sarà rosso perché è stata trovata una corrispondenza perfetta nello stile aggiunto più di recente (rosso). Quando il pulsante viene premuto, il colore grigio chiaro è una corrispondenza migliore perché descrive perfettamente lo stato corrente, quindi il pulsante sarà grigio chiaro.

Ereditarietà

Alcune proprietà (in genere quelle relative al testo) possono essere ereditate dagli stili del Widget genitore. L'ereditarietà viene applicata solo se la proprietà specificata non è impostata negli stili del Widget (nemmeno nello stato di default). In questo caso, se la proprietà è ereditabile, il valore della proprietà verrà ricercato nella gerarchia padre finché un Widget non ne specifica un valore. I genitori utilizzeranno il proprio stato per determinare il valore. Quindi, un pulsante è "pressed" [premuto] e il colore del testo proviene da un genitore, verrà utilizzato il colore del testo "pressed".

Parti

I Widget possono essere composti da *parti*, ciascuna delle quali può avere i propri stili.

In LVGL sono presenti le seguenti parti predefinite:

- *LV_PART_MAIN*: (0x000000) Uno background simile a un rettangolo
- *LV_PART_SCROLLBAR*: (0x010000) Le barre di scorrimento
- *LV_PART_INDICATOR*: (0x020000) Indicatore, ad esempio per cursore, barra, interruttore o checkbox
- *LV_PART_KNOB*: (0x030000) Come una maniglia da afferrare per regolare un valore
- *LV_PART_SELECTED*: (0x040000) Indica l'opzione o la sezione attualmente selezionata
- *LV_PART_ITEMS*: (0x050000) Utilizzato se il widget ha più elementi simili (ad esempio, celle di tabella)
- *LV_PART_CURSOR*: (0x060000) Contrassegna un punto specifico, ad esempio il cursore dell'area di testo o del grafico
- *LV_PART_CUSTOM_FIRST*: (0x080000) Da qui è possibile aggiungere identificatori di parti personalizzati.
- *LV_PART_ANY*: (0x0F0000) In alcune funzioni è possibile utilizzare un valore speciale per indirizzare tutte le parti.

Ad esempio, uno *Slider* ha tre parti:

- Main (background)
- Indicatore
- Knob [maniglia]

Ciò significa che tutte e tre le parti dello slider possono avere i propri stili. Più avanti si vedrà come aggiungere stili a Widget e a parti.

Poiché *LV_PART_MAIN* e *LV_STATE_DEFAULT* hanno entrambi valore zero, si può semplicemente passare 0 come argomento selector invece di *LV_PART_MAIN* | *LV_STATE_DEFAULT* come scorciano quando si aggiungono stili a un oggetto.

Proprietà che Richiedono Nuovi Layer

Se le proprietà *opa*, *blend_mode*, *transform_angle* o *transform_zoom* sono impostate su un valore non di default, LVGL crea un'istantanea del widget e dei suoi elementi secondari per fondere l'intero widget con le proprietà di opacità, modalità di fusione e trasformazione impostate.

Queste proprietà hanno questo effetto solo sulla parte **MAIN** del widget.

L'istantanea creata è chiamata "intermediate layer" [livello intermedio] o semplicemente "layer". Se solo *opa* e/o *blend_mode* sono impostate su un valore non predefinito, LVGL può creare il layer da blocchi più piccoli. La dimensione di questi blocchi può essere configurata tramite le seguenti proprietà in *lv_conf.h*:

- *LV_LAYER_SIMPLE_BUF_SIZE*: [bytes] la dimensione ottimale del buffer target. LVGL tenterà di allocare questa dimensione di memoria.
- *LV_LAYER_SIMPLE_FALLBACK_BUF_SIZE*: [bytes] utilizzati se *LV_LAYER_SIMPLE_BUF_SIZE* non è stato possibile allocare.

Se sono state utilizzate anche le proprietà di trasformazione, il layer non può essere renderizzato in blocchi.

Proprietà Tipiche del Background

Nella documentazione dei widget si troveranno frasi come "Il Widget XY utilizza le proprietà tipiche del background". Queste "proprietà tipiche del background" sono le proprietà a cui si fa riferimento:

- Background
- Border [bordatura]
- Outline [contorno]

- Shadow
- Padding [riempimento]
- Trasformazione di larghezza e altezza
- Traslazione X e Y

Esempi

Stili per le Dimensioni

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/**
 * Using the Size, Position and Padding style properties
 */
void lv_example_style_1(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_radius(&style, 5);

    /*Make a gradient*/
    lv_style_set_width(&style, 150);
    lv_style_set_height(&style, LV_SIZE_CONTENT);

    lv_style_set_pad_ver(&style, 20);
    lv_style_set_pad_left(&style, 5);

    lv_style_set_x(&style, lv_pct(50));
    lv_style_set_y(&style, 80);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label, "Hello");
}

#endif
```

Stili per il Background

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/**
 * Using the background style properties
 */
void lv_example_style_2(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_radius(&style, 5);

    /*Make a gradient*/
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    static lv_grad_desc_t grad;
    grad.dir = LV_GRAD_DIR_VER;
    grad.stops_count = 2;
    grad.stops[0].color = lv_palette_lighten(LV_PALETTE_GREY, 1);
    grad.stops[0].opa = LV_OPA_COVER;
    grad.stops[1].color = lv_palette_main(LV_PALETTE_BLUE);
    grad.stops[1].opa = LV_OPA_COVER;

    /*Shift the gradient to the bottom*/
    grad.stops[0].frac = 128;
    grad.stops[1].frac = 192;

    lv_style_set_bg_grad(&style, &grad);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

Stili per i Bordi

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/**
 * Using the border style properties
 */
void lv_example_style_3(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 10);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_set_bg_opa(&style, LV_OPA_COVER);
lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

/*Add border to the bottom+right*/
lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));
lv_style_set_border_width(&style, 5);
lv_style_set_border_opa(&style, LV_OPA_50);
lv_style_set_border_side(&style, (lv_border_side_t)(LV_BORDER_SIDE_BOTTOM | LV_BORDER_SIDE_RIGHT));

/*Create an object with the new style*/
lv_obj_t * obj = lv_obj_create(lv_screen_active());
lv_obj_add_style(obj, &style, 0);
lv_obj_center(obj);
}

#endif

```

Stili per i Contorni

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the outline style properties
 */
void lv_example_style_4(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add outline*/
    lv_style_set_outline_width(&style, 2);
    lv_style_set_outline_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_outline_pad(&style, 8);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif

```

Stili per le Ombre

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the Shadow style properties
 */
void lv_example_style_5(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add a shadow*/
    lv_style_set_shadow_width(&style, 55);
    lv_style_set_shadow_color(&style, lv_palette_main(LV_PALETTE_BLUE));

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif

```

Stili per le Immagini

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Using the Image style properties
 */
void lv_example_style_6(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 3));
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_set_image_recolor(&style, lv_palette_main(LV_PALETTE_BLUE));
lv_style_set_image_recolor_opa(&style, LV_OPA_50);
lv_style_set_transform_rotation(&style, 300);

/*Create an object with the new style/
lv_obj_t * obj = lv_image_create(lv_screen_active());
lv_obj_add_style(obj, &style, 0);

LV_IMAGE_DECLARE(img_cogwheel_argb);
lv_image_set_src(obj, &img_cogwheel_argb);

lv_obj_center(obj);
}

#endif

```

Stili per gli Archi

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_ARC

/***
 * Using the Arc style properties
 */
void lv_example_style_7(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_arc_color(&style, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_arc_width(&style, 4);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_arc_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}
#endif

```

Stili per il Testo

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

/***
 * Using the text style properties
 */
void lv_example_style_8(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 2));
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_pad_all(&style, 10);

    lv_style_set_text_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_text_letter_space(&style, 5);
    lv_style_set_text_line_space(&style, 20);
    lv_style_set_text_decor(&style, LV_TEXT_DECOR_UNDERLINE);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_label_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_label_set_text(obj, "Text of\n"
                         "a label");

    lv_obj_center(obj);
}
#endif

```

Stili per le Linee

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LINE

/***
 * Using the line style properties
 */
void lv_example_style_9(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_line_color(&style, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_line_width(&style, 6);
    lv_style_set_line_rounded(&style, true);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_line_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);

    static lv_point_precise_t p[] = {{10, 30}, {30, 50}, {100, 0}};
    lv_line_set_points(obj, p, 3);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

    lv_obj_center(obj);
}

#endif

```

Transizione

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Creating a transition
 */
void lv_example_style_10(void)
{
    static const lv_style_prop_t props[] = {LV_STYLE_BG_COLOR, LV_STYLE_BORDER_COLOR, LV_STYLE_BORDER_WIDTH, 0};

    /* A default transition
     * Make it fast (100ms) and start with some delay (200 ms)*/
    static lv_style_transition_dsc_t trans_def;
    lv_style_transition_dsc_init(&trans_def, props, lv_anim_path_linear, 100, 200, NULL);

    /* A special transition when going to pressed state
     * Make it slow (500 ms) but start without delay*/
    static lv_style_transition_dsc_t trans_pr;
    lv_style_transition_dsc_init(&trans_pr, props, lv_anim_path_linear, 500, 0, NULL);

    static lv_style_t style_def;
    lv_style_init(&style_def);
    lv_style_set_transition(&style_def, &trans_def);

    static lv_style_t style_pr;
    lv_style_init(&style_pr);
    lv_style_set_bg_color(&style_pr, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_border_width(&style_pr, 6);
    lv_style_set_border_color(&style_pr, lv_palette_darken(LV_PALETTE_RED, 3));
    lv_style_set_transition(&style_pr, &trans_pr);

    /*Create an object with the new style_pr*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style_def, 0);
    lv_obj_add_style(obj, &style_pr, LV_STATE_PRESSED);

    lv_obj_center(obj);
}
#endif

```

Utilizzo di stili multipli

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Using multiple styles
 */
void lv_example_style_11(void)
{
    /*A base style*/
    static lv_style_t style_base;
    lv_style_init(&style_base);
    lv_style_set_bg_color(&style_base, lv_palette_main(LV_PALETTE_LIGHT_BLUE));
    lv_style_set_border_color(&style_base, lv_palette_darken(LV_PALETTE_LIGHT_BLUE, 3));
    lv_style_set_border_width(&style_base, 2);
    lv_style_set_radius(&style_base, 10);
    lv_style_set_shadow_width(&style_base, 10);
    lv_style_set_shadow_offset_y(&style_base, 5);
    lv_style_set_shadow_opa(&style_base, LV_OPA_50);
    lv_style_set_text_color(&style_base, lv_color_white());
    lv_style_set_width(&style_base, 100);
    lv_style_set_height(&style_base, LV_SIZE_CONTENT);

    /*Set only the properties that should be different*/
    static lv_style_t style_warning;
    lv_style_init(&style_warning);
    lv_style_set_bg_color(&style_warning, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_border_color(&style_warning, lv_palette_darken(LV_PALETTE_YELLOW, 3));
    lv_style_set_text_color(&style_warning, lv_palette_darken(LV_PALETTE_YELLOW, 4));

    /*Create an object with the base style only*/
    lv_obj_t * obj_base = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj_base, &style_base, 0);
    lv_obj_align(obj_base, LV_ALIGN_LEFT_MID, 20, 0);

    lv_obj_t * label = lv_label_create(obj_base);
    lv_label_set_text(label, "Base");
    lv_obj_center(label);

    /*Create another object with the base style and earnings style too*/
    lv_obj_t * obj_warning = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj_warning, &style_base, 0);
    lv_obj_add_style(obj_warning, &style_warning, 0);
    lv_obj_align(obj_warning, LV_ALIGN_RIGHT_MID, -20, 0);

    label = lv_label_create(obj_warning);
    lv_label_set_text(label, "Warning");
    lv_obj_center(label);
}

#endif

```

Stili Locali

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Local styles
 */
void lv_example_style_12(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_GREEN));
    lv_style_set_border_color(&style, lv_palette_lighten(LV_PALETTE_GREEN, 3));
    lv_style_set_border_width(&style, 3);

    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);

    /*Overwrite the background color locally*/
    lv_obj_set_style_bg_color(obj, lv_palette_main(LV_PALETTE_ORANGE), LV_PART_MAIN);

    lv_obj_center(obj);
}

#endif
```

Aggiunta di stili a parti e a stati

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Add styles to parts and states
 */
void lv_example_style_13(void)
{
    static lv_style_t style_indic;
    lv_style_init(&style_indic);
    lv_style_set_bg_color(&style_indic, lv_palette_lighten(LV_PALETTE_RED, 3));
    lv_style_set_bg_grad_color(&style_indic, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_bg_grad_dir(&style_indic, LV_GRAD_DIR_HOR);

    static lv_style_t style_indic_pr;
    lv_style_init(&style_indic_pr);
    lv_style_set_shadow_color(&style_indic_pr, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_shadow_width(&style_indic_pr, 10);
    lv_style_set_shadow_spread(&style_indic_pr, 3);

    /*Create an object with the new style_pr*/
    lv_obj_t * obj = lv_slider_create(lv_screen_active());
    lv_obj_add_style(obj, &style_indic, LV_PART_INDICATOR);
    lv_obj_add_style(obj, &style_indic_pr, LV_PART_INDICATOR | LV_STATE_PRESSED);
    lv_slider_set_value(obj, 70, LV_ANIM_OFF);
    lv_obj_center(obj);
}

#endif
```

Estensione del tema corrente

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE
#include "../lvgl_private.h"

static lv_style_t style_btn;

/*Will be called when the styles of the base theme are already added
to add new styles*/
static void new_theme_apply_cb(lv_theme_t * th, lv_obj_t * obj)
{
    LV_UNUSED(th);

    if(lv_obj_check_type(obj, &lv_button_class)) {
        lv_obj_add_style(obj, &style_btn, 0);
    }
}

static void new_theme_init_and_set(void)
{
    /*Initialize the styles*/
    lv_style_init(&style_btn);
    lv_style_set_bg_color(&style_btn, lv_palette_main(LV_PALETTE_GREEN));
    lv_style_set_border_color(&style_btn, lv_palette_darken(LV_PALETTE_GREEN, 3));
    lv_style_set_border_width(&style_btn, 3);

    /*Initialize the new theme from the current theme*/
    lv_theme_t * th_act = lv_display_get_theme(NULL);
    static lv_theme_t th_new;
    th_new = *th_act;

    /*Set the parent theme and the style apply callback for the new theme*/
    lv_theme_set_parent(&th_new, th_act);
    lv_theme_set_apply_cb(&th_new, new_theme_apply_cb);

    /*Assign the new theme to the current display*/
    lv_display_set_theme(NULL, &th_new);
}

/***
 * Extending the current theme
 */
void lv_example_style_14(void)
```

(continues on next page)

(continua dalla pagina precedente)

```
{
    lv_obj_t * btn;
    lv_obj_t * label;

    btn = lv_button_create(lv_screen_active());
    lv_obj_align(btn, LV_ALIGN_TOP_MID, 0, 20);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Original theme");

    new_theme_init_and_set();

    btn = lv_button_create(lv_screen_active());
    lv_obj_align(btn, LV_ALIGN_BOTTOM_MID, 0, -20);

    label = lv_label_create(btn);
    lv_label_set_text(label, "New theme");
}

#endif
```

Opacità e trasformazioni

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_BUTTON && LV_USE_LABEL

/***
 * Opacity and Transformations
 */
void lv_example_style_15(void)
{
    lv_obj_t * btn;
    lv_obj_t * label;

    /*Normal button*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 40);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, -70);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Normal");
    lv_obj_center(label);

    /*Set opacity
    *The button and the label is rendered to a layer first and that layer is blended*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 40);
    lv_obj_set_style_opa(btn, LV_OPA_50, 0);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, 0);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Opa:50%");
    lv_obj_center(label);

    /*Set transformations
    *The button and the label is rendered to a layer first and that layer is transformed*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 40);
    lv_obj_set_style_transform_rotation(btn, 150, 0);           /*15 deg*/
    lv_obj_set_style_transform_scale(btn, 256 + 64, 0);        /*1.25x*/
    lv_obj_set_style_transform_pivot_x(btn, 50, 0);
    lv_obj_set_style_transform_pivot_y(btn, 20, 0);
    lv_obj_set_style_opa(btn, LV_OPA_50, 0);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, 70);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Transf.");
    lv_obj_center(label);
}

#endif
```

Manopola metallica con gradiente conico

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

/***
 * Simulate metallic knob using conical gradient
 * For best effect set LV_GRADIENT_MAX_STOPS to 8 or at least 3
 */
void lv_example_style_16(void)
{
#if LV_GRADIENT_MAX_STOPS >= 8
    static const lv_color_t grad_colors[8] = {
        LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
        LV_COLOR_MAKE(0xff, 0xff, 0xff),
        LV_COLOR_MAKE(0xfa, 0xfa, 0xfa),
        LV_COLOR_MAKE(0x79, 0x79, 0x79),
        LV_COLOR_MAKE(0x4b, 0x4b, 0x4b),
        LV_COLOR_MAKE(0x4b, 0x4b, 0x4b),
        LV_COLOR_MAKE(0x70, 0x70, 0x70),
        LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
    };
#elif LV_GRADIENT_MAX_STOPS >= 3
    static const lv_color_t grad_colors[3] = {
        LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
        LV_COLOR_MAKE(0xff, 0xff, 0xff),
        LV_COLOR_MAKE(0x79, 0x79, 0x79),
    };
#else

```

(continues on next page)

(continua dalla pagina precedente)

```

static const lv_color_t grad_colors[2] = {
    LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
    LV_COLOR_MAKE(0x79, 0x79, 0x79),
};

/*Create a style with gradient background and shadow*/
static lv_style_t style;
lv_style_init(&style);
lv_style_set_radius(&style, 500);
lv_style_set_bg_opa(&style, LV_OPA_COVER);
lv_style_set_shadow_color(&style, lv_color_black());
lv_style_set_shadow_width(&style, 50);
lv_style_set_shadow_offset_x(&style, 20);
lv_style_set_shadow_offset_y(&style, 20);
lv_style_set_shadow_opa(&style, LV_OPA_50);

/*First define a color gradient. In this example we use a gray color map with random values.*/
static lv_grad_desc_t grad;

lv_grad_init_stops(&grad, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));

/*Make a conical gradient with the center in the middle of the object*/
#if LV_GRADIENT_MAX_STOPS >= 8
    lv_grad_conical_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, 0, 120, LV_GRAD_EXTEND_REFLECT);
#elif LV_GRADIENT_MAX_STOPS >= 3
    lv_grad_conical_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, 45, 125, LV_GRAD_EXTEND_REFLECT);
#else
    lv_grad_conical_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, 45, 110, LV_GRAD_EXTEND_REFLECT);
#endif

/*Set gradient as background*/
lv_style_set_bg_grad(&style, &grad);

/*Create an object with the new style*/
lv_obj_t * obj = lv_obj_create(lv_screen_active());
lv_obj_add_style(obj, &style, 0);
lv_obj_set_size(obj, 200, 200);
lv_obj_center(obj);
}

#else

void lv_example_style_16(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, LV_PCT(80));
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS is not enabled");
    lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif /*LV_BUILD_EXAMPLES*/

```

Gradiente radiale come sfondo

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

/***
 * Using radial gradient as background
 */
void lv_example_style_17(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0x9B, 0x18, 0x42),
        LV_COLOR_MAKE(0x00, 0x00, 0x00),
    };

    int32_t width = lv_display_get_horizontal_resolution(NULL);
    int32_t height = lv_display_get_vertical_resolution(NULL);

    static lv_style_t style;
    lv_style_init(&style);

    /*First define a color gradient. In this example we use a purple to black color map.*/
    static lv_grad_desc_t grad;

    lv_grad_init_stops(&grad, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));

    /*Make a radial gradient with the center in the middle of the object, extending to the farthest corner*/
    lv_grad_radial_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, LV_GRAD_RIGHT, LV_GRAD_BOTTOM, LV_GRAD_EXTEND_PAD);

    /*Set gradient as background*/
    lv_style_set_bg_grad(&style, &grad);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_set_size(obj, width, height);
    lv_obj_center(obj);
}

#else

void lv_example_style_17(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, LV_PCT(80));
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS is not enabled");
    lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
    lv_obj_center(label);
}

```

(continues on next page)

(continua dalla pagina precedente)

```
#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif /*LV_BUILD_EXAMPLES*/
```

Gradienti per lo sfondo del pulsante

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

/***
 * Using various gradients for button background
 */
void lv_example_style_18(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0x26, 0xa0, 0xda),
        LV_COLOR_MAKE(0x31, 0x47, 0x55),
    };

    /*Create a linear gradient going from the top left corner to the bottom at an angle, with reflected color map*/
    static lv_style_t style_with_linear_gradient_bg;
    static lv_grad_dsc_t linear_gradient_dsc;      /*NOTE: the gradient descriptor must be static or global variable!*/

    lv_style_init(&style_with_linear_gradient_bg);
    lv_grad_init_stops(&linear_gradient_dsc, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));
    lv_grad_linear_init(&linear_gradient_dsc, lv_pct(0), lv_pct(0), lv_pct(20), lv_pct(100), LV_GRAD_EXTEND_REFLECT);
    lv_style_set_bg_grad(&style_with_linear_gradient_bg, &linear_gradient_dsc);
    lv_style_set_bg_opa(&style_with_linear_gradient_bg, LV_OPA_COVER);

    /*Create a radial gradient with the center in the top left 1/3rd of the object, extending to the bottom right corner, with reflected color map*/
    static lv_style_t style_with_radial_gradient_bg;
    static lv_grad_dsc_t radial_gradient_dsc;      /*NOTE: the gradient descriptor must be static or global variable!*/

    lv_style_init(&style_with_radial_gradient_bg);
    lv_grad_init_stops(&radial_gradient_dsc, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));
    lv_grad_radial_init(&radial_gradient_dsc, lv_pct(30), lv_pct(30), lv_pct(100), lv_pct(100), LV_GRAD_EXTEND_REFLECT);
    lv_style_set_bg_grad(&style_with_radial_gradient_bg, &radial_gradient_dsc);
    lv_style_set_bg_opa(&style_with_radial_gradient_bg, LV_OPA_COVER);

    /*Create buttons with different gradient styles*/

    lv_obj_t * btn;
    lv_obj_t * label;

    /*Simple horizontal gradient*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_style_bg_color(btn, grad_colors[0], 0);
    lv_obj_set_style_bg_grad_color(btn, grad_colors[1], 0);
    lv_obj_set_style_bg_grad_dir(btn, LV_GRAD_DIR_HOR, 0);
    lv_obj_set_size(btn, 150, 50);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, -100);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Horizontal");
    lv_obj_center(label);

    /*Simple vertical gradient*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_style_bg_color(btn, grad_colors[0], 0);
    lv_obj_set_style_bg_grad_color(btn, grad_colors[1], 0);
    lv_obj_set_style_bg_grad_dir(btn, LV_GRAD_DIR_VER, 0);
    lv_obj_set_size(btn, 150, 50);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, -40);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Vertical");
    lv_obj_center(label);

    /*Complex linear gradient*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_add_style(btn, &style_with_linear_gradient_bg, 0);
    lv_obj_set_size(btn, 150, 50);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, 20);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Linear");
    lv_obj_center(label);

    /*Complex radial gradient*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_add_style(btn, &style_with_radial_gradient_bg, 0);
    lv_obj_set_size(btn, 150, 50);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, 80);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Radial");
    lv_obj_center(label);
}

#else

void lv_example_style_18(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, LV_PCT(80));
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS is not enabled");
    lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif /*LV_BUILD_EXAMPLES*/
```

Test tra stile di ricolorazione o sfondo modale pieno

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SLIDER && LV_USE_LOG

/***
 * Test between a full background modal and a recolor modal
 */
void lv_example_style_19(void)
{
    /*Add lv_example_style_11 as background*/
    lv_example_style_11();

    /* Set to 1 to enable recolor overlay instead of solid background */
#if 0
    /* Apply a screen-wide tint using recolor (efficient overlay).
     * This modifies the visual appearance by blending a semi-transparent color
     * over existing content without creating additional objects.
     * It's lighter on performance compared to a full-size background object. */
    lv_obj_set_style_recolor(lv_screen_active(), lv_color_black(), 0);
    lv_obj_set_style_recolor_opa(lv_screen_active(), LV_OPA_50, 0);
#else
    /* Simulate a modal background by setting a semi-transparent black background
     * on lv_layer_top(), the highest built-in layer.
     * This method creates a new full-screen object and can consume more resources
     * compared to recolor, especially when using images or gradients. */
    lv_obj_set_style_bg_color(lv_layer_top(), lv_color_black(), 0);
    lv_obj_set_style_bg_opa(lv_layer_top(), LV_OPA_50, 0);
#endif

    lv_obj_t * obj = lv_slider_create(lv_layer_top());
    lv_obj_center(obj);

    lv_refr_now(NULL); /*Update layouts and render*/

    lv_obj_invalidate(lv_screen_active());

    uint32_t t = lv_tick_get();
    lv_refr_now(NULL); /*Render only*/
    LV_LOG_USER("%s LV_PRIu32 " ms"\n", lv_tick_elaps(t));
}

#endif
```

5.8.2 Fogli di Stile

Panoramica

Inizializzazione degli Stili

Gli stili vengono memorizzati nelle variabili `lv_style_t`. Le variabili di stile devono essere `static`, globali o allocate dinamicamente. In altre parole, non possono essere variabili locali in funzioni le quali non sono più valide dopo l'uscita dalla funzione. Prima di utilizzare uno stile, è necessario inizializzarlo con `lv_style_init(&my_style)`. Dopo aver inizializzato uno stile, è possibile aggiungere o modificare proprietà.

Le funzioni per settare le proprietà sono le seguenti: `lv_style_set_<property_name>(&style, <value>)`; Per esempio:

```
static lv_style_t style_btn;
lv_style_init(&style_btn);
lv_style_set_bg_color(&style_btn, lv_color_hex(0x115588));
lv_style_set_bg_opa(&style_btn, LV_OPA_50);
lv_style_set_border_width(&style_btn, 2);
lv_style_set_border_color(&style_btn, lv_color_black());

static lv_style_t style_btn_red;
lv_style_init(&style_btn_red);
lv_style_set_bg_color(&style_btn_red, lv_palette_main(LV_PALETTE_RED));
lv_style_set_bg_opa(&style_btn_red, LV_OPA_COVER);
```

Per rimuovere una proprietà, utilizzare:

```
lv_style_remove_prop(&style, LV_STYLE_BG_COLOR);
```

Per ottenere il valore di una proprietà da uno stile:

```
lv_style_value_t v;
lv_result_t res = lv_style_get_prop(&style, LV_STYLE_BG_COLOR, &v);
if(res == LV_RESULT_OK) { /* Found */
    do_something(v.color);
}
```

`lv_style_value_t` ha 3 campi, di cui solo uno verrà applicato, a seconda del tipo di proprietà a cui viene applicato:

- `num`: per le proprietà intere, booleane e di opacità
- `color`: per le proprietà relative al colore
- `ptr`: per le proprietà relative al puntatore

Per resettare uno stile (liberando tutti i suoi dati) utilizzare:

```
lv_style_reset(&style);
```

Se non è necessario modificare uno stile in fase di runtime, gli stili possono essere creati anche come variabili "const" per risparmiare RAM:

```
const lv_style_const_prop_t style1_props[] = {
    LV_STYLE_CONST_WIDTH(50),
    LV_STYLE_CONST_HEIGHT(50),
    LV_STYLE_CONST_PROPS_END
};

LV_STYLE_CONST_INIT(style1, style1_props);
```

In seguito, lo stile const potrà essere utilizzato come qualsiasi altro stile, ma (ovviamente) non sarà possibile aggiungere nuove proprietà.

Aggiungere Stili ai Widget

Uno stile di per sé non ha effetto finché non viene aggiunto (assegnato) a un Widget.

Per aggiungere uno stile a un Widget, utilizzare `lv_obj_add_style(widget, &style, <selector>)`. `<selector>` è un valore OR tra le parti e lo stato a cui aggiungere lo stile. Alcuni esempi:

- `LV_PART_MAIN | LV_STATE_DEFAULT`
- `LV_STATE_PRESSED`: La parte "main" [principale] in stato premuto. `LV_PART_MAIN` può essere omesso
- `LV_PART_SCROLLBAR`: La parte della barra di scorrimento nello stato di default. `LV_STATE_DEFAULT` può essere omesso.
- `LV_PART_SCROLLBAR | LV_STATE_SCROLLED`: La parte della barra di scorrimento quando si scorre il widget.
- `LV_PART_INDICATOR | LV_STATE_PRESSED | LV_STATE_CHECKED` La parte dell'indicatore quando il widget viene premuto e selezionato contemporaneamente.

Uso di `lv_obj_add_style()`:

```
lv_obj_add_style(btn, &style_btn, 0);           /* Default button style */
lv_obj_add_style(btn, &btn_red, LV_STATE_PRESSED); /* Overwrite only some colors to red when pressed */
```

Rimozione degli Stili

Per rimuovere tutti gli stili da un widget, utilizzare `lv_obj_remove_style_all(widget)`.

Per rimuovere stili specifici, si usa `lv_obj_remove_style(widget, &style, selector)`. Questa funzione rimuoverà `style` solo se il `selector` corrisponde al `selector` utilizzato in `lv_obj_add_style()`. L'argomento `style` può essere `NULL` per trovare solo `selector` e rimuovere tutti gli stili corrispondenti. Il `selector` può anche utilizzare i valori `LV_STATE_ANY` e `LV_PART_ANY` per rimuovere lo stile da qualsiasi stato o parte.

Sostituzione di Stili

Per sostituire uno stile specifico di un widget, si usa `lv_obj_replace_style(widget, old_style, new_style, selector)`. Questa funzione sostituirà `old_style` con `new_style` se il `selector` corrisponde al `selector` utilizzato in `lv_obj_add_style()`. Sia `old_style` che `new_style` non devono essere `NULL`. Esistono funzioni separate per aggiungere e rimuovere stili. Se la combinazione di `old_style` e `selector` esiste più volte negli stili di `obj`, tutte le occorrenze verranno sostituite. Il valore restituito dalla funzione indica se è avvenuta almeno una sostituzione corretta.

Uso di `lv_obj_replace_style()`:

```
lv_obj_add_style(btn, &style_btn, 0);           /* Add a button style */
lv_obj_replace_style(btn, &style_btn, &new_style_btn, 0); /* Replace the button style with a different one */
```

Segnalare le Modifiche di Stile

Se uno stile già assegnato a un widget cambia (ad esempio, viene aggiunta o modificata una proprietà), i Widget che utilizzano quello stile devono essere avvisati. Ci sono 3 opzioni per farlo:

1. Se si sa che le proprietà modificate possono essere applicate con un semplice ridisegno (ad esempio, modifiche di colore o opacità), si chiama semplicemente `lv_obj_invalidate(widget)` o `lv_obj_invalidate(lv_screen_active())`.
2. Se sono state modificate o aggiunte proprietà di stile più complesse e si sanno quali Widget sono interessati da quello stile, si chiama `lv_obj_refresh_style(widget, part, property)`. Per aggiornare tutte le parti e le proprietà, utilizzare `lv_obj_refresh_style(widget, LV_PART_ANY, LV_STYLE_PROP_ANY)`.
3. Per fare in modo che LVGL controlli tutti i Widget per vedere se utilizzano uno stile e aggiornarli quando necessario, chiamare `lv_obj_report_style_change(&style)`. Se `style` è NULL tutti i Widget riceveranno una notifica di modifica dello stile.

5.8.3 Stili Locali

Panoramica

Oltre agli stili "normali", i Widget possono anche memorizzare stili locali. Questo concetto è simile agli stili inline in CSS (ad esempio `<div style="color:red">`) con alcune modifiche.

Gli stili locali sono come gli stili normali, ma non possono essere condivisi tra altri Widget. Se utilizzati, gli stili locali vengono allocati automaticamente e liberati quando il Widget viene eliminato. Sono utili per aggiungere personalizzazioni locali a un widget.

A differenza di CSS, gli stili locali LVGL possono essere assegnati a stati (pseudo-classes) e parti (pseudo-elements).

Utilizzo

Per impostare una proprietà locale, utilizzare funzioni come `lv_obj_set_style_<property_name>(widget, <value>, <selector>)`; Per esempio:

```
lv_obj_set_style_bg_color(slider, lv_color_red(), LV_PART_INDICATOR | LV_STATE_FOCUSED);
```

5.8.4 Transizioni

Panoramica

Per default, quando un Widget cambia stato (ad esempio, quando viene premuto), le nuove proprietà del nuovo stato vengono impostate immediatamente. Tuttavia, con le transizioni è possibile riprodurre un'animazione al cambio di stato. Ad esempio, premendo un pulsante, il suo colore di sfondo può essere animato per 300 ms fino a raggiungere il colore del pulsante premuto.

I parametri delle transizioni sono memorizzati negli stili. È possibile impostare

- il tempo della transizione
- il ritardo prima dell'inizio della transizione
- il path dell'animazione (noto anche come funzione di temporizzazione o "easing" [allentamento])
- le proprietà da animare

Le proprietà della transizione possono essere definite per ogni stato. Ad esempio, impostando un tempo di transizione di 500 ms nello stato di default, quando il widget passa allo stato di default viene applicato un tempo di transizione di 500 ms. Impostando un tempo di transizione di 100 ms nello stato premuto, si ottiene una transizione di 100 ms quando si passa allo stato premuto. Questa configurazione di esempio comporta un rapido passaggio allo stato premuto e un lento ritorno allo stato di default.

Utilizzo

Per descrivere una transizione, è necessario inizializzare una variabile `lv_style_transition_dsc_t` e aggiungerla a uno stile:

```
/* Only its pointer is saved so must static, global or dynamically allocated */
static const lv_style_prop_t trans_props[] = {
    LV_STYLE_BG_OPA, LV_STYLE_BG_COLOR,
    0, /* End marker */
};

static lv_style_transition_dsc_t transl;
lv_style_transition_dsc_init(&transl, trans_props, lv_anim_path_ease_out, duration_ms, delay_ms);

lv_style_set_transition(&style1, &transl);
```

5.8.5 Temi

Panoramica

Un Tema è una raccolta di stili con la consapevolezza di quale sottoinsieme di stili viene applicato a quali tipi di Widget. Se è presente un tema attivo, LVGL lo applica a ogni widget appena creato. Questo darà un aspetto predefinito all'interfaccia utente, che potrà poi essere modificato aggiungendo ulteriori stili.

Ogni display può avere un tema diverso. Ad esempio, è possibile avere un tema colorato su un TFT e un tema monocromatico su un display monocromatico secondario.

Per impostare un tema per un display, sono necessari due passaggi:

1. Inizializzare il Tema
2. Assegnare il tema inizializzato al display.

Le funzioni di inizializzazione del tema possono avere prototipi diversi. Questo esempio mostra come impostare il tema di "default":

```
lv_theme_t * th = lv_theme_default_init(display,
    LV_COLOR_PALETTE_BLUE, /* Use DPI, size, etc. from this display */
    /* Primary and secondary palette */
    LV_COLOR_PALETTE_CYAN,
    false, /* Dark theme? False = light theme. */
    &lv_font_montserrat_10, /* Small, normal, large fonts */
    &lv_font_montserrat_14,
    &lv_font_montserrat_18);

lv_display_set_theme(display, th); /* Assign theme to display */
```

I temi inclusi sono abilitati in `lv_conf.h`. Se il tema di default è abilitato da `LV_USE_THEME_DEFAULT` LVGL lo inizializza e lo imposta automaticamente quando viene creato un display.

Estensione dei temi

I temi nativi possono essere estesi. Se viene creato un tema personalizzato, è possibile selezionare un tema genitore. Gli stili del tema genitore verranno aggiunti prima degli stili del tema personalizzato. È possibile concatenare un numero qualsiasi di temi in questo modo. Ad esempio, tema di default -> tema personalizzato -> tema scuro.

`lv_theme_set_parent(new_theme, base_theme)` estende `base_theme` con `new_theme`.

Un esempio di questo è presente tra gli *esempi di Stile*.

5.8.6 Proprietà di Stile

Dimensioni e posizione

Proprietà relative a dimensioni, posizione, allineamento e layout dei Widget.

width

Imposta la larghezza del widget. È possibile utilizzare i valori in pixel, percentuale e `LV_SIZE_CONTENT`. I valori percentuali sono relativi alla larghezza dell'area del "content" dell'elemento padre.

min_width

Imposta una larghezza minima. È possibile utilizzare i valori in pixel e in percentuale. I valori percentuali sono relativi alla larghezza dell'area del "content" dell'elemento padre.

max_width

Imposta una larghezza massima. È possibile utilizzare i valori in pixel e in percentuale. I valori percentuali sono relativi alla larghezza dell'area del "content" dell'elemento padre.

height

Imposta l'altezza del Widget. È possibile utilizzare i valori in pixel, percentuale e `LV_SIZE_CONTENT`. I valori percentuali sono relativi all'altezza dell'area del "content" dell'elemento padre.

min_height

Imposta un'altezza minima. È possibile utilizzare i valori in pixel e in percentuale. I valori percentuali sono relativi alla larghezza dell'area del "content" dell'elemento padre.

max_height

Imposta un'altezza massima. È possibile utilizzare i valori in pixel e in percentuale. I valori percentuali sono relativi all'altezza dell'area del "content" dell'elemento padre.

length

Il suo significato dipende dal tipo di Widget. Ad esempio, nel caso di `lv_scale`, indica la lunghezza dei tick.

x

Imposta la coordinata X del Widget considerando l'impostazione `align`. È possibile utilizzare i valori in pixel e in percentuale. I valori percentuali sono relativi alla larghezza dell'area del "content" dell'elemento padre.

y

Imposta la coordinata Y del Widget considerando l'impostazione `align`. È possibile utilizzare i valori in pixel e in percentuale. I valori percentuali sono relativi all'altezza dell'area del "content" dell'elemento padre.

align

Imposta l'allineamento che indica da quale punto dell'oggetto padre devono essere interpretate le coordinate X e Y. I valori possibili sono: `LV_ALIGN_DEFAULT`, `LV_ALIGN_TOP_LEFT/MID/RIGHT`, `LV_ALIGN_BOTTOM_LEFT/MID/RIGHT`, `LV_ALIGN_LEFT/RIGHT_MID`, `LV_ALIGN_CENTER`. `LV_ALIGN_DEFAULT` significa `LV_ALIGN_TOP_LEFT` con direzione di base LTR e `LV_ALIGN_TOP_RIGHT` con direzione di base RTL.

transform_width

Questo valore allarga il Widget su entrambi i lati. È possibile utilizzare valori in pixel e percentuali (con `lv_pct(x)`). Sposta il widget con questo valore nella direzione Y.

transform_height

Questo valore alza il Widget su entrambi i lati. È possibile utilizzare valori in pixel e percentuali (con `lv_pct(x)`). I valori percentuali sono relativi all'altezza del Widget.

translate_x

Sposta il Widget con questo valore nella direzione X. Applicato dopo layout, allineamenti e altri posizionamenti. È possibile utilizzare valori in pixel e percentuali (con `lv_pct(x)`). Sposta il widget con questo valore nella direzione Y.

translate_y

Sposta il Widget con questo valore nella direzione Y. Applicato dopo layout, allineamenti e altri posizionamenti. È possibile utilizzare valori in pixel e percentuali (con `lv_pct(x)`). I valori percentuali sono relativi all'altezza del Widget.

translate_radial

Sposta l'oggetto attorno al centro dell'oggetto padre (ad esempio attorno alla circonferenza di una scala)

transform_scale_x

Zoom orizzontale del Widget. Il valore 256 (o `LV_SCALE_NONE`) indica dimensioni normali, 128 dimensioni dimezzate, 512 dimensioni doppie e così via.

transform_scale_y

Zoom verticale del Widget. Il valore 256 (o `LV_SCALE_NONE`) indica dimensioni normali, 128 dimensioni dimezzate, 512 dimensioni doppie e così via.

transform_rotation

Ruota il Widget. Il valore viene interpretato in unità di 0,1 gradi. Ad esempio, 450 significa 45 gradi.

transform_pivot_x

Imposta la coordinata X del punto pivot per le trasformazioni. Relativamente all'angolo in alto a sinistra del Widget.

transform_pivot_y

Imposta la coordinata Y del punto pivot per le trasformazioni. Relativamente all'angolo in alto a sinistra del Widget.

transform_skew_x

Inclina il Widget orizzontalmente. Il valore viene interpretato in unità di 0,1 gradi. Ad esempio, 450 significa 45 gradi.

transform_skew_y

Inclina il Widget verticalmente. Il valore viene interpretato in unità di 0,1 gradi. Ad esempio, 450 significa 45 gradi.

Padding [riempimento]

Proprietà che descrivono la spaziatura tra i lati del genitore e quelli dei figli e tra i figli. Molto simile alle proprietà di padding in HTML.

pad_top

Imposta il padding in alto. Riduce l'area del contenuto in questa direzione.

pad_bottom

Imposta il padding in basso. Riduce l'area del contenuto in questa direzione.

pad_left

Imposta il padding a sinistra. Riduce l'area del contenuto in questa direzione.

pad_right

Imposta il padding a destra. Riduce l'area del contenuto in questa direzione.

pad_row

Imposta il padding tra le righe. Utilizzato dai layout.

pad_column

Imposta il padding tra le colonne. Utilizzato dai layout.

pad_radial

Allontana le etichette di testo dai segni di graduazione/dal resto di LV_PART_

Margin

Proprietà per descrivere la spaziatura attorno a un Widget. Molto simile alle proprietà margin in HTML.

margin_top

Imposta il margine in alto. Il Widget manterrà questo spazio rispetto ai suoi "fratelli" nei layout.

margin_bottom

Imposta il margine in basso. Il Widget manterrà questo spazio rispetto ai suoi "fratelli" nei layout.

margin_left

Imposta il margine a sinistra. Il Widget manterrà questo spazio rispetto ai suoi "fratelli" nei layout.

margin_right

Imposta il margine a destra. Il Widget manterrà questo spazio rispetto ai suoi "fratelli" nei layout.

Background

Proprietà per descrivere il colore di sfondo e l'immagine del Widget.

bg_color

Imposta il colore di sfondo del Widget.

bg_opa

Imposta l'opacità dello sfondo. Il valore 0, LV_OPA_0 o LV_OPA_TRANSP significa completamente trasparente, 255, LV_OPA_100 o LV_OPA_COVER significa completamente coprente, altri valori o LV_OPA_10, LV_OPA_20, ecc. indicano semi-trasparenza.

bg_grad_color

Imposta il colore del gradiente dello sfondo. Utilizzato solo se `grad_dir` non è `LV_GRAD_DIR_NONE`

bg_grad_dir

Imposta la direzione del gradiente dello sfondo. I valori possibili sono `LV_GRAD_DIR_NONE/HOR/VER`.

bg_main_stop

Imposta il punto da cui deve iniziare il colore di sfondo per i gradienti. 0 indica il lato superiore/sinistro, 255 il lato inferiore/destro, 128 il centro e così via.

bg_grad_stop

Imposta il punto da cui deve iniziare il colore del gradiente dello sfondo. 0 indica il lato superiore/sinistro, 255 il lato inferiore/destro, 128 il centro e così via.

bg_main_opa

Imposta l'opacità del primo colore del gradiente.

bg_grad_opa

Imposta l'opacità del secondo colore del gradiente.

bg_grad

Imposta la definizione del gradiente. L'istanza puntata deve esistere mentre il Widget è attivo. `NULL` per disabilitare. Include `BG_GRAD_COLOR`, `BG_GRAD_DIR`, `BG_MAIN_STOP` e `BG_GRAD_STOP` in un unico descrittore e consente di creare gradienti anche con più colori. Se impostato, le altre proprietà relative al gradiente verranno ignorate.

bg_image_src

Imposta un'immagine di sfondo. Può essere un puntatore a `lv_image_dsc_t`, un path a un file o un `LV_SYMBOL_...`

bg_image_opa

Imposta l'opacità dell'immagine di sfondo. Il valore 0, `LV_OPA_0` o `LV_OPA_TRANSP` significa completamente trasparente, 255, `LV_OPA_100` o `LV_OPA_COVER` significa completamente coprente, altri valori o `LV_OPA_10`, `LV_OPA_20`, ecc. indicano semi-trasparenza.

bg_image_recolor

Imposta un colore da miscelare con l'immagine di sfondo.

bg_image_recolor_opa

Imposta l'intensità della ricolorazione dell'immagine di sfondo. Il valore 0, `LV_OPA_0` o `LV_OPA_TRANSP` significa nessuna miscelazione, 255, `LV_OPA_100` o `LV_OPA_COVER` significa ricolorazione completa, altri valori o `LV_OPA_10`, `LV_OPA_20`, ecc. vengono interpretati proporzionalmente.

bg_image_tiled

Se abilitato, l'immagine di sfondo verrà ricolorata. I valori possibili sono `true` o `false`.

Border [bordatura]

Proprietà per descrivere i bordi

border_color

Imposta il colore del bordo

border_opa

Imposta l'opacità del bordo. Il valore 0, LV_OPA_0 o LV_OPA_TRANSP significa completamente trasparente, 255, LV_OPA_100 o LV_OPA_COVER significa completamente coprente, altri valori o LV_OPA_10, LV_OPA_20, ecc. indicano semi-trasparenza.

border_width

Imposta lo spessore del bordo. Possono essere utilizzati solo valori in pixel.

border_side

Imposta solo il lato o i lati su cui deve essere disegnato il bordo. I valori possibili sono LV_BORDER_SIDE_NONE/TOP/BOTTOM/LEFT/RIGHT/INTERNAL. È possibile utilizzare anche valori OR, ad esempio LV_BORDER_SIDE_TOP | LV_BORDER_SIDE_LEFT.

border_post

Indica se il bordo deve essere disegnato prima o dopo i figli. `true`: dopo i figli, `false`: prima dei figli

Outline [contorno]

Proprietà per descrivere il contorno. È come un bordo, ma disegnato all'esterno dei rettangoli.

outline_width

Imposta la larghezza del contorno in pixel.

outline_color

Imposta il colore del contorno.

outline_opa

Imposta l'opacità del contorno. Il valore 0, LV_OPA_0 o LV_OPA_TRANSP significa completamente trasparente, 255, LV_OPA_100 o LV_OPA_COVER significa completamente coprente, altri valori o LV_OPA_10, LV_OPA_20, ecc. indicano semi-trasparenza.

outline_pad

Imposta il riempimento del contorno, ovvero lo spazio tra il Widget e il contorno.

Shadow

Proprietà per descrivere l'ombra disegnata sotto i rettangoli.

shadow_width

Imposta la larghezza dell'ombra in pixel. Il valore deve essere ≥ 0 .

shadow_offset_x

Imposta un offset dell'ombra in pixel nella direzione X.

shadow_offset_y

Imposta un offset dell'ombra in pixel nella direzione Y.

shadow_spread

Esegue il calcolo dell'ombra per utilizzare un rettangolo più grande o più piccolo come base. Il valore può essere in pixel per ingrandire/ridurre l'area.

shadow_color

Imposta il colore dell'ombra

shadow_opa

Imposta l'opacità dell'ombra. Il valore 0, LV_OPA_0 o LV_OPA_TRANSP significa completamente trasparente, 255, LV_OPA_100 o LV_OPA_COVER significa completamente coprente, altri valori o LV_OPA_10, LV_OPA_20, ecc. indicano semi-trasparenza.

Immagine

Proprietà per descrivere le immagini

image_opa

Imposta l'opacità di un'immagine. Il valore 0, LV_OPA_0 o LV_OPA_TRANSP significa completamente trasparente, 255, LV_OPA_100 o LV_OPA_COVER significa completamente coprente, altri valori o LV_OPA_10, LV_OPA_20, ecc. indicano semi-trasparenza.

image_recolor

Imposta il colore da mescolare con l'immagine.

image_recolor_opa

Imposta l'intensità della miscelazione dei colori. Il valore 0, LV_OPA_0 o LV_OPA_TRANSP significa completamente trasparente, 255, LV_OPA_100 o LV_OPA_COVER significa completamente coprente, altri valori o LV_OPA_10, LV_OPA_20, ecc. indicano semi-trasparenza.

image_colorkey

Impostare la definizione del colorkey dell'immagine. lv_image_colorkey_t contiene due valori di colore: `high_color` e `low_color`. il colore dei pixel che vanno da `low_color` a `high_color` sarà trasparente.

Line

Proprietà per descrivere i Widget lineari

line_width

Imposta la larghezza delle linee in pixel.

line_dash_width

Imposta la larghezza dei trattini in pixel. Nota: il trattino funziona solo su linee orizzontali e verticali.

line_dash_gap

Imposta la distanza tra i trattini in pixel. Nota: il trattino funziona solo su linee orizzontali e verticali.

line_rounded

Arrotonda le estremità delle linee. `true`: arrotondata, `false`: fine linea perpendicolare.

line_color

Imposta il colore delle linee.

line_opa

Imposta l'opacità delle linee.

Arco

TODO [da fare]

arc_width

Imposta la larghezza (spessore) degli archi in pixel.

arc_rounded

Arrotonda le estremità degli archi. `true`: arrotondata, `false`: fine linea perpendicolare.

arc_color

Imposta il colore dell'arco.

arc_opa

Imposta l'opacità degli archi.

arc_image_src

Imposta un'immagine da cui verrà mascherato l'arco. È utile per visualizzare effetti complessi sugli archi. Può essere un puntatore a `lv_image_dsc_t` o un percorso a un file.

Text

Proprietà per descrivere le proprietà del testo. Tutte queste proprietà sono ereditate.

text_color

Imposta il colore del testo.

text_opa

Imposta l'opacità del testo. Il valore 0, LV_OPA_0 o LV_OPA_TRANSP significa completamente trasparente, 255, LV_OPA_100 o LV_OPA_COVER significa completamente coprente, altri valori o LV_OPA_10, LV_OPA_20, ecc. indicano semi-trasparenza.

text_font

Imposta il font del testo (un puntatore `lv_font_t *`).

text_letter_space

Imposta la spaziatura delle lettere in pixel

text_line_space

Imposta la spaziatura delle righe in pixel.

text_decor

Imposta la decorazione del testo. I valori possibili sono LV_TEXT_DECOR_NONE/UNDERLINE/STRIKETHROUGH. È possibile utilizzare anche valori con OR.

text_align

Imposta come allineare le righe del testo. Nota: non allinea il Widget stesso, ma solo le righe al suo interno. I valori possibili sono LV_TEXT_ALIGN_LEFT/CENTER/RIGHT/AUTO. LV_TEXT_ALIGN_AUTO rileva la direzione di base del testo e utilizza l'allineamento a sinistra o a destra di conseguenza.

text_outline_stroke_color

Imposta il colore del contorno delle lettere.

text_outline_stroke_width

Imposta la larghezza del tratto del contorno della lettera in pixel.

text_outline_stroke_opa

Imposta l'opacità del tratto del contorno della lettera. Il valore 0, LV_OPA_0 o LV_OPA_TRANSP significa completamente trasparente, 255, LV_OPA_100 o LV_OPA_COVER significa completamente coprente, altri valori o LV_OPA_10, LV_OPA_20, ecc. indicano semi-trasparenza.

Varie

Proprietà miste per vari scopi.

raggio

Imposta il raggio su ogni angolo. Il valore viene interpretato in pixel (≥ 0) o LV_RADIUS_CIRCLE per il massimo del raggio

radial_offset

Sposta il punto iniziale dell'oggetto (ad esempio, il segno di ridimensionamento) radialmente.

clip_corner

Abilita il clipping del contenuto che eccede gli angoli arrotondati del Widget padre. Può essere `true` o `false`.

opa

Riduce tutti i valori di opacità del Widget di questo fattore. Il valore 0, `LV_OPA_0` o `LV_OPA_TRANSP` significa completamente trasparente, 255, `LV_OPA_100` o `LV_OPA_COVER` significa completamente coprente, altri valori o `LV_OPA_10`, `LV_OPA_20`, ecc. indicano semi-trasparenza.

opa_layered

Per prima cosa disegna il Widget sul layer, poi riduce il fattore di opacità del layer. Il valore 0, `LV_OPA_0` o `LV_OPA_TRANSP` significa completamente trasparente, 255, `LV_OPA_100` o `LV_OPA_COVER` significa completamente coprente, altri valori o `LV_OPA_10`, `LV_OPA_20`, ecc. indicano semi-trasparenza.

color_filter_dsc

Mescola un colore con tutti i colori del Widget.

color_filter_opa

Intensità di miscelazione del filtro colore.

recolor

Imposta un colore da mescolare con obj.

recolor_opa

Imposta l'intensità della miscelazione dei colori. Il valore 0, `LV_OPA_0` o `LV_OPA_TRANSP` significa completamente trasparente. Un valore di 255, `LV_OPA_100` o `LV_OPA_COVER` significa completamente opaco. Valori intermedi come `LV_OPA_10`, `LV_OPA_20`, ecc. generano una semi-trasparenza.

anim

Template per l'animazione del Widget. Dovrebbe essere un puntatore a `lv_anim_t`. I parametri di animazione sono specifici del widget, il tempo di animazione potrebbe essere, ad esempio, il tempo di lampeggio del cursore sull'area di testo o il tempo di scorrimento di un roller. Consultare la documentazione dei Widget per saperne di più.

anim_duration

Durata dell'animazione in millisecondi. Il suo significato è specifico del widget. Ad esempio, il tempo di lampeggio del cursore sulla Text Area o il tempo di scorrimento di un roller. Consultare la documentazione dei Widget per saperne di più.

transition

Un `lv_style_transition_dsc_t` inizializzato per descrivere una transizione.

blend_mode

Describe come sfumare i colori sullo sfondo. I valori possibili sono `LV_BLEND_MODE_NORMAL/ADDITIVE/SUBTRACTIVE/MULTIPLY/DIFFERENCE`

layout

Imposta il layout del Widget. I figli verranno riposizionati e ridimensionati in base alle policy impostate per il layout. Per i valori possibili, consultare la documentazione dei layout.

base_dir

Imposta la direzione di base del Widget. I valori possibili sono LV_BIDI_DIR_LTR/RTL/AUTO.

bitmap_mask_src

Se impostato, verrà creato un layer per il widget e il layer verrà mascherato con questa maschera bitmap A8.

rotary_sensitivity

Regola la sensibilità per i codificatori rotativi in unità 1/256. Con significato: 128: rallenta la rotazione della metà, 512: raddoppia la velocità, 256: nessuna modifica

Flex

Proprietà del layout Flex.

flex_flow

Definisce in quale direzione il layout Flex deve disporre i figli.

flex_main_place

Definisce come allineare i figli nella direzione del flusso del Flex.

flex_cross_place

Definisce come allineare i figli perpendicolarmente alla direzione del flusso del Flex.

flex_track_place

Definisce come allineare le tracce del flusso.

flex_grow

Definisce quanto spazio occupare proporzionalmente dallo spazio libero della traccia del Widget.

Grid

Proprietà del layout della griglia.

grid_column_dsc_array

Un array per descrivere le colonne della griglia. Dovrebbe essere terminato con LV_GRID_TEMPLATE_LAST

grid_column_align

Definisce come distribuire le colonne.

grid_row_dsc_array

Un array per descrivere le righe della griglia. Dovrebbe essere terminato con LV_GRID_TEMPLATE_LAST

grid_row_align

Definisce come distribuire le righe.

grid_cell_column_pos

Imposta la colonna in cui posizionare il Widget.

grid_cell_x_align

Imposta come allineare il Widget orizzontalmente.

grid_cell_column_span

Imposta il numero di colonne che il Widget deve occupare. Deve essere ≥ 1 .

grid_cell_row_pos

Imposta la riga in cui posizionare il widget.

grid_cell_y_align

Imposta come allineare il Widget verticalmente.

grid_cell_row_span

Imposta il numero di righe che il Widget deve occupare. Deve essere ≥ 1 .

5.9 Eventi

Gli eventi vengono attivati in LVGL quando accade qualcosa che potrebbe essere interessante per l'utente, ad esempio quando un Widget:

- viene cliccato
- viene fatto scorrere
- ne viene modificato un suo stesso valore
- viene ridisegnato, ecc.

Oltre ai widget, è possibile registrare eventi anche per display e dispositivi di input. Non viene spiegato in dettaglio di seguito, ma è possibile farlo modificando il prefisso delle funzioni da `lv_obj_` a `lv_display_` o `lv_indev_`.

5.9.1 Aggiunta di Eventi a un Widget

L'utente può assegnare funzioni di callback a un widget per elaborare gli eventi. In pratica, si presenta così:

```
lv_obj_t * btn = lv_button_create(lv_screen_active());
lv_obj_add_event_cb(btn, my_event_cb, LV_EVENT_CLICKED, user_data); /* Assign an event callback */

...
static void my_event_cb(lv_event_t * event)
{
    printf("Clicked\n");
}
```

Nell'esempio `LV_EVENT_CLICKED` significa che solo l'evento clic chiamerà `my_event_cb`. Vedere la [lista dei codici evento](#) per tutte le opzioni. `LV_EVENT_ALL` può essere utilizzato per ricevere tutti gli eventi.

L'ultimo parametro di `lv_obj_add_event_cb()` è un puntatore a eventuali dati personalizzati che saranno disponibili nell'evento. È possibile passare NULL per questo argomento se non è necessario utilizzare tali dati durante l'elaborazione dell'evento. È possibile recuperare il puntatore passato impostando la funzione di callback in questo modo:

```
my_user_data_t * user_data;
...
user_data = lv_event_get_user_data(e);
```

È possibile aggiungere altri eventi a un Widget, in questo modo:

```
lv_obj_add_event_cb(widget, my_event_cb_1, LV_EVENT_CLICKED, NULL);
lv_obj_add_event_cb(widget, my_event_cb_2, LV_EVENT_PRESSED, NULL);
lv_obj_add_event_cb(widget, my_event_cb_3, LV_EVENT_ALL, NULL); /* No filtering, receive all events */
```

Anche lo stesso callback di evento può essere utilizzato su un Widget con `user_data` diversi. Ad esempio:

```
lv_obj_add_event_cb(widget, increment_on_click, LV_EVENT_CLICKED, &num1);
lv_obj_add_event_cb(widget, increment_on_click, LV_EVENT_CLICKED, &num2);
```

Gli eventi verranno richiamati nell'ordine in cui sono stati aggiunti.

Altri Widget possono utilizzare la stessa *callback dell'evento*.

Allo stesso modo, gli eventi possono essere associati a dispositivi di input e display in questo modo:

```
lv_display_add_event_cb(disp, event_cb, LV_EVENT_RESOLUTION_CHANGED, NULL);
lv_indev_add_event_cb(indev, event_cb, LV_EVENT_CLICKED, NULL);
```

5.9.2 Rimozione di Eventi dai Widget

```
uint32_t i;
uint32_t event_cnt = lv_obj_get_event_count(widget);
for(i = 0; i < event_cnt; i++) {
    lv_event_dsc_t * event_dsc = lv_obj_get_event_dsc(widget, i);
    if(lv_event_dsc_get_cb(event_dsc) == some_event_cb) {
        lv_obj_remove_event(widget, i);
        break;
    }
}
```

5.9.3 Codici Evento

I codici evento possono essere raggruppati in queste categorie: - Eventi dei dispositivi di input - Eventi di disegno - Altri eventi - Eventi speciali - Eventi personalizzati

Tutti i Widget (come Pulsanti/Label/Slider ecc.), indipendentemente dal tipo, ricevono gli eventi *Input device*, *Drawing* e *Altri*.

Tuttavia, gli *Special event* sono specifici di un particolare tipo di widget. Consultare la [documentazione dei widget](#) per sapere quando vengono inviati.

Gli *eventi personalizzati* vengono aggiunti dall'utente e non vengono mai inviati da LVGL.

Esistono i seguenti codici evento:

Eventi del Dispositivo di Input

- `LV_EVENT_PRESSED`: Il Widget è stato premuto
- `LV_EVENT_PRESSING`: Il Widget è premuto (chiamato continuamente durante la pressione)
- `LV_EVENT_PRESS_LOST`: Il Widget è ancora premuto, ma il cursore/dito è scivolato fuori dal Widget
- `LV_EVENT_SHORT_CLICKED`: Il Widget è stato premuto per un breve periodo di tempo e poi rilasciato senza scorrere.
- `LV_EVENT_SINGLE_CLICKED`: Il Widget è stato premuto per un breve periodo di tempo e poi rilasciato senza scorrere, per la prima volta in una serie di clic. Una serie di clic si riferisce a più clic brevi in un breve periodo di tempo e a una distanza ridotta.

- *LV_EVENT_DOUBLE_CLICKED*: Il Widget è stato premuto per un breve periodo di tempo e poi rilasciato senza scorrere, per la seconda volta in una serie di clic.
- *LV_EVENT_TRIPLE_CLICKED*: Il Widget è stato premuto per un breve periodo di tempo e poi rilasciato senza scorrere, per la terza volta in una serie di clic.
- *LV_EVENT_LONG_PRESSED*: Il Widget è stato premuto per almeno `long_press_time`. Non richiamato se si scorre.
- *LV_EVENT_LONG_PRESSED_REPEAT*: Richiamato dopo `long_press_time` ogni `long_press_repeat_time` ms. Non richiamato se si scorre.
- *LV_EVENT_CLICKED*: Richiamato al rilascio se non si "scolla" (indipendentemente dalla pressione prolungata)
- *LV_EVENT_RELEASED*: Richiamato ogni volta che il Widget viene rilasciato
- *LV_EVENT_SCROLL_BEGIN*: Inizia lo Scrolling. Il parametro "event" è un puntatore all'animazione dello "scroll". Può essere modificato
- *LV_EVENT_SCROLL_THROW_BEGIN*:
- *LV_EVENT_SCROLL_END*: Termina lo "scrolling"
- *LV_EVENT_SCROLL*: Scrolling
- *LV_EVENT_GESTURE*: È stato rilevato un gesto. Il gesto si ottiene con `lv_indev_get_gesture_dir(lv_indev_active())`
- *LV_EVENT_KEY*: Un tasto viene inviato al Widget. Si ottiene il tasto con `lv_indev_get_key(lv_indev_active())`
- *LV_EVENT_FOCUSED*: Il Widget ha ricevuto il focus
- *LV_EVENT_DFOCUSSED*: Il Widget ha perso il focus
- *LV_EVENT_LEAVE*: Il Widget ha perso il focus ma è ancora selezionato
- *LV_EVENT_HIT_TEST*: Esegue un hit-test avanzato
- *LV_EVENT_INDEV_RESET*: Indev [Input device] è stato resettato
- *LV_EVENT_HOVER_OVER*: Passaggio dell'Indev sul Widget
- *LV_EVENT_HOVER_LEAVE*: L'Indev lascia il Widget

Eventi di Disegno

- *LV_EVENT_COVER_CHECK*: Verifica se il Widget copre completamente un'area. Il parametro evento è `lv_cover_check_info_t` *.
- *LV_EVENT_REFR_EXT_DRAW_SIZE*: Ottiene l'area di disegno aggiuntiva richiesta attorno al Widget (ad esempio per l'ombra). Il parametro evento è `int32_t` * per memorizzare le dimensioni.
- *LV_EVENT_DRAW_MAIN_BEGIN*: Avvio della fase di disegno principale
- *LV_EVENT_DRAW_MAIN*: Esecuzione del disegno principale
- *LV_EVENT_DRAW_MAIN_END*: Termine della fase di disegno principale
- *LV_EVENT_DRAW_POST_BEGIN*: Avvio della fase di post-disegno (quando tutti i figli sono stati disegnati)
- *LV_EVENT_DRAW_POST*: Esecuzione della fase di post-disegno (quando tutti i figli sono stati disegnati)
- *LV_EVENT_DRAW_POST_END*: Termine della fase di post-disegno (quando tutti i figli sono stati disegnati)
- *LV_EVENT_DRAW_TASK_ADDED*: Aggiunta di un task di disegno

Attenzione

Nelle funzioni di callback degli eventi di disegno, la sequenza di rendering è già iniziata e, durante questo periodo, è vietato apportare modifiche agli attributi o agli stili di qualsiasi widget o creare/eliminare widget. Se si tenta di farlo, LVGL genererà un errore di asserzione con un messaggio che indica che l'invalidazione di un'area non è consentita durante il rendering.

Eventi Speciali

- *LV_EVENT_VALUE_CHANGED*: Il valore del Widget è cambiato (ad esempio, il cursore è stato spostato)
- *LV_EVENT_INSERT*: Un testo è stato inserito nel Widget. I dati dell'evento sono `char *` in fase di inserimento.
- *LV_EVENT_REFRESH*: Notifica al Widget di aggiornare qualcosa (per l'utente)
- *LV_EVENT_READY*: Un processo è terminato
- *LV_EVENT_CANCEL*: Un processo è stato annullato

Altri Eventi

- *LV_EVENT_CREATE*: Il Widget è in fase di creazione
- *LV_EVENT_DELETE*: Il Widget è in fase di eliminazione
- *LV_EVENT_CHILD_CHANGED*: Il figlio è stato rimosso, aggiunto o le sue dimensioni o posizione sono state modificate.
- *LV_EVENT_CHILD_CREATED*: Il figlio è stato creato, viene sempre inoltrato a tutti i genitori.
- *LV_EVENT_CHILD_DELETED*: Il figlio è stato eliminato, viene sempre inoltrato a tutti i genitori.
- *LV_EVENT_SCREEN_UNLOAD_START*: È stato avviata la dismissione di una schermata, attivata immediatamente alla chiamata di `scr_load`.
- *LV_EVENT_SCREEN_LOAD_START*: È stato avviato il caricamento di una schermata, attivato allo scadere del ritardo per il cambio schermata.
- *LV_EVENT_SCREEN_LOADED*: È stata caricata una schermata.
- *LV_EVENT_SCREEN_UNLOADED*: È stata dismessa una schermata
- *LV_EVENT_SIZE_CHANGED*: Le coordinate/dimensioni del Widget sono cambiate
- *LV_EVENT_STYLE_CHANGED*: Lo stile del Widget è cambiato
- *LV_EVENT_LAYOUT_CHANGED*: La posizione degli elementi figlio è cambiata a causa di un ricalcolo del layout
- *LV_EVENT_GET_SELF_SIZE*: Ottiene la dimensione interna di un widget

Eventi del Display

- *LV_EVENT_INVALIDATE_AREA*: Un'area viene invalidata (contrassegnata per il ridisegno). `lv_event_get_param(e)` restituisce un puntatore a un oggetto `lv_area_t` con le coordinate dell'area da invalidare. L'area può essere modificata liberamente se necessario per adattarla a un requisito specifico del display. Di solito necessario con i display monocromatici per invalidare $N \times 8$ righe o colonne in un unico passaggio.
- *LV_EVENT_RESOLUTION_CHANGED*: Inviato quando la risoluzione cambia a causa di `lv_display_set_resolution()` o di `lv_display_set_rotation()`.
- *LV_EVENT_COLOR_FORMAT_CHANGED*: Inviato come risultato di qualsiasi chiamata a `lv_display_set_color_format()`.
- *LV_EVENT_REFR_REQUEST*: Inviato quando si è verificato un evento che richiede un ridisegno.
- *LV_EVENT_REFR_START*: Inviato prima dell'inizio di un ciclo di aggiornamento. Inviato anche se non c'è nulla da ridisegnare.

- *LV_EVENT_REFR_READY*: Inviato al completamento dell'aggiornamento (dopo il rendering e la chiamata a *Callback Flush*). Inviato anche se non si è verificato alcun ridisegno.
- *LV_EVENT_RENDER_START*: Inviato appena prima dell'inizio del rendering.
- *LV_EVENT_RENDER_READY*: Inviato dopo il completamento del rendering (prima di chiamare *Callback Flush*)
- *LV_EVENT_FLUSH_START*: Inviato prima della chiamata a *Callback Flush*.
- *LV_EVENT_FLUSH_FINISH*: Inviato dopo il ritorno della chiamata a *Callback Flush*.
- *LV_EVENT_FLUSH_WAIT_START*: Inviato all'inizio della chiamata interna a `wait_for_flushing()` -- si verifica indipendentemente dal fatto che si verifichi effettivamente un'attesa. La chiamata termina immediatamente se `disp->flushing == 0`.
- *LV_EVENT_FLUSH_WAIT_FINISH*: Inviato quando la chiamata a `wait_for_flushing()` sta per terminare, indipendentemente dal fatto che si sia verificata un'attesa effettiva.

Eventi Personalizzati

È possibile registrare un numero qualsiasi di codici evento personalizzati tramite `uint32_t MY_EVENT_1 = lv_event_register_id()`

Possono essere inviati a qualsiasi widget con `lv_obj_send_event(widget, MY_EVENT_1, &some_data)`

Evento del Refresh

LV_EVENT_REFRESH è un evento speciale perché è progettato per consentire all'utente di notificare a un widget di aggiornarsi. Alcuni esempi:

- notificare a un'etichetta di aggiornare il testo in base a una o più variabili (ad esempio, l'ora corrente)
- aggiornare un'etichetta quando cambia la lingua
- abilitare un pulsante se vengono soddisfatte determinate condizioni (ad esempio, viene inserito il PIN corretto)
- aggiungere/rimuovere stili da/a un widget se viene superato un limite, ecc.

5.9.4 Invio Manuale di Eventi

Per inviare manualmente eventi a un widget, utilizzare `lv_obj_send_event(widget, <EVENT_CODE>, &some_data)`.

Ad esempio, questo può essere utilizzato per chiudere manualmente una finestra di messaggio simulando la pressione di un pulsante (anche se esistono modi più semplici per farlo):

```
/* Simulate the press of the first button (indexes start from zero) */
uint32_t btn_id = 0;
lv_obj_send_event(mbox, LV_EVENT_VALUE_CHANGED, &btn_id);
```

Lo stesso vale per i dispositivi di visualizzazione e input con `lv_display_send_event(widget, <EVENT_CODE>, &some_data)` e `lv_indev_send_event(widget, <EVENT_CODE>, &some_data)`.

5.9.5 Campi di `lv_event_t`

`lv_event_t` è l'unico parametro passato alla callback dell'evento e contiene tutti i dati relativi all'evento. Da esso si possono recuperare i seguenti valori:

- *lv_event_get_code(e)*: ottiene il codice dell'evento
- *lv_event_get_current_target(e)*: ottiene il Widget a cui è stato inviato un evento. Ovvero il Widget il cui gestore eventi viene chiamato.
- *lv_event_get_target(e)*: ottiene il Widget, Display o Indev che ha originariamente attivato l'evento (diverso da *lv_event_get_current_target()* se *event bubbling* è abilitato).
- *lv_event_get_target_obj(e)*: ottiene il Widget che ha originariamente attivato l'evento.

- `lv_event_get_user_data(e)`: ottiene il puntatore passato come ultimo parametro di `lv_obj_add_event_cb()`.
- `lv_event_get_param(e)`: ottiene il parametro passato come ultimo parametro di `lv_obj_send_event_cb()`

Suggerimento

Quando si utilizza C++, preferire `lv_event_get_target_obj(e)` a `lv_event_get_target(e)` quando si sa che il target è un Widget, poiché restituisce il tipo corretto senza richiedere un cast.

Avvertimento

Chiamare `lv_event_get_target_obj(e)` solo quando si sa che il target dell'evento è un Widget. Chiamarlo per target Display o Indev è considerato un Comportamento Indefinito.

5.9.6 Event Bubbling

Se `lv_obj_add_flag(widget, LV_OBJ_FLAG_EVENT_BUBBLE)` è abilitato, tutti gli eventi verranno inviati anche al Widget padre. Se anche il Widget padre ha `LV_OBJ_FLAG_EVENT_BUBBLE` abilitato, l'evento verrà inviato al Widget padre e così via.

Il parametro *target* dell'evento è sempre il Widget di destinazione corrente, non il Widget originale. Per ottenere il target originale, si chiama `lv_event_get_target_obj(e)` nel gestore eventi.

5.9.7 Event Trickle

Noto anche come "Event Capturing", se `lv_obj_add_flag(widget, LV_OBJ_FLAG_EVENT_TRICKLE)` è abilitato, tutti gli eventi verranno inviati anche ai Widget figli. Questo è l'opposto del "bubbling" degli eventi: invece di propagarsi verso l'alto, gli eventi si propagano verso il basso fino ai widget figli.

Il meccanismo di trickle riguarda solo i widget figli immediati, non i nipoti o i discendenti più profondi. Affinché gli eventi si propaghino a livelli più profondi, ogni widget figlio deve avere il flag `LV_OBJ_FLAG_EVENT_TRICKLE` abilitato.

Come con il bubbling, il parametro *target* dell'evento è sempre il Widget di destinazione corrente, non il Widget originale. Per ottenere il target originale, si chiama `lv_event_get_target_obj(e)` nel gestore eventi.

5.9.8 Esempi

Evento clic pulsante

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void event_cb(lv_event_t * e)
{
    LV_LOG_USER("Clicked");

    static uint32_t cnt = 1;
    lv_obj_t * btn = lv_event_get_target_obj(e);
    lv_obj_t * label = lv_obj_get_child(btn, 0);
    lv_label_set_text_fmt(label, "%" LV_PRIu32, cnt);
    cnt++;
}

/**
 * Add click event to a button
 */
void lv_example_event_click(void)
{
    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 50);
    lv_obj_center(btn);
    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_CLICKED, NULL);

    lv_obj_t * label = lv_label_create(btn);
    lv_label_set_text(label, "Click me!");
    lv_obj_center(label);
}

#endif
```

Sequenze di clic

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

static void short_click_event_cb(lv_event_t * e)
{
    LV_LOG_USER("Short clicked");

    lv_obj_t * info_label = (lv_obj_t *) lv_event_get_user_data(e);
    lv_indev_t * indev = (lv_indev_t *) lv_event_get_param(e);
    uint8_t cnt = lv_indev_get_short_click_streak(indev);
    lv_label_set_text_fmt(info_label, "Short click streak: %u", cnt);
}

static void streak_event_cb(lv_event_t * e)
{
    lv_obj_t * btn = lv_event_get_target_obj(e);
    lv_obj_t * label = lv_obj_get_child(btn, 0);
    const char * text = (const char *) lv_event_get_user_data(e);
    lv_label_set_text(label, text);
}

void lv_example_event_streak(void)
{
    lv_obj_t * info_label = lv_label_create(lv_screen_active());
    lv_label_set_text(info_label, "No events yet");

    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 50);
    lv_obj_center(btn);
    lv_obj_add_event_cb(btn, short_click_event_cb, LV_EVENT_SHORT_CLICKED, info_label);
    lv_obj_add_event_cb(btn, streak_event_cb, LV_EVENT_SINGLE_CLICKED, (void *) "Single clicked");
    lv_obj_add_event_cb(btn, streak_event_cb, LV_EVENT_DOUBLE_CLICKED, (void *) "Double clicked");
    lv_obj_add_event_cb(btn, streak_event_cb, LV_EVENT_TRIPLE_CLICKED, (void *) "Triple clicked");

    lv_obj_t * label = lv_label_create(btn);
    lv_label_set_text(label, "Click me!");
    lv_obj_center(label);
}

#endif
```

Gestione di eventi multipli

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * label = (lv_obj_t *) lv_event_get_user_data(e);

    switch(code) {
        case LV_EVENT_PRESSED:
            lv_label_set_text(label, "The last button event:\nLV_EVENT_PRESSED");
            break;
        case LV_EVENT_CLICKED:
            lv_label_set_text(label, "The last button event:\nLV_EVENT_CLICKED");
            break;
        case LV_EVENT_LONG_PRESSED:
            lv_label_set_text(label, "The last button event:\nLV_EVENT_LONG_PRESSED");
            break;
        case LV_EVENT_LONG_PRESSED_REPEAT:
            lv_label_set_text(label, "The last button event:\nLV_EVENT_LONG_PRESSED_REPEAT");
            break;
        default:
            break;
    }
}

/** 
 * Handle multiple events
 */
void lv_example_event_button(void)
{
    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 50);
    lv_obj_center(btn);

    lv_obj_t * btn_label = lv_label_create(btn);
    lv_label_set_text(btn_label, "Click me!");

    lv_obj_t * info_label = lv_label_create(lv_screen_active());
    lv_label_set_text(info_label, "The last button event:\nNone");

    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_ALL, info_label);
}

#endif
```

Evento "bubbling"

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static void event_cb(lv_event_t * e)
{
    /*The original target of the event. Can be the buttons or the container*/
    lv_obj_t * target = lv_event_get_target_obj(e);

    /*The current target is always the container as the event is added to it*/
    lv_obj_t * cont = lv_event_get_current_target_obj(e);
```

(continues on next page)

(continua dalla pagina precedente)

```

/*If container was clicked do nothing*/
if(target == cont) return;

/*Make the clicked buttons red*/
lv_obj_set_style_bg_color(target, lv_palette_main(LV_PALETTE_RED), 0);
}

/***
 * Demonstrate event bubbling
 */
void lv_example_event_bubble(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 290, 200);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    uint32_t i;
    for(i = 0; i < 30; i++) {
        lv_obj_t * btn = lv_button_create(cont);
        lv_obj_set_size(btn, 70, 50);
        lv_obj_add_flag(btn, LV_OBJ_FLAG_EVENT_BUBBLE);

        lv_obj_t * label = lv_label_create(btn);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);
        lv_obj_center(label);
    }

    lv_obj_add_event_cb(cont, event_cb, LV_EVENT_CLICKED, NULL);
}

#endif

```

Evento "trickle-down"

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

/***
 * Demonstrate event trickle
 */
void lv_example_event_trickle(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 290, 200);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    static lv_style_t style_black;
    lv_style_init(&style_black);
    lv_style_set_text_color(&style_black, lv_color_white());
    lv_style_set_bg_color(&style_black, lv_color_black());

    /*Enable event trickle-down on the container*/
    lv_obj_add_flag(cont, LV_OBJ_FLAG_EVENT_TRICKLE);

    lv_obj_add_style(cont, &style_black, LV_STATE_PRESSED);

    uint32_t i;
    for(i = 0; i < 9; i++) {
        lv_obj_t * subcont = lv_obj_create(cont);
        lv_obj_set_size(subcont, 70, 50);
        lv_obj_t * label = lv_label_create(subcont);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);

        /*Add style to the label when clicked*/
        lv_obj_add_style(subcont, &style_black, LV_STATE_FOCUSED);
    }
}
#endif

```

Evento "Draw"

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

static int32_t size = 0;
static bool size_dec = false;

static void timer_cb(lv_timer_t * timer)
{
    lv_obj_invalidate((lv_obj_t *) lv_timer_get_user_data(timer));
    if(size_dec) size--;
    else size++;

    if(size == 50) size_dec = true;
    else if(size == 0) size_dec = false;
}

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_dsc_base_t * base_dsc = (lv_dsc_base_t *) lv_draw_task_get_dsc(draw_task);
    if(base_dsc->part == LV_PART_MAIN) {
        lv_draw_rect_dsc_t draw_dsc;
        lv_draw_rect_dsc_init(&draw_dsc);
        draw_dsc.bg_color = lv_color_hex(0xffffaa);
        draw_dsc.radius = LV_RADIUS_CIRCLE;
        draw_dsc.border_color = lv_color_hex(0xff5555);
        draw_dsc.border_width = 2;
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

draw_dsc.outline_color = lv_color_hex(0xffff0000);
draw_dsc.outline_pad = 3;
draw_dsc.outline_width = 2;

lv_area_t a;
a.x1 = 0;
a.y1 = 0;
a.x2 = size;
a.y2 = size;
lv_obj_get_coords(obj, &obj_coords);
lv_obj_align(&obj_coords, &a, LV_ALIGN_CENTER, 0, 0);

lv_draw_rect(base_dsc->layer, &draw_dsc, &a);
}

/***
 * Demonstrate the usage of draw event
 */
void lv_example_event_draw(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 200, 200);
    lv_obj_center(cont);
    lv_obj_add_event_cb(cont, event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(cont, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
    lv_timer_create(timer_cb, 30, cont);
}
#endif

```

5.9.9 API

lv_obj.h
lv_obj_event.h
lv_obj_event_private.h
lv_display.h
lv_indev.h
lv_event.h
lv_event_private.h

5.10 I Flag

Alcuni attributi dei widget possono essere abilitati o disabilitati utilizzando `lv_obj_add_flag(widget, LV_OBJ_FLAG_...)` e `lv_obj_remove_flag(widget, LV_OBJ_FLAG_...)`.

Per risparmiare memoria, i widget memorizzano questi flag in un bitfield. Per verificare se un flag è impostato, utilizzare: `lv_obj_has_flag(obj, LV_OBJ_FLAG_...)`.

I flag disponibili sono:

- `LV_OBJ_FLAG_HIDDEN`: Nasconde il widget (come se non ci fosse).
- `LV_OBJ_FLAG_CLICKABLE`: Rende il widget cliccabile dai dispositivi di input.
- `LV_OBJ_FLAG_CLICK_FOCUSABLE`: Aggiunge lo stato di focus al widget quando viene cliccato.
- `LV_OBJ_FLAG_CHECKABLE`: Attiva/disattiva lo stato di "checked" quando si clicca sul widget.
- `LV_OBJ_FLAG_SCROLLABLE`: Rende il widget "scrollable".
- `LV_OBJ_FLAG_SCROLL_ELASTIC`: Consente lo scorrimento elastico con movimento più lento.
- `LV_OBJ_FLAG_SCROLL_MOMENTUM`: Abilita lo scorrimento "momentum" [abbrevio] (continua lo scorrimento quando "lanciato").
- `LV_OBJ_FLAG_SCROLL_ONE`: Consente lo scrolling di un solo elemento figlio "snappable" [agganciabile].
- `LV_OBJ_FLAG_SCROLL_CHAIN_HOR`: Propaga lo scorrimento orizzontale all'elemento padre.
- `LV_OBJ_FLAG_SCROLL_CHAIN_VER`: Propaga lo scorrimento verticale all'elemento padre.
- `LV_OBJ_FLAG_SCROLL_CHAIN`: Abbreviazione di `SCROLL_CHAIN_HOR | SCROLL_CHAIN_VER`.

- *LV_OBJ_FLAG_SCROLL_ON_FOCUS*: Scorre automaticamente per rendere visibile il widget quando è attivo.
- *LV_OBJ_FLAG_SCROLL_WITH_ARROW*: Consente lo scrolling del widget attivo con i tasti freccia.
- *LV_OBJ_FLAG_SNAPPABLE*: Consente lo "snap" [aggancio] del widget se l'elemento padre ha abilitato l'aggancio con scorrimento.
- *LV_OBJ_FLAG_PRESS_LOCK*: Mantiene il widget nello stato "pressed" anche se il puntatore si sposta al di fuori di esso.
- *LV_OBJ_FLAG_EVENT_BUBBLE*: Propaga gli eventi all'elemento padre.
- *LV_OBJ_FLAG_EVENT_TRICKLE*: Propaga gli eventi ai figli.
- *LV_OBJ_FLAG_STATE_TRICKLE*: Propaga le modifiche di stato agli elementi figlio.
- *LV_OBJ_FLAG_GESTURE_BUBBLE*: Propaga i gesti all'elemento padre.
- *LV_OBJ_FLAG_ADV_HITTEST*: Abilita un test di hit (clic) più accurato (ad esempio, tenendo conto degli angoli arrotondati).
- *LV_OBJ_FLAG_IGNORE_LAYOUT*: Esclude il widget dal posizionamento del layout.
- *LV_OBJ_FLAG_FLOATING*: Non scorre con l'elemento padre e ignora il layout.
- *LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS*: Abilita l'invio di eventi `LV_EVENT_DRAW_TASK_ADDED`.
- *LV_OBJ_FLAG_OVERFLOW_VISIBLE*: Consente ai figli di superare i limiti del widget.
- *LV_OBJ_FLAG_FLEX_IN_NEW_TRACK*: Comincia una nuova "track" flex su questo elemento.
- *LV_OBJ_FLAG_LAYOUT_1*: Flag custom, utilizzabile liberamente dai layout.
- *LV_OBJ_FLAG_LAYOUT_2*: Flag custom, utilizzabile liberamente dai layout.
- *LV_OBJ_FLAG_WIDGET_1*: Flag custom, utilizzabile gratuitamente dai widget.
- *LV_OBJ_FLAG_WIDGET_2*: Flag custom, utilizzabile liberamente dai widget.
- *LV_OBJ_FLAG_USER_1*: Flag custom, libero da usare da parte dell'utente.
- *LV_OBJ_FLAG_USER_2*: Flag custom, libero da usare da parte dell'utente.
- *LV_OBJ_FLAG_USER_3*: Flag custom, liberamente utilizzabile dall'utente.
- *LV_OBJ_FLAG_USER_4*: Flag custom, liberamente utilizzabile dall'utente.

Alcuni esempi:

```
/* Hide a Widget */
lv_obj_add_flag(widget, LV_OBJ_FLAG_HIDDEN);

/* Make a Widget non-clickable */
lv_obj_remove_flag(widget, LV_OBJ_FLAG_CLICKABLE);

/* Check if it is clickable */
if(lv_obj_has_flag(widget, LV_OBJ_FLAG_CLICKABLE)) printf("Clickable\n");
```

5.10.1 Aggiunta e/o Rimozione di Più Flag

Quando si aggiungono o si rimuovono più flag, si hanno due opzioni:

Opzione 1: Chiamate multiple (Consigliata)

Questo approccio è più chiaro e funziona perfettamente sia in C che in C++:

```
lv_obj_add_flag(widget, LV_OBJ_FLAG_CLICKABLE);
lv_obj_add_flag(widget, LV_OBJ_FLAG_EVENT_BUBBLE);
```

Opzione 2: Chiamata singola con OR bit a bit

È possibile combinare più flag in un'unica chiamata utilizzando l'operatore OR bit a bit (`|`). Quando si utilizza un compilatore C++, è necessario eseguire il cast del risultato:

```
lv_obj_add_flag(widget, (lv_obj_flag_t)(LV_OBJ_FLAG_CLICKABLE | LV_OBJ_FLAG_EVENT_BUBBLE));
```

i Nota

Il cast a `lv_obj_flag_t` è obbligatorio in C++ a causa del controllo di tipo più rigoroso, ma è facoltativo in C.

5.11 Layout

5.11.1 Panoramica

I layout assegnati a un Widget gestiscono le posizioni e, in alcuni casi, le dimensioni degli elementi figlio di quel Widget. Possono essere utilizzati per disporre automaticamente gli elementi figlio in una riga, in una colonna o in schemi più complessi.

La posizione e le dimensioni impostate dal layout sovrascrivono le impostazioni "normali" di x, y e, in alcuni casi, di larghezza e altezza.

C'è una sola funzione che è la stessa per ogni layout: `lv_obj_set_layout()` (`widget, <LAYOUT_NAME>`) imposta il layout su un Widget. Per ulteriori impostazioni dell'elemento padre e degli elementi figlio, consultare la documentazione del layout in questione.

Layout Nativi

LVGL include due layout molto potenti:

- Flexbox: organizza i Widget secondari in righe o colonne, con supporto per il wrapping e l'espansione degli elementi.
- Grid: organizza i widget secondari in posizioni fisse in una tabella 2D.

Entrambi sono fortemente ispirati ai layout CSS con lo stesso nome. I layout sono descritti in dettaglio nella relativa sezione della documentazione.

I Flag

Esistono alcuni flag che possono essere utilizzati sui Widget per influenzarne il comportamento con i layout:

- `LV_OBJ_FLAG_HIDDEN` I Widget nascosti vengono ignorati nei calcoli del layout.
- `LV_OBJ_FLAG_IGNORE_LAYOUT` Il Widget viene semplicemente ignorato dai layout. Le sue coordinate possono essere impostate come di consueto.
- `LV_OBJ_FLAG_FLOATING` Uguale a `LV_OBJ_FLAG_IGNORE_LAYOUT` ma il Widget con `LV_OBJ_FLAG_FLOATING` verrà ignorato nei calcoli di `LV_SIZE_CONTENT`.

Questi flag possono essere aggiunti/rimossi con `lv_obj_add_flag(widget, FLAG)` e `lv_obj_remove_flag(widget, FLAG)`

Aggiungere Nuovi Layout

LVGL può essere liberamente esteso con un layout personalizzato come questo:

```
uint32_t MY_LAYOUT;
...
MY_LAYOUT = lv_layout_register(my_layout_update, &user_data);
...
void my_layout_update(lv_obj_t * widget, void * user_data)
{
    /* Will be called automatically if it's required to reposition/resize the children of "widget" */
}
```

È possibile aggiungere proprietà di stile personalizzate che possono essere recuperate e utilizzate nella callback di aggiornamento. Ad esempio:

```

uint32_t LV_STYLE_MY_PROP;
...

LV_STYLE_MY_PROP = lv_style_register_prop(LV_STYLE_PROP_FLAG_NONE);

...
static inline void lv_style_set_my_prop(lv_style_t * style, uint32_t value)
{
    lv_style_value_t v = {
        .num = (int32_t)value
    };
    lv_style_set_prop(style, LV_STYLE_MY_PROP, v);
}

```

5.11.2 Flex

Panoramica

Flexbox (o Flex in breve) è un sottoinsieme dei comportamenti del [CSS Flexbox](#).

Si possono disporre gli elementi (Widget figli) in righe o colonne (track), gestire il wrapping, regolare la spaziatura tra elementi e track, gestire l'aumento di dimensione per far sì che gli elementi riempiano lo spazio rimanente rispetto alla larghezza e all'altezza minime/massime.

Per trasformare un Widget in un contenitore Flex, si chiama `lv_obj_set_layout(widget, LV_LAYOUT_FLEX)`.

Si noti che la funzionalità di layout Flex di LVGL deve essere abilitata globalmente con `LV_USE_FLEX` in `lv_conf.h`.

Termini

- **track** [tracce]: righe o colonne
- **direzione principale**: riga o colonna, la direzione in cui vengono posizionati per primi più elementi
- **direzione trasversale**: la direzione perpendicolare alla **direzione principale**
- **wrap**: se non c'è più spazio nella traccia, ne viene iniziata una nuova
- **grow** [crescita]: se impostato su un elemento, "crescerà" per riempire lo spazio rimanente nella traccia. Lo spazio disponibile verrà distribuito tra gli elementi in base al loro valore di crescita (un valore maggiore significa più spazio)
- **gap**: lo spazio tra righe e colonne o tra gli elementi di una track

Vedere [CSS Flexbox](#) per illustrazioni che mostrano il significato di questi termini.

Interfaccia Semplice

Si usano le seguenti funzioni per impostare e controllare il layout Flex su qualsiasi Widget genitore.

Nota

Affinché questi stili abbiano effetto, il Widget genitore deve essere un contenitore Flex. Le funzioni seguenti fanno sì che il Widget genitore diventi un contenitore Flex, se non lo è già.

Flusso Flex

`lv_obj_set_flex_flow(widget, flex_flow)`

I possibili valori per `flex_flow` sono:

- `LV_FLEX_FLOW_ROW`: Posiziona gli elementi figlio in una riga senza riavvolgimento
- `LV_FLEX_FLOW_COLUMN`: Posiziona gli elementi figlio in una colonna senza riavvolgimento
- `LV_FLEX_FLOW_ROW_WRAP`: Posiziona gli elementi figlio in una riga con riavvolgimento
- `LV_FLEX_FLOW_COLUMN_WRAP`: Posiziona gli elementi figlio in una colonna con riavvolgimento
- `LV_FLEX_FLOW_ROW_REVERSE`: Posiziona gli elementi figlio in una riga senza riavvolgimento ma in ordine inverso

- *LV_FLEX_FLOW_COLUMN_REVERSE*: Posiziona gli elementi figlio in una colonna senza riavvolgimento ma in ordine inverso
- *LV_FLEX_FLOW_ROW_WRAP_REVERSE*: Posiziona gli elementi figlio in una riga con riavvolgimento ma in ordine inverso
- *LV_FLEX_FLOW_COLUMN_WRAP_REVERSE*: Posiziona gli elementi figlio in una colonna con riavvolgimento ma in ordine inverso

Questi valori fanno sì che il comportamento del layout del Widget modelli il comportamento di [CSS Flexbox](#) combinando `flex-direction` e `flex-wrap` come definito in `flex-flow`.

Allineamento Flex

Per gestire il posizionamento degli elementi figlio, si usa `lv_obj_set_flex_align(widget, main_place, cross_place, track_cross_place)` che fa sì che il modello del widget padre imposti il comportamento del contenitore Flex definito [qui](#).

- `main_place` determina come distribuire gli elementi nella loro track sull'asse principale. Ad esempio, allinea gli elementi a destra su *LV_FLEX_FLOW_ROW_WRAP*. (In CSS si chiama `justify-content`.)
- `cross_place` determina come distribuire gli elementi nella loro track sull'asse trasversale. Ad esempio, se gli elementi hanno altezze diverse, li allinea alla parte inferiore della track. (In CSS si chiama `align-items`.)
- `track_cross_place` determina come distribuire le track (in CSS si chiama `align-content`.)

I valori possibili sono:

- *LV_FLEX_ALIGN_START*: indica sinistra quando la direzione è orizzontale, in alto quando è verticale (il default)
- *LV_FLEX_ALIGN_END*: indica destra quando la direzione è orizzontale, in basso quando è verticale
- *LV_FLEX_ALIGN_CENTER*: semplicemente centra rispetto alla direzione
- *LV_FLEX_ALIGN_SPACE_EVENLY*: gli elementi vengono distribuiti in modo che la spaziatura tra due elementi qualsiasi (e lo spazio tra i bordi) sia uguale. Non si applica a `track_cross_place`.
- *LV_FLEX_ALIGN_SPACE_AROUND*: gli elementi sono distribuiti uniformemente nella track con uguale spazio attorno a loro. Si noti che visivamente gli spazi non sono uguali poiché tutti gli elementi hanno uguale spazio su entrambi i lati. Questo fa sì che lo spazio tra gli elementi sia il doppio dello spazio tra gli elementi sul bordo e il bordo del contenitore. Non si applica a `track_cross_place`.
- *LV_FLEX_ALIGN_SPACE_BETWEEN*: gli elementi sono distribuiti uniformemente nella traccia senza spazio prima e dopo il primo e l'ultimo elemento. Non si applica a `track_cross_place`.

Vedere [justify-content](#), [align-items](#) e [align-content](#) per le illustrazioni di questi valori.

Flex grow

Flex grow [crescita] può essere utilizzato per far sì che uno o più widget figlio occupino lo spazio disponibile nella track. Quando più di un widget figlio ha valori di grow diversi da zero, tutto lo spazio disponibile verrà distribuito in proporzione ai rispettivi valori di grow. Ad esempio, se ci sono 400 px di spazio rimanenti e 3 widget figlio con valori di crescita diversi da zero:

- A con grow = 1
- B con grow = 1
- C con grow = 2

A e B occuperanno 100 px, e C occuperà 200 px.

Flex grow può essere impostato su un Widget figlio con `lv_obj_set_flex_grow(child, value)`. `value` deve essere ≥ 1 o 0 per disabilitare la crescita sul widget figlio.

Vedere [flex-grow](#) per un'illustrazione di questo comportamento.

Interfaccia dello Stile

Tutti i valori relativi a Flex sono proprietà di stile nascoste; quindi si possono usare come qualsiasi altra proprietà di stile.

Esistono le seguenti proprietà di stile relative a Flex:

- FLEX_FLOW
- FLEX_MAIN_PLACE
- FLEX_CROSS_PLACE
- FLEX_TRACK_PLACE
- FLEX_GROW

Spaziatura interna

Per modificare lo spazio minimo inserito da Flexbox tra i Widget, è possibile utilizzare le seguenti funzioni per impostare lo stile della spaziatura del contenitore Flex:

- `lv_style_set_pad_row()` imposta la spaziatura interna tra le righe.
- `lv_style_set_pad_column()` imposta la spaziatura interna tra le colonne.

Possono essere utilizzati, ad esempio, se non si desidera alcun padding tra i Widgets: `lv_style_set_pad_column(&row_container_style, 0)`

Altre Funzionalità

RTL

Se la direzione di base del contenitore è impostata su `LV_BASE_DIRRTL` il significato di `LV_FLEX_ALIGN_START` e `LV_FLEX_ALIGN_END` viene scambiato nei layout ROW. Ad esempio, START significherà destra.

Gli elementi nei layout ROW e le track di COLUMN verranno posizionati da destra a sinistra.

Nuova track

È possibile forzare Flex a inserire un elemento in una nuova riga con `lv_obj_add_flag(child, LV_OBJ_FLAG_FLEX_IN_NEW_TRACK)`.

Ulteriori Informazioni

Maggiori informazioni su CSS Flexbox.

Esempi

Un semplice layout di riga e colonna con flexbox

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/***
 * A simple row and a column layout with flexbox
 */
void lv_example_flex_1(void)
{
    /*Create a container with ROW flex direction*/
    lv_obj_t * cont_row = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont_row, 300, 75);
    lv_obj_align(cont_row, LV_ALIGN_TOP_MID, 0, 5);
    lv_obj_set_flex_flow(cont_row, LV_FLEX_FLOW_ROW);

    /*Create a container with COLUMN flex direction*/
    lv_obj_t * cont_col = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont_col, 200, 150);
    lv_obj_align_to(cont_col, cont_row, LV_ALIGN_OUT_BOTTOM_MID, 0, 5);
    lv_obj_set_flex_flow(cont_col, LV_FLEX_FLOW_COLUMN);

    uint32_t i;
    for(i = 0; i < 10; i++) {

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * obj;
lv_obj_t * label;

/*Add items to the row*/
obj = lv_button_create(cont_row);
lv_obj_set_size(obj, 100, LV_PCT(100));

label = lv_label_create(obj);
lv_label_set_text_fmt(label, "Item: %" LV_PRIu32, i);
lv_obj_center(label);

/*Add items to the column*/
obj = lv_button_create(cont_col);
lv_obj_set_size(obj, LV_PCT(100), LV_SIZE_CONTENT);

label = lv_label_create(obj);
lv_label_set_text_fmt(label, "Item: %" LV_PRIu32, i);
lv_obj_center(label);
}

#endif

```

Disporre gli elementi in righe con "wrap" [riavvolgimento] e spaziatura uniforme

```

#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/***
 * Arrange items in rows with wrap and place the items to get even space around them.
 */
void lv_example_flex_2(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_flex_flow(&style, LV_FLEX_FLOW_ROW_WRAP);
    lv_style_set_flex_main_place(&style, LV_FLEX_ALIGN_SPACE_EVENLY);
    lv_style_set_layout(&style, LV_LAYOUT_FLEX);

    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_add_style(cont, &style, 0);

    uint32_t i;
    for(i = 0; i < 8; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);
        lv_obj_add_flag(obj, LV_OBJ_FLAG_CHECKABLE);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);
        lv_obj_center(label);
    }
}
#endif

```

Dimostrazione della flex grow.

```

#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/***
 * Demonstrate flex grow.
 */
void lv_example_flex_3(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW);

    lv_obj_t * obj;
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, 40, 40); /*Fix size*/

    obj = lv_obj_create(cont);
    lv_obj_set_height(obj, 40);
    lv_obj_set_flex_grow(obj, 1); /*1 portion from the free space*/

    obj = lv_obj_create(cont);
    lv_obj_set_height(obj, 40);
    lv_obj_set_flex_grow(obj, 2); /*2 portion from the free space*/

    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, 40, 40); /*Fix size. It is flushed to the right by the "grow" items*/
}
#endif

```

Dimostrazione della flex grow.

```

#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/***
 * Reverse the order of flex items
 */

```

(continues on next page)

(continua dalla pagina precedente)

```
/*
void lv_example_flex_4(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_COLUMN_REVERSE);

    uint32_t i;
    for(i = 0; i < 6; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 100, 50);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item: %" LV_PRIu32, i);
        lv_obj_center(label);
    }
}

#endif
```

Dimostrazione delle proprietà di stile degli spazi tra colonne e righe

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

static void row_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_row((lv_obj_t *) obj, v, 0);
}

static void column_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_column((lv_obj_t *) obj, v, 0);
}

/**
 * Demonstrate the effect of column and row gap style properties
 */
void lv_example_flex_5(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    uint32_t i;
    for(i = 0; i < 9; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);
        lv_obj_center(label);
    }

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, cont);
    lv_anim_set_values(&a, 0, 10);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);

    lv_anim_set_exec_cb(&a, row_gap_anim);
    lv_anim_set_duration(&a, 500);
    lv_anim_set_reverse_duration(&a, 500);
    lv_anim_start(&a);

    lv_anim_set_exec_cb(&a, column_gap_anim);
    lv_anim_set_duration(&a, 3000);
    lv_anim_set_reverse_duration(&a, 3000);
    lv_anim_start(&a);
}
#endif
```

La direzione di base RTL modifica l'ordine degli elementi

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/**
 * RTL base direction changes order of the items.
 * Also demonstrate how horizontal scrolling works with RTL.
 */
void lv_example_flex_6(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_style_base_dir(cont, LV_BASE_DIRRTL, 0);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    uint32_t i;
    for(i = 0; i < 20; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);
        lv_obj_center(label);
    }
}
#endif
```

API

lv_flex.h

5.11.3 Grid

Panoramica

Il layout "Grid" [a griglia] è un sottoinsieme del layout CSS Grid.

Può disporre gli elementi (Widget figlio) in una "tabella" 2D con righe e colonne (track). Un elemento può estendersi su più colonne o righe. La dimensione della track può essere impostata in pixel, in base all'elemento più grande ([LV_GRID_CONTENT](#)) o in base a una frazione dello spazio libero disponibile (ad esempio [Free \[FR\] Units](#)) per distribuire lo spazio libero in modo proporzionale.

Per trasformare un Widget in un contenitore Grid, chiamare `lv_obj_set_layout(widget, LV_LAYOUT_GRID)`.

Si noti che la funzionalità di layout Grid di LVGL deve essere abilitata globalmente con `LV_USE_GRID` in `lv_conf.h`.

Termini

- **track** [tracce]: righe o colonne
- **free (FR) units**: se la dimensione di una track è impostata in unità FR, questa aumenterà fino a riempire lo spazio rimanente nel Widget padre (contenitore), in proporzione alle altre tracce che hanno valori di unità FR diversi da zero.
- **gap**: lo spazio tra righe e colonne o tra gli elementi di una track

Interfaccia Semplice

Con le seguenti funzioni è possibile impostare qualsiasi Widget padre su un layout Grid.

Nota

Come per i contenitori Flex, il Widget padre deve essere un contenitore Grid affinché questi stili abbiano effetto. Le funzioni seguenti fanno sì che il Widget padre diventi un contenitore Grid, se non lo è già.

Descrittori di Grid

Per prima cosa è necessario descrivere la dimensione di righe e colonne. Questo può essere fatto dichiarando 2 array e le dimensioni delle track in essi contenute. L'ultimo elemento deve essere `LV_GRID_TEMPLATE_LAST`.

Ad esempio:

```
static int32_t column_dsc[] = {100, 400, LV_GRID_TEMPLATE_LAST}; /* 2 columns with 100- and 400-px width */
static int32_t row_dsc[] = {100, 100, 100, LV_GRID_TEMPLATE_LAST}; /* 3 100-px tall rows */
```

Per impostare i descrittori su un elemento padre, si usa `lv_obj_set_grid_dsc_array(widget, col_dsc, row_dsc)`.

Oltre a impostare le dimensioni in pixel, è possibile utilizzare due valori speciali:

- `LV_GRID_CONTENT` impone la dimensione in modo che si adatti all'elemento figlio più grande di questa track
- `LV_GRID_FR(X)` determina quale porzione dello spazio rimanente deve essere utilizzata da questa track. Valori più grandi indicano spazio maggiore.

Elementi della Grid

Per default, gli elementi figlio di un Widget non vengono aggiunti alla grid. Devono essere aggiunti manualmente a una cella.

Per farlo, si chiama `lv_obj_set_grid_cell(child, column_align, column_pos, column_span, row_align, row_pos, row_span)`.

`column_align` e `row_align` determinano come allineare il Widget figlio nella sua cella. I valori possibili sono:

- `LV_GRID_ALIGN_START`: indica a sinistra quando la direzione è orizzontale e in alto quando è verticale (il default)
- `LV_GRID_ALIGN_END`: indica a destra quando la direzione è orizzontale e in basso quando è verticale.
- `LV_GRID_ALIGN_CENTER`: semplicemente centra `column_pos` e `row_pos`, indica l'indice a partire da zero della cella in cui deve essere posizionato l'elemento.

`column_span` e `row_span` indicano quante track devono essere occupate dalla cella iniziale. Deve essere ≥ 1 .

Allineamento nella Grid

Se c'è dello spazio vuoto, gli elementi (Widget) nelle track della Grid possono essere allineati in diversi modi:

- `LV_GRID_ALIGN_START`: indica sinistra quando la direzione è orizzontale e in alto quando è verticale. (Il default).
- `LV_GRID_ALIGN_END`: indica a destra quando la direzione è orizzontale e in basso quando è verticale.
- `LV_GRID_ALIGN_CENTER`: semplicemente centra
- `LV_GRID_ALIGN_SPACE_EVENLY`: gli elementi sono distribuiti in modo che la spaziatura tra due elementi qualsiasi (e lo spazio fino ai bordi) sia uguale. Non si applica a `track_cross_place`.
- `LV_GRID_ALIGN_SPACE_AROUND`: gli elementi sono distribuiti uniformemente nella track con lo stesso spazio intorno a loro. Si noti che visivamente gli spazi non sono uguali, poiché tutti gli elementi hanno lo stesso spazio su entrambi i lati. Questo fa sì che lo spazio tra gli elementi sia il doppio dello spazio tra gli elementi sul bordo e il bordo del contenitore. Non si applica a `track_cross_place`.
- `LV_GRID_ALIGN_SPACE_BETWEEN`: gli elementi sono distribuiti uniformemente nella track con il primo e l'ultimo elemento adiacenti ai bordi del contenitore. Non si applica a `track_cross_place`.

Per impostare l'allineamento della track, si usa `lv_obj_set_grid_align(widget, column_align, row_align)`.

Sub grid

Se si impostano i descrittori di grid di colonna e/o riga di un widget su `NULL`, verranno utilizzati i descrittori di grid del genitore.

Ad esempio, se si crea un elemento grid che si estende sulle colonne 2..6 e sulle righe 1..3 della grid, l'elemento grid occuperà 5 colonne e 4 righe con le dimensioni di track corrispondenti al contenitore Grid padre.

In questo modo, anche se nella grid viene utilizzato un elemento wrapper, può essere reso "trasparente" dal punto di vista della grid.

Limitazioni:

- La sub-grid viene risolta solo fino a una profondità di 1 livello. Ovvero, una grid può avere una sub-grid figlia, ma tale sub-grid non può avere un'altra sub-grid.
- `LV_GRID_CONTENT` le track sulla grid non vengono gestite nella sub-grid, ma solo nella propria grid.

La funzionalità della sub-grid funziona come in CSS. Per ulteriori informazioni, vedere [CSS Subgrid](#).

Interfaccia dello Stile

Tutti i valori relativi alla Grid sono proprietà di stile nascoste, quindi è possibile utilizzarle come qualsiasi altra proprietà di stile.

Esistono le seguenti proprietà di stile relative alla Grid:

- `GRID_COLUMN_DSC_ARRAY`
- `GRID_ROW_DSC_ARRAY`
- `GRID_COLUMN_ALIGN`
- `GRID_ROW_ALIGN`

- GRID_CELL_X_ALIGN
- GRID_CELL_COLUMN_POS
- GRID_CELL_COLUMN_SPAN
- GRID_CELL_Y_ALIGN
- GRID_CELL_ROW_POS
- GRID_CELL_ROW_SPAN

Spaziatura interna

Per modificare lo spazio minimo di inserimento della Grid tra i widget, è possibile impostare le seguenti proprietà nello stile del contenitore della Grid:

- `lv_style_set_pad_row()` imposta la spaziatura interna tra le righe.
- `lv_style_set_pad_column()` imposta la spaziatura interna tra le colonne.

Altre funzionalità

RTL

Se la direzione di base del contenitore è impostata su `LV_BASE_DIRRTL`, il significato di `LV_GRID_ALIGN_START` e `LV_GRID_ALIGN_END` viene invertito. Ad esempio, START significherà "più a destra".

Le colonne verranno posizionate da destra a sinistra.

Ulteriori Informazioni

- Maggiori informazioni sul layout [CSS Grid](#).
- Maggiori informazioni sul layout [CSS Subgrid](#).

Esempio

Una semplice grid

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/***
 * A simple grid
 */
void lv_example_grid_1(void)
{
    static int32_t col_dsc[] = {70, 70, 70, LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {50, 50, 50, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_style_grid_column_dsc_array(cont, col_dsc, 0);
    lv_obj_set_style_grid_row_dsc_array(cont, row_dsc, 0);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_layout(cont, LV_LAYOUT_GRID);

    lv_obj_t * label;
    lv_obj_t * obj;

    uint8_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_button_create(cont);
        /*Stretch the cell horizontally and vertically too
         *Set span to 1 to make the cell 1 column/row sized*/
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                             LV_GRID_ALIGN_STRETCH, row, 1);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "c%d, r%d", col, row);
        lv_obj_center(label);
    }
}

#endif
```

Posizionamento e ampiezza delle celle

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/***
 * Demonstrate cell placement and span
 */
void lv_example_grid_2(void)
{
    static int32_t col_dsc[] = {70, 70, 70, LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {50, 50, 50, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);

    lv_obj_t * label;
    lv_obj_t * obj;

    /*Cell to 0:0 and align to the start (left/top) horizontally and vertically too*/
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_START, 0, 1,
                        LV_GRID_ALIGN_START, 0, 1);
    label = lv_label_create(obj);
    lv_label_set_text(label, "c0, r0");

    /*Cell to 1:0 and align to the start (left) horizontally and center vertically too*/
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_START, 1, 1,
                        LV_GRID_ALIGN_CENTER, 0, 1);
    label = lv_label_create(obj);
    lv_label_set_text(label, "c1, r0");

    /*Cell to 2:0 and align to the start (left) horizontally and end (bottom) vertically too*/
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_START, 2, 1,
                        LV_GRID_ALIGN_END, 0, 1);
    label = lv_label_create(obj);
    lv_label_set_text(label, "c2, r0");

    /*Cell to 1:1 but 2 column wide (span = 2).Set width and height to stretched.*/
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, 1, 2,
                        LV_GRID_ALIGN_STRETCH, 1, 1);
    label = lv_label_create(obj);
    lv_label_set_text(label, "c1-2, r1");

    /*Cell to 0:1 but 2 rows tall (span = 2).Set width and height to stretched.*/
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, 0, 1,
                        LV_GRID_ALIGN_STRETCH, 1, 2);
    label = lv_label_create(obj);
    lv_label_set_text(label, "c0\nr1-2");
}

#endif
```

La -free unit- della grid

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/***
 * Demonstrate grid's "free unit"
 */
void lv_example_grid_3(void)
{
    /*Column 1: fix width 60 px
     *Column 2: 1 unit from the remaining free space
     *Column 3: 2 unit from the remaining free space*/
    static int32_t col_dsc[] = {60, LV_GRID_FR(1), LV_GRID_FR(2), LV_GRID_TEMPLATE_LAST};

    /*Row 1: fix width 50 px
     *Row 2: 1 unit from the remaining free space
     *Row 3: fix width 50 px*/
    static int32_t row_dsc[] = {50, LV_GRID_FR(1), 50, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);

    lv_obj_t * label;
    lv_obj_t * obj;
    uint8_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_obj_create(cont);
        /*Stretch the cell horizontally and vertically too
         *Set span to 1 to make the cell 1 column/row sized*/
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                            LV_GRID_ALIGN_STRETCH, row, 1);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%d,%d", col, row);
        lv_obj_center(label);
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```
}
#endif
```

Posizionamento delle tracce

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/***
 * Demonstrate track placement
 */
void lv_example_grid_4(void)
{
    static int32_t col_dsc[] = {60, 60, 60, LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {45, 45, 45, LV_GRID_TEMPLATE_LAST};

    /*Add space between the columns and move the rows to the bottom (end)*/

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_grid_align(cont, LV_GRID_ALIGN_SPACE_BETWEEN, LV_GRID_ALIGN_END);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);

    lv_obj_t * label;
    lv_obj_t * obj;
    uint8_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_obj_create(cont);
        /*Stretch the cell horizontally and vertically too
         *Set span to 1 to make the cell 1 column/row sized*/
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                             LV_GRID_ALIGN_STRETCH, row, 1);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%d,%d", col, row);
        lv_obj_center(label);
    }
}
#endif
```

Spazi tra colonne e righe

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

static void row_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_row((lv_obj_t * ) obj, v, 0);
}

static void column_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_column((lv_obj_t * ) obj, v, 0);
}

/***
 * Demonstrate column and row gap
 */
void lv_example_grid_5(void)
{
    /*60x60 cells*/
    static int32_t col_dsc[] = {60, 60, 60, LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {45, 45, 45, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);

    lv_obj_t * label;
    lv_obj_t * obj;
    uint8_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_obj_create(cont);
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                             LV_GRID_ALIGN_STRETCH, row, 1);
        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%d,%d", col, row);
        lv_obj_center(label);
    }

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, cont);
    lv_anim_set_values(&a, 0, 10);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);

    lv_anim_set_exec_cb(&a, row_gap_anim);
    lv_anim_set_duration(&a, 500);
    lv_anim_set_reverse_duration(&a, 500);
    lv_anim_start(&a);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_anim_set_exec_cb(&a, column_gap_anim);
lv_anim_set_duration(&a, 3000);
lv_anim_set_reverse_duration(&a, 3000);
lv_anim_start(&a);
}

#endif

```

Direzione RTL sulla griglia

```

#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/**
 * Demonstrate RTL direction on grid
 */
void lv_example_grid_6(void)
{
    static int32_t col_dsc[] = {60, 60, 60, LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {45, 45, 45, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_style_base_dir(cont, LV_BASE_DIR_RTL, 0);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);

    lv_obj_t * label;
    lv_obj_t * obj;
    uint8_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_obj_create(cont);
        /*Stretch the cell horizontally and vertically too
        *Set span to 1 to make the cell 1 column/row sized*/
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                            LV_GRID_ALIGN_STRETCH, row, 1);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%d,%d", col, row);
        lv_obj_center(label);
    }
}
#endif

```

API

lv_gridnav.h

lv_grid.h

5.12 Scrolling

5.12.1 Panoramica

In LVGL lo "scrolling" [scorrimento] funziona in modo molto intuitivo: se un Widget si trova al di fuori dell'area del contenuto principale (la dimensione senza padding), quest'ultima diventa scorrevole e vengono visualizzate le barre di scorrimento. Tutto qui.

Qualsiasi widget può essere "scrollable" [scorrevole], inclusi *Widget Base (lv_obj)*, *lv_image*, *lv_button*, *lv_meter*, ecc.

Il Widget può essere scorrevole orizzontalmente o verticalmente con un solo movimento; lo scorrimento diagonale non è possibile.

Scrollbar

Modalità

Le barre di scorrimento vengono visualizzate in base alla *scrollbar-mode* configurata. Sono disponibili le seguenti modalità:

- *LV_SCROLLBAR_MODE_OFF*: Non mostrare mai le barre di scorrimento
- *LV_SCROLLBAR_MODE_ON*: Mostra sempre le barre di scorrimento

- *LV_SCROLLBAR_MODE_ACTIVE*: Mostra le barre di scorrimento durante lo scrolling di un Widget
- *LV_SCROLLBAR_MODE_AUTO*: Mostra le barre quando il contenuto è sufficientemente grande da poter essere scorrevole

`lv_obj_set_scrollbar_mode(widget, LV_SCROLLBAR_MODE_...)` imposta la modalità della barra di scorrimento per un widget.

Stili

Una "Scrollbar" [barra di scorrimento] è una parte dedicata di un widget, chiamata *LV_PART_SCROLLBAR*. Ad esempio, una barra di scorrimento può diventare rossa in questo modo:

```
static lv_style_t style_red;
lv_style_init(&style_red);
lv_style_set_bg_color(&style_red, lv_color_red());

...
lv_obj_add_style(widget, &style_red, LV_PART_SCROLLBAR);
```

Un Widget passa allo stato *LV_STATE_SCROLLED* durante lo scorrimento. Questo consente di aggiungere diversi stili al Widget, che saranno attivi durante lo scorrimento. Ad esempio, questo codice rende la barra di scorrimento blu durante lo scorrimento del Widget:

```
static lv_style_t style_blue;
lv_style_init(&style_blue);
lv_style_set_bg_color(&style_blue, lv_color_blue());

...
lv_obj_add_style(widget, &style_blue, LV_STATE_SCROLLED | LV_PART_SCROLLBAR);
```

Se la direzione di base di *LV_PART_SCROLLBAR* è RTL (*LV_BASE_DIRRTL*) la barra di scorrimento verticale verrà posizionata a sinistra. Si noti che la proprietà di stile *base_dir* viene ereditata. Pertanto, può essere impostato direttamente sulla parte *LV_PART_SCROLLBAR* di un Widget, o sulla parte *LV_PART_MAIN* del Widget, o su quella di uno qualsiasi dei suoi elementi padre, per far sì che una barra di scorrimento erediti la direzione di base.

pad_left/right/top/bottom imposta la spaziatura intorno alle barre di scorrimento, *width* imposta la larghezza della barra e *length* ne imposta la lunghezza: se *length* non è impostato o è impostato a 0, la lunghezza della barra di scorrimento verrà impostata automaticamente in base alla lunghezza del contenuto.

```
static lv_style_t style_scrollbar;
lv_style_init(&style_scrollbar);
lv_style_set_pad_left(&style_scrollbar, 2);
lv_style_set_pad_right(&style_scrollbar, 2);
lv_style_set_pad_top(&style_scrollbar, 2);
lv_style_set_pad_bottom(&style_scrollbar, 2);
lv_style_set_width(&style_scrollbar, 10);
lv_style_set_length(&style_scrollbar, 50);

...
lv_obj_add_style(widget, &style_scrollbar, LV_PART_SCROLLBAR);
```

La lunghezza minima della barra di scorrimento è fissata a 10, mentre la lunghezza massima è limitata dall'altezza o dalla larghezza del widget, a seconda che la barra sia verticale o orizzontale. Qualsiasi valore di lunghezza impostato al di fuori di questi limiti comporterà automaticamente una lunghezza fissata a uno dei due limiti.

Eventi dello Scrolling

I seguenti eventi vengono generati come parte dello scorrimento:

- *LV_EVENT_SCROLL_BEGIN*: Segnala l'inizio dello scrolling. Il parametro *event* è NULL o *lv_anim_t* * con un descrittore di animazione di scorrimento che può essere modificato se necessario.
- *LV_EVENT_SCROLL_END*: Segnala la fine dello scrolling.
- *LV_EVENT_SCROLL*: Segnala la modifica della posizione di scorrimento; attivato a ogni cambio di posizione.

5.12.2 Funzionalità dello Scrolling

Oltre alla gestione dello scorrimento "normale", ci sono molte funzionalità aggiuntive interessanti e utili.

Scrollable

È possibile rendere un Widget non-scrollabile con `lv_obj_remove_flag(widget, LV_OBJ_FLAG_SCROLLABLE)`.

I Widget non-scrollabile possono comunque propagare lo scorrimento (chain [catena]) ai widget padre.

La direzione dello scorrimento può essere controllata da `lv_obj_set_scroll_dir(widget, LV_DIR_...)`.

Per la direzione è possibile utilizzare i seguenti valori:

- `LV_DIR_TOP`: scorrimento solo verso l'alto
- `LV_DIR_LEFT`: scorrimento solo verso sinistra
- `LV_DIR_BOTTOM`: scorrimento solo verso il basso
- `LV_DIR_RIGHT`: scorrimento solo verso destra
- `LV_DIR_HOR`: scorrimento solo orizzontalmente
- `LV_DIR_VER`: scorrimento solo verticalmente
- `LV_DIR_ALL`: scorrimento in qualsiasi direzione

Sono possibili anche valori in OR. Es. `LV_DIR_TOP | LV_DIR_LEFT`.

Concatenamento dello Scroll

Se un widget non può essere ulteriormente scorrevole (ad esempio, perché il suo contenuto ha raggiunto la posizione più bassa), lo scorrimento aggiuntivo viene propagato al suo elemento padre. Se il genitore può essere scorrevole in quella direzione, verrà invece "scrollato". Continua a propagarsi lungo la gerarchia del genitore del Widget fino a *Screen*.

La propagazione durante lo scorrimento è chiamata "scroll chaining" e può essere abilitata/disabilitata con il flag `LV_OBJ_FLAG_SCROLL_CHAIN_HOR/VER`. Se il chaining è disabilitato, la propagazione si interrompe sul Widget e il genitore/i non verrà fatto scorrere.

Scorrimento dinamico

Quando l'utente scorre un widget e lo rilascia, LVGL può emulare lo slancio inerziale per lo scorrimento. È come se il widget fosse stato "lanciato" e lo scorrimento rallenta gradualmente.

Lo slancio dinamico può essere abilitato/disabilitato con il flag `LV_OBJ_FLAG_SCROLL_MOMENTUM`.

Scorrimento elastico

Normalmente non è possibile scorrere un Widget oltre le estremità del suo contenuto. Ovvero, il lato superiore del contenuto non può trovarsi al di sotto del lato superiore del Widget e viceversa per il lato inferiore.

Tuttavia, con `LV_OBJ_FLAG_SCROLL_ELASTIC` viene aggiunto un effetto particolare quando l'utente "scorre eccessivamente" il contenuto. Lo scorrimento rallenta e il contenuto può essere fatto scorrere all'interno del Widget. Quando il Widget viene rilasciato, il contenuto che scorre al suo interno viene riportato alla posizione valida più vicina.

Aggancio

I figli di un Widget possono essere agganciati secondo regole specifiche al termine dello scorrimento. I figli possono essere resi agganciabili individualmente con il flag `LV_OBJ_FLAG_SNAPPABLE`.

Un Widget può allineare i figli agganciati in quattro modi:

- `LV_SCROLL_SNAP_NONE`: Lo "Snap" [aggancio] è disabilitato. (Il default).
- `LV_SCROLL_SNAP_START`: Allinea gli elementi figlio a sinistra/in alto di un Widget scorrevole

- [LV_SCROLL_SNAP_END](#): Allinea gli elementi figlio a destra/in basso di un Widget scorrevole
- [LV_SCROLL_SNAP_CENTER](#): Allinea gli elementi figlio al centro di un Widget scorrevole

L'allineamento snap viene impostato con `lv_obj_set_scroll_snap_x(widget, LV_SCROLL_SNAP_...)` e `lv_obj_set_scroll_snap_y(widget, LV_SCROLL_SNAP_...)`.

Ecco cosa succede dietro le quinte:

1. L'utente scorre e rilascia un Widget;
2. LVGL calcola dove terminerebbe lo scorrimento considerando l'abbrivio dello scorrimento;
3. LVGL trova il punto di scorrimento più vicino;
4. LVGL scorre fino al punto di aggancio con un'animazione.

Scroll one

La funzione "scroll one" [scorrimento singolo] indica a LVGL di consentire lo scorrimento di un solo elemento figlio agganciabile alla volta. Ciò richiede di rendere gli elementi figlio agganciabili e di impostare l'allineamento dello snap dello scorrimento su un valore diverso da [LV_SCROLL_SNAP_NONE](#).

Questa funzionalità può essere abilitata tramite il flag [LV_OBJ_FLAG_SCROLL_ONE](#).

Scorrimento con focus

Si immagini che ci siano molti Widget in un gruppo che si trovano su un Widget scorrevole. Premendo il pulsante "Tab" si sposta il focus sul Widget successivo, ma potrebbe trovarsi al di fuori dell'area visibile del Widget scorrevole. Se la funzione "scroll on focus" è abilitata, LVGL scorrerà automaticamente i Widget per visualizzare il Widget figlio con focus. Lo scorrimento avviene in modo ricorsivo, quindi anche i Widget scorrevoli annidati vengono gestiti correttamente. Il Widget verrà visualizzato anche se si trova su una pagina diversa di una visualizzazione a schede.

5.12.3 Scorrimento Programmatico

Le seguenti funzioni API consentono lo scorrimento programmatico dei Widget:

- `lv_obj_scroll_by(widget, x, y, LV_ANIM_ON/OFF)` scorre in base ai valori x e y
- `lv_obj_scroll_to(widget, x, y, LV_ANIM_ON/OFF)` scorre per portare la coordinata specificata nell'angolo in alto a sinistra
- `lv_obj_scroll_to_x(widget, x, LV_ANIM_ON/OFF)` scorre per portare la coordinata specificata sul lato sinistro
- `lv_obj_scroll_to_y(widget, y, LV_ANIM_ON/OFF)` scorre per portare la coordinata specificata sul lato superiore

Di tanto in tanto potrebbe essere necessario recuperare la *posizione di scorrimento* di un widget scorrevole, per ripristinarla in seguito o per visualizzarne dinamicamente alcuni elementi in base alla loro posizione corrente dello scorrimento. Ecco un esempio per illustrare come combinare l'evento di scorrimento e memorizzare la posizione di scorrimento superiore.

```
static int scroll_value = 0;

static void store_scroll_top_value_event_cb(lv_event_t* e) {
    lv_obj_t * scr = lv_event_get_target(e);
    scroll_value = lv_obj_get_scroll_top(scr);
    printf("%d pixels are scrolled above top edge of display.\n", scroll_value);
}

lv_obj_t * scr = lv_obj_create(NULL);
lv_obj_add_event_cb(scr, store_scroll_top_value_event_cb, LV_EVENT_SCROLL, NULL);
```

Le coordinate di scorrimento possono essere recuperate da assi diversi con queste funzioni:

- `lv_obj_get_scroll_x(widget)` Pixel che hanno superato il bordo sinistro della finestra di visualizzazione del Widget.
- `lv_obj_get_scroll_y(widget)` Pixel che hanno superato la parte superiore della finestra di visualizzazione del Widget.

- `lv_obj_get_scroll_top(widget)` Identico a `lv_obj_get_scroll_y(widget)`
 - `lv_obj_get_scroll_bottom(widget)` Pixel che hanno superato la parte inferiore della finestra di visualizzazione del Widget.
 - `lv_obj_get_scroll_left(widget)` Identico a `lv_obj_get_scroll_x(widget)`.
 - `lv_obj_get_scroll_right(widget)` Pixel scorsi oltre il bordo destro della finestra di visualizzazione del Widget.

L'impostazione della posizione di scorrimento può essere effettuata con queste funzioni:

- `lv_obj_scroll_by(widget, dx, dy, anim_enable)` Scorrimento di un numero specificato di pixel.
 - `lv_obj_scroll_by_bounded(widget, dx, dy, animation_enable)` Scorre di un numero specificato di pixel.
 - `lv_obj_scroll_to(widget, x, y, animation_enable)` Scorre fino alla coordinata specificata sul Widget.
 - `lv_obj_scroll_to_x(widget, x, animation_enable)` Scorre fino alla coordinata X sul Widget.
 - `lv_obj_scroll_to_y(widget, y, animation_enable)` Scorre fino alla coordinata Y sul Widget.
 - `lv_obj_scroll_to_view(widget, animation_enable)` Scorre il Widget padre di obj finché obj non diventa visibile.
 - `lv_obj_scroll_to_view_recursive(widget, animation_enable)` Scorre ricorsivamente i widget padre di obj finché obj non diventa visibile.

5.12.4 Dimensione Automatica

La "Self size" [dimensione automatica] è una proprietà di un Widget. Normalmente, l'utente non dovrebbe utilizzare questo parametro, ma se si creasse un widget personalizzato potrebbe essere utile.

In breve, il self size stabilisce la dimensione del contenuto di un Widget. Per capirlo meglio, prendiamo l'esempio di una tabella. Supponiamo che abbia 10 righe, ciascuna con un'altezza di 50 px. Quindi l'altezza totale del contenuto è di 500 px. In altre parole, la "self height" è di 500 px. Se l'utente imposta solo un'altezza di 200 px per la tabella, LVGL vedrà che la dimensione automatica è maggiore e renderà la tabella scorrevole.

Ciò significa che non solo i figli possono rendere scorrevole un Widget, ma anche una "self size" maggiore.

LVGL utilizza l'evento `LV_EVENT_GET_SELF_SIZE` per ottenere la dimensione automatica di un Widget. Ecco un esempio per vedere come gestire l'evento:

```
if(event_code == LV_EVENT_GET_SELF_SIZE) {
    lv_point_t * p = lv_event_get_param(e);

    /* If x or y < 0 then it doesn't need to be calculated now. */
    if(p->x >= 0) {
        p->x = 200; /* Set or calculate self width. */
    }

    if(p->y >= 0) {
        p->y = 50; /* Set or calculate self height. */
    }
}
```

5.12.5 Esempi

Scrolling nidificato

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

static lv_obj_t * panel;
static lv_obj_t * save_button;
static lv_obj_t * restore_button;
static int saved_scroll_x;
static int saved_scroll_y;

static void scroll_update_cb(lv_event_t * e);
static void button_event_cb(lv_event_t * e);

static void scroll_update_cb(lv_event_t * e)
{
    LV_UNUSED(e);

    LV_LOG("scroll info: x:%d LV_PRId32", y:%d LV_PRId32, top:%d LV_PRId32, 'bottom:%d LV_PRId32", left:%d LV_PRId32, right:%d LV_PRId32"\n",
        e->param.scroll.x, e->param.scroll.y, e->param.scroll.top, e->param.scroll.bottom, e->param.scroll.left, e->param.scroll.right);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

    lv_obj_get_scroll_x(panel),
    lv_obj_get_scroll_y(panel),
    lv_obj_get_scroll_top(panel),
    lv_obj_get_scroll_bottom(panel),
    lv_obj_get_scroll_left(panel),
    lv_obj_get_scroll_right(panel)
);
}

static void button_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);

    if(obj == save_button) {
        saved_scroll_x = lv_obj_get_scroll_x(panel);
        saved_scroll_y = lv_obj_get_scroll_y(panel);
    }
    else {
        lv_obj_scroll_to(panel, saved_scroll_x, saved_scroll_y, LV_ANIM_ON);
    }
}

/** Demonstrate how scrolling appears automatically
 */
void lv_example_scroll_1(void)
{
    /*Create an object with the new style*/
    lv_obj_t * scr;
    scr = lv_screen_active();
    panel = lv_obj_create(scr);
    lv_obj_set_size(panel, 200, 200);
    lv_obj_align(panel, LV_ALIGN_CENTER, 44, 0);

    lv_obj_t * child;
    lv_obj_t * label;

    child = lv_obj_create(panel);
    lv_obj_set_pos(child, 0, 0);
    lv_obj_set_size(child, 70, 70);
    label = lv_label_create(child);
    lv_label_set_text(label, "Zero");
    lv_obj_center(label);

    child = lv_obj_create(panel);
    lv_obj_set_pos(child, 160, 80);
    lv_obj_set_size(child, 80, 80);

    lv_obj_t * child2 = lv_button_create(child);
    lv_obj_set_size(child2, 100, 50);

    label = lv_label_create(child2);
    lv_label_set_text(label, "Right");
    lv_obj_center(label);

    child = lv_obj_create(panel);
    lv_obj_set_pos(child, 40, 160);
    lv_obj_set_size(child, 100, 70);
    label = lv_label_create(child);
    lv_label_set_text(label, "Bottom");
    lv_obj_center(label);

    /* When LV_OBJ_FLAG_SCROLL_ELASTIC is cleared, scrolling does not go past edge boundaries. */
    /* lv_obj_remove_flag(panel, LV_OBJ_FLAG_SCROLL_ELASTIC); */

    /* Call 'scroll_update_cb' while panel is being scrolled. */
    lv_obj_add_event_cb(panel, scroll_update_cb, LV_EVENT_SCROLL, NULL);

    /* Set up buttons that save and restore scroll position. */
    save_button = lv_button_create(scr);
    restore_button = lv_button_create(scr);
    lv_obj_t * lbl;
    lbl = lv_label_create(save_button);
    lv_label_set_text_static(lbl, "Save");
    lbl = lv_label_create(restore_button);
    lv_label_set_text_static(lbl, "Restore");
    lv_obj_align_to(save_button, panel, LV_ALIGN_OUT_LEFT_MID, -10, -20);
    lv_obj_align_to(restore_button, panel, LV_ALIGN_OUT_LEFT_MID, -10, 20);
    lv_obj_add_event_cb(save_button, button_event_cb, LV_EVENT_CLICKED, NULL);
    lv_obj_add_event_cb(restore_button, button_event_cb, LV_EVENT_CLICKED, NULL);
}
#endif

```

Aggancio

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static void sw_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * sw = lv_event_get_target_obj(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        lv_obj_t * list = (lv_obj_t *) lv_event_get_user_data(e);

        if(lv_obj_has_state(sw, LV_STATE_CHECKED)) lv_obj_add_flag(list, LV_OBJ_FLAG_SCROLL_ONE);
        else lv_obj_remove_flag(list, LV_OBJ_FLAG_SCROLL_ONE);
    }
}

/** Show an example to scroll snap
 */
void lv_example_scroll_2(void)
{
    lv_obj_t * panel = lv_obj_create(lv_screen_active());

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_size(panel, 280, 120);
lv_obj_set_scroll_snap_x(panel, LV_SCROLL_SNAP_CENTER);
lv_obj_set_flex_flow(panel, LV_FLEX_FLOW_ROW);
lv_obj_align(panel, LV_ALIGN_CENTER, 0, 20);

uint32_t i;
for(i = 0; i < 10; i++) {
    lv_obj_t * btn = lv_button_create(panel);
    lv_obj_set_size(btn, 150, lv_pct(100));

    lv_obj_t * label = lv_label_create(btn);
    if(i == 3) {
        lv_label_set_text_fmt(label, "Panel %" LV_PRIu32"\nno snap", i);
        lv_obj_remove_flag(btn, LV_OBJ_FLAG_SNAPPABLE);
    } else {
        lv_label_set_text_fmt(label, "Panel %" LV_PRIu32, i);
    }

    lv_obj_center(label);
}
lv_obj_update_snap(panel, LV_ANIM_ON);

#if LV_USE_SWITCH
/*Switch between "One scroll" and "Normal scroll" mode*/
lv_obj_t * sw = lv_switch_create(lv_screen_active());
lv_obj_align(sw, LV_ALIGN_TOP_RIGHT, -20, 10);
lv_obj_add_event_cb(sw, sw_event_cb, LV_EVENT_ALL, panel);
lv_obj_t * label = lv_label_create(lv_screen_active());
lv_label_set_text(label, "One scroll");
lv_obj_align_to(label, sw, LV_ALIGN_OUT_BOTTOM_MID, 0, 5);
#endif
}
#endif

```

Pulsante mobile

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LIST

static uint32_t btn_cnt = 1;

static void float_button_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * float_btn = lv_event_get_target_obj(e);

    if(code == LV_EVENT_CLICKED) {
        lv_obj_t * list = (lv_obj_t *) lv_event_get_user_data(e);
        char buf[32];
        lv_snprintf(buf, sizeof(buf), "Track %d", (int)btn_cnt);
        lv_obj_t * list_btn = lv_list_add_button(list, LV_SYMBOL_AUDIO, buf);
        btn_cnt++;

        /* Move the button to the foreground*/
        lv_obj_move_to_index(float_btn, -1);

        lv_obj_scroll_to_view(list_btn, LV_ANIM_ON);
    }
}

/**
 * Create a list with a floating button
 */
void lv_example_scroll_3(void)
{
    lv_obj_t * list = lv_list_create(lv_screen_active());
    lv_obj_set_size(list, 280, 220);
    lv_obj_center(list);

    for(btn_cnt = 1; btn_cnt <= 2; btn_cnt++) {
        char buf[32];
        lv_snprintf(buf, sizeof(buf), "Track %d", (int)btn_cnt);
        lv_list_add_button(list, LV_SYMBOL_AUDIO, buf);
    }

    lv_obj_t * float_btn = lv_button_create(list);
    lv_obj_set_size(float_btn, 50, 50);
    lv_obj_add_flag(float_btn, LV_OBJ_FLAG_FLOATING);
    lv_obj_align(float_btn, LV_ALIGN_BOTTOM_RIGHT, 0, -lv_obj_get_style_pad_right(list, LV_PART_MAIN));
    lv_obj_add_event_cb(float_btn, float_button_event_cb, LV_EVENT_ALL, list);
    lv_obj_set_style_radius(float_btn, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_bg_image_src(float_btn, LV_SYMBOL_PLUS, 0);
    lv_obj_set_style_text_font(float_btn, lv_theme_get_font_large(float_btn), 0);
}
#endif

```

Stile delle barre di scorrimento

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LIST

/***
 * Styling the scrollbars
 */
void lv_example_scroll_4(void)
{
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_set_size(obj, 200, 100);
    lv_obj_center(obj);

    lv_obj_t * label = lv_label_create(obj);

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_label_set_text(label,
    "Lorem ipsum dolor sit amet, consectetur adipiscing elit.\n"
    "Etiam dictum, tortor vestibulum lacinia laoreet, mi neque consectetur neque, vel mattis odio dolor egestas ligula. \n"
    "Sed vestibulum sapien nulla, id convallis ex porttitor nec. \n"
    "Duis et massa eu libero accumsan faucibus a in arcu. \n"
    "Ut pulvinar odio lorem, vel tempus turpis condimentum quis. Nam consectetur condimentum sem in auctor. \n"
    "Sed nisl augue, venenatis in blandit et, gravida ac tortor. \n"
    "Etiam dapibus elementum suscipit. \n"
    "Proin mollis sollicitudin convallis. \n"
    "Integer dapibus tempus arcu nec viverra. \n"
    "Donec molestie nulla enim, eu interdum velit placerat quis. \n"
    "Donec id efficitur risus, at molestie turpis. \n"
    "Suspendisse vestibulum consectetur nunc ut commodo. \n"
    "Fusce molestie rhoncus nisi sit amet tincidunt. \n"
    "Suspendisse a nunc ut magna ornare volutpat.");

/*Remove the style of scrollbar to have clean start*/
lv_obj_remove_style(obj, NULL, LV_PART_SCROLLBAR | LV_STATE_ANY);

/*Create a transition the animate the some properties on state change*/
static const lv_style_prop_t props[] = {LV_STYLE_BG_OPA, LV_STYLE_WIDTH, 0};
static lv_style_transition_dsc_t trans;
lv_style_transition_dsc_init(&trans, props, lv_anim_path_linear, 200, 0, NULL);

/*Create a style for the scrollbars*/
static lv_style_t style;
lv_style_init(&style);
lv_style_set_width(&style, 4); /*Width of the scrollbar*/
lv_style_set_length(&style, 20); /*Length of the scrollbar*/
lv_style_set_pad_right(&style, 5); /*Space from the parallel side*/
lv_style_set_pad_top(&style, 5); /*Space from the perpendicular side*/

lv_style_set_radius(&style, 2);
lv_style_set_bg_opa(&style, LV_OPA_70);
lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_BLUE));
lv_style_set_border_color(&style, lv_palette_darken(LV_PALETTE_BLUE, 3));
lv_style_set_border_width(&style, 2);
lv_style_set_shadow_width(&style, 8);
lv_style_set_shadow_spread(&style, 2);
lv_style_set_shadow_color(&style, lv_palette_darken(LV_PALETTE_BLUE, 1));

lv_style_set_transition(&style, &trans);

/*Make the scrollbars wider and use 100% opacity when scrolled*/
static lv_style_t style_scrolled;
lv_style_init(&style_scrolled);
lv_style_set_width(&style_scrolled, 8);
lv_style_set_bg_opa(&style_scrolled, LV_OPA_COVER);

lv_obj_add_style(obj, &style, LV_PART_SCROLLBAR);
lv_obj_add_style(obj, &style_scrolled, LV_PART_SCROLLBAR | LV_STATE_SCROLLED);
}

#endif

```

Scorrimento da destra a sinistra

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_FONT_DEJAVU_16_PERSIAN_HEBREW

/***
 * Scrolling with Right To Left base direction
 */
void lv_example_scroll_5(void)
{
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_set_style_base_dir(obj, LV_BASE_DIRRTL, 0);
    lv_obj_set_size(obj, 200, 100);
    lv_obj_center(obj);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label,
        "I) دیجیتال و مادون موجات رمی (ROM) یعنی رام که در پردازنده (CPU) قرار دارد که از همان دستگاه استفاده می شود. CPU دو کار را انجام می دهد: ۱) ارسال دستورات (Serial Port) به دستگاه های خارجی از سیستم (LCD، موتورها، دیسکها، ...). ۲) دریافت داده هایی که از دستگاه های خارجی آمدند. این داده هایی ممکن است از سیستم خارجی باشند (مثلاً داده هایی که از یک ماژول GPS آمدند) یا در سیستم داخلی باشند (مثلاً داده هایی که از یک ماژول GPS آمدند).");
    lv_obj_set_width(label, 400);
    lv_obj_set_style_text_font(label, &lv_font_dejavu_16_persian_hebrew, 0);
}

#endif

```

Traslazione durante lo scorrimento

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static void scroll_event_cb(lv_event_t * e)
{
    lv_obj_t * cont = lv_event_get_target_obj(e);

    lv_area_t cont_a;
    lv_obj_get_coords(cont, &cont_a);
    int32_t cont_y_center = cont_a.y1 + lv_area_get_height(&cont_a) / 2;

    int32_t r = lv_obj_get_height(cont) * 7 / 10;
    int32_t i;
    int32_t child_cnt = (int32_t)lv_obj_get_child_count(cont);
    for(i = 0; i < child_cnt; i++) {
        lv_obj_t * child = lv_obj_get_child(cont, i);
        lv_area_t child_a;
        lv_obj_get_coords(child, &child_a);

        int32_t child_y_center = child_a.y1 + lv_area_get_height(&child_a) / 2;

```

(continues on next page)

(continua dalla pagina precedente)

```

int32_t diff_y = child_y_center - cont_y_center;
diff_y = LV_ABS(diff_y);

/*Get the x of diff_y on a circle.*/
int32_t x;
/*If diff_y is out of the circle use the last point of the circle (the radius)*/
if(diff_y >= r) {
    x = r;
}
else {
    /*Use Pythagoras theorem to get x from radius and y*/
    uint32_t x_sqr = r * r - diff_y * diff_y;
    lv_sqrt_res_t res;
    lv_sqrt(x_sqr, &res); /*Use lvgl's built in sqrt root function*/
    x = r - res.i;
}

/*Translate the item by the calculated X coordinate*/
lv_obj_set_style_translate_x(child, x, 0);

/*Use some opacity with larger translations*/
lv_opa_t opa = (lv_opa_t)lv_map(x, 0, r, LV_OPA_TRANSP, LV_OPA_COVER);
lv_obj_set_style_opa(child, LV_OPA_COVER - opa, 0);
}

/** Translate the object as they scroll */
void lv_example_scroll_6(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 200, 200);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_COLUMN);
    lv_obj_add_event_cb(cont, scroll_event_cb, LV_EVENT_SCROLL, NULL);
    lv_obj_set_style_radius(cont, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_clip_corner(cont, true, 0);
    lv_obj_set_scroll_dir(cont, LV_DIR_VER);
    lv_obj_set_scroll_snap_y(cont, LV_SCROLL_SNAP_CENTER);
    lv_obj_set_scrollbar_mode(cont, LV_SCROLLBAR_MODE_OFF);

    uint32_t i;
    for(i = 0; i < 20; i++) {
        lv_obj_t * btn = lv_button_create(cont);
        lv_obj_set_width(btn, lv_pct(100));

        lv_obj_t * label = lv_label_create(btn);
        lv_label_set_text_fmt(label, "Button %" LV_PRIu32, i);
    }

    /*Update the buttons position manually for first*/
    lv_obj_send_event(cont, LV_EVENT_SCROLL, NULL);

    /*Be sure the fist button is in the middle*/
    lv_obj_scroll_to_view(lv_obj_get_child(cont, 0), LV_ANIM_OFF);
}

#endif

```

Scrolling infinito

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static lv_obj_t * high_label;
static lv_obj_t * low_label;
static int32_t top_num;
static int32_t bottom_num;
static bool update_scroll_running = false;

static lv_obj_t * load_item(lv_obj_t * parent, int32_t num)
{
    lv_obj_t * obj = (lv_obj_t *) lv_obj_create(parent);
    lv_obj_set_size(obj, LV_PCT(100), LV_SIZE_CONTENT);
    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text_fmt(label, "%" PRId32, num);
    return obj;
}

static void update_scroll(lv_obj_t * obj)
{
    /* do not re-enter this function when `lv_obj_scroll_by` triggers this callback again.
     */
    if(update_scroll_running) return;
    update_scroll_running = true;

    int32_t top_num_original = top_num;
    int32_t bottom_num_original = bottom_num;

    /* load items we're getting close to */
    while(bottom_num > -30 && lv_obj_get_scroll_bottom(obj) < 200) {
        bottom_num -= 1;
        load_item(obj, bottom_num);
        lv_obj_update_layout(obj);
        LV_LOG_USER("loaded bottom num: %" PRId32, bottom_num);
    }

    while(top_num < 30 && lv_obj_get_scroll_top(obj) < 200) {
        top_num += 1;
        int32_t bottom_before = lv_obj_get_scroll_bottom(obj);
        lv_obj_t * new_item = load_item(obj, top_num);
        lv_obj_move_to_index(new_item, 0);
        lv_obj_update_layout(obj);
        int32_t bottom_after = lv_obj_get_scroll_bottom(obj);
        lv_obj_scroll_by(obj, 0, bottom_before - bottom_after, LV_ANIM_OFF);
        LV_LOG_USER("loaded top num: %" PRId32, top_num);
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

}

/* delete far-away items */
while(lv_obj_get_scroll_bottom(obj) > 600) {
    bottom_num += 1;
    lv_obj_t * child = lv_obj_get_child(obj, -1);
    lv_obj_delete(child);
    lv_obj_update_layout(obj);
    LV_LOG_USER("deleted bottom num: %" PRId32, bottom_num);
}

while(lv_obj_get_scroll_top(obj) > 600) {
    top_num -= 1;
    int32_t bottom_before = lv_obj_get_scroll_bottom(obj);
    lv_obj_t * child = lv_obj_get_child(obj, 0);
    lv_obj_delete(child);
    lv_obj_update_layout(obj);
    int32_t bottom_after = lv_obj_get_scroll_bottom(obj);
    lv_obj_scroll_by(obj, 0, bottom_before - bottom_after, LV_ANIM_OFF);
    LV_LOG_USER("deleted top num: %" PRId32, top_num);
}

if(top_num != top_num_original) {
    lv_label_set_text_fmt(high_label, "current largest\nloaded value:\n%" PRId32, top_num);
}
if(bottom_num != bottom_num_original) {
    lv_label_set_text_fmt(low_label, "current smallest\nloaded value:\n%" PRId32, bottom_num);
}

update_scroll_running = false;
}

static void scroll_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    update_scroll(obj);
}

static void checkbox_cb(lv_event_t * e)
{
    lv_obj_t * checkbox = lv_event_get_target_obj(e);
    lv_obj_t * obj = (lv_obj_t *) lv_event_get_user_data(e);
    bool checked = lv_obj_has_state(checkbox, LV_STATE_CHECKED);
    lv_obj_set_style_opa(obj, checked ? LV_OPA_COVER : LV_OPA_TRANS, LV_PART_SCROLLBAR);
}

/**
 * Dynamically load widgets while scrolling
 */
void lv_example_scroll_7(void)
{
    lv_obj_t * scr = lv_screen_active();
    lv_obj_t * obj = lv_obj_create(scr);
    lv_obj_set_size(obj, 160, 220);
    lv_obj_align(obj, LV_ALIGN_RIGHT_MID, -10, 0);
    lv_obj_set_flex_flow(obj, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_style_opa(obj, LV_OPA_TRANS, LV_PART_SCROLLBAR);

    high_label = lv_label_create(scr);
    lv_label_set_text_static(high_label, "current largest\nloaded value:");
    lv_obj_align(high_label, LV_ALIGN_TOP_LEFT, 10, 10);

    lv_obj_t * checkbox = lv_checkbox_create(scr);
    lv_checkbox_set_text_static(checkbox, "show\nscrollbar");
    lv_obj_align(checkbox, LV_ALIGN_LEFT_MID, 10, 0);
    lv_obj_add_event_cb(checkbox, checkbox_cb, LV_EVENT_VALUE_CHANGED, obj);

    low_label = lv_label_create(scr);
    lv_label_set_text_static(low_label, "current smallest\nloaded value:");
    lv_obj_align(low_label, LV_ALIGN_BOTTOM_LEFT, 10, -10);

    load_item(obj, 3);
    /* These counters hold the highest/lowest number currently loaded. */
    top_num = 3;
    bottom_num = 3;

    lv_obj_update_layout(obj);
    update_scroll(obj);
    lv_obj_add_event_cb(obj, scroll_cb, LV_EVENT_SCROLL, NULL);
}

#endif

```

Scrolling circolare

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

/*
 * Circular list implementation based on:
 * Reference: https://blog.csdn.net/TOW4321/article/details/145434819
 */

/* Applies to both width (for row items) and height (for column items) */
#define ITEM_SIZE     80

/* Function to calculate the total content width of the container */
static int32_t get_content_width(lv_obj_t * cont)
{
    int32_t w = 0;
    int32_t pad_column = lv_obj_get_style_pad_column(cont, LV_PART_MAIN);
    int32_t total = (int32_t)lv_obj_get_child_count(cont);

    for(int32_t i = 0; i < total; i++) {
        w += lv_obj_get_width(lv_obj_get_child(cont, i));
        if(i < total - 1) w += pad_column;
    }
    return w + lv_obj_get_style_pad_left(cont, LV_PART_MAIN)
}

```

(continues on next page)

(continua dalla pagina precedente)

```

        + lv_obj_get_style_pad_right(cont, LV_PART_MAIN);
}

/* Scroll event callback for row layout */
static void cont_row_scroll_event_cb(lv_event_t * e)
{
    static bool is_adjusting = false;
    lv_obj_t * cont = lv_event_get_current_target_obj(e);

    if(!is_adjusting) {
        is_adjusting = true;
        int32_t scroll_x = lv_obj_get_scroll_x(cont);
        int32_t cont_w = lv_obj_get_width(cont);
        int32_t content_w = get_content_width(cont);

        /* Use ITEM_SIZE as horizontal item width */
        const int32_t item_width = ITEM_SIZE;

        if(scroll_x <= 0) {
            lv_obj_t * last_child = lv_obj_get_child(cont, (int32_t)(lv_obj_get_child_count(cont) - 1));
            lv_obj_move_to_index(last_child, 0);
            lv_obj_scroll_to_x(cont, scroll_x + item_width, LV_ANIM_OFF);
        }
        else if(scroll_x > content_w - cont_w) {
            lv_obj_t * first_child = lv_obj_get_child(cont, 0);
            lv_obj_move_to_index(first_child, (int32_t)(lv_obj_get_child_count(cont) - 1));
            lv_obj_scroll_to_x(cont, scroll_x - item_width, LV_ANIM_OFF);
        }
        is_adjusting = false;
    }
}

/* Function to calculate the total content height of the container */
static int32_t get_content_height(lv_obj_t * cont)
{
    int32_t h = 0;
    int32_t pad_row = lv_obj_get_style_pad_row(cont, LV_PART_MAIN);
    int32_t total = (int32_t)lv_obj_get_child_count(cont);

    for(int i = 0; i < total; i++) {
        h += lv_obj_get_height(lv_obj_get_child(cont, i));
        if(i < total - 1) h += pad_row;
    }
    return h + lv_obj_get_style_pad_top(cont, LV_PART_MAIN)
        + lv_obj_get_style_pad_bottom(cont, LV_PART_MAIN);
}

/* Scroll event callback for column layout */
static void cont_col_scroll_event_cb(lv_event_t * e)
{
    static bool is_adjusting = false;
    lv_obj_t * cont = lv_event_get_current_target_obj(e);

    if(!is_adjusting) {
        is_adjusting = true;
        int32_t scroll_y = lv_obj_get_scroll_y(cont);
        int32_t cont_h = lv_obj_get_height(cont);
        int32_t content_h = get_content_height(cont);

        /* Use ITEM_SIZE as vertical item height */
        const int32_t item_height = ITEM_SIZE;

        if(scroll_y <= 0) {
            lv_obj_t * last_child = lv_obj_get_child(cont, (int32_t)(lv_obj_get_child_count(cont) - 1));
            lv_obj_move_to_index(last_child, 0);
            lv_obj_scroll_to_y(cont, scroll_y + item_height, LV_ANIM_OFF);
        }
        else if(scroll_y > content_h - cont_h) {
            lv_obj_t * first_child = lv_obj_get_child(cont, 0);
            lv_obj_move_to_index(first_child, (int32_t)(lv_obj_get_child_count(cont) - 1));
            lv_obj_scroll_to_y(cont, scroll_y - item_height, LV_ANIM_OFF);
        }
        is_adjusting = false;
    }
}

void lv_example_scroll_8(void)
{
    /* Create a scroll container with ROW flex direction */
    lv_obj_t * cont_row = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont_row, 300, 75);
    lv_obj_align(cont_row, LV_ALIGN_TOP_MID, 0, 5);
    lv_obj_set_flex_flow(cont_row, LV_FLEX_FLOW_ROW);
    lv_obj_add_event_cb(cont_row, cont_row_scroll_event_cb, LV_EVENT_SCROLL, NULL);
    /* Hide scrollbar visuals */
    lv_obj_set_scrollbar_mode(cont_row, LV_SCROLLBAR_MODE_OFF);

    /* Create a scroll container with COLUMN flex direction */
    lv_obj_t * cont_col = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont_col, 200, 150);
    lv_obj_align_to(cont_col, cont_row, LV_ALIGN_OUT_BOTTOM_MID, 0, 5);
    lv_obj_set_flex_flow(cont_col, LV_FLEX_FLOW_COLUMN);
    lv_obj_add_event_cb(cont_col, cont_col_scroll_event_cb, LV_EVENT_SCROLL, NULL);
    /* Hide scrollbar visuals */
    lv_obj_set_scrollbar_mode(cont_col, LV_SCROLLBAR_MODE_OFF);

    /* If the number of items is less than 3, scrolling may not be needed or may cause unexpected behavior. */
    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_obj_t * obj;
        lv_obj_t * label;

        /*Add items to the row*/
        obj = lv_button_create(cont_row);
        lv_obj_set_size(obj, ITEM_SIZE, LV_PCT(100));

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item %" LV_PRIu32, i + 1);
        lv_obj_center(label);

        /*Add items to the column*/
        obj = lv_button_create(cont_col);
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_size(obj, LV_PCT(100), ITEM_SIZE);

label = lv_label_create(obj);
lv_label_set_text_fmt(label, "Item %" LV_PRIu32, i + 1);
lv_obj_center(label);
}

#endif

```

Proprietà dello Scrolling

```

/**
 * @file lv_example_scroll_9.c
 */

#include "lv_example_scroll.h"

#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static lv_obj_t * switch_create(lv_obj_t * parent, const char * title, lv_obj_flag_t flag, bool en);
static void generic_switch_event_cb(lv_event_t * e);

static lv_obj_t * list;

void lv_example_scroll_9(void)
{
    lv_obj_t * panel = lv_obj_create(lv_screen_active());
    lv_obj_set_style_shadow_width(panel, 16, 0);
    lv_obj_set_style_shadow_offset_y(panel, 8, 0);
    lv_obj_set_style_shadow_offset_x(panel, 4, 0);
    lv_obj_set_style_shadow_opa(panel, LV_OPA_40, 0);

    lv_obj_set_size(panel, lv_pct(70), lv_pct(90));
    lv_obj_center(panel);
    lv_obj_set_flex_flow(panel, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(panel, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    list = lv_list_create(panel);
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image1.png");
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image2.png");
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image3.png");
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image4.png");
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image5.png");
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image6.png");
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image7.png");
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image8.png");
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image9.png");
    lv_list_add_button(list, LV_SYMBOL_IMAGE, "Image10.png");

    switch_create(panel, "Scollable", LV_OBJ_FLAG_SCROLLABLE, true);
    switch_create(panel, "Scroll chain", LV_OBJ_FLAG_SCROLL_CHAIN, true);
    switch_create(panel, "Elastic scroll", LV_OBJ_FLAG_SCROLL_ELASTIC, true);
    switch_create(panel, "Add scroll momentum", LV_OBJ_FLAG_SCROLL_MOMENTUM, true);

    /*Show the switches first*/
    lv_obj_move_to_index(list, -1);
}

static lv_obj_t * switch_create(lv_obj_t * parent, const char * title, lv_obj_flag_t flag, bool en)
{
    lv_obj_t * cont = lv_obj_create(parent);
    lv_obj_remove_style_all(cont);
    lv_obj_set_size(cont, lv_pct(100), LV_SIZE_CONTENT);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW);
    lv_obj_set_flex_align(cont, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_START);

    lv_obj_t * label = lv_label_create(cont);
    lv_label_set_text(label, title);
    lv_obj_set_flex_grow(label, 1);

    lv_obj_t * sw = lv_switch_create(cont);
    lv_obj_add_event_cb(sw, generic_switch_event_cb, LV_EVENT_VALUE_CHANGED, (void *)((lv_uintptr_t)flag));
    if(en) {
        lv_obj_add_state(sw, LV_STATE_CHECKED);
        lv_obj_add_flag(list, flag);
    }
    else {
        lv_obj_remove_state(sw, LV_STATE_CHECKED);
        lv_obj_remove_flag(list, flag);
    }

    return cont;
}

static void generic_switch_event_cb(lv_event_t * e)
{
    lv_obj_t * sw = lv_event_get_target_obj(e);
    lv_obj_flag_t flag = ((lv_obj_flag_t)((lv_uintptr_t)lv_event_get_user_data(e)));

    if(lv_obj_has_state(sw, LV_STATE_CHECKED)) lv_obj_add_flag(list, flag);
    else lv_obj_remove_flag(list, flag);
}
#endif

```

5.12.6 API

CAPITOLO 6

Tutti i Widget

6.1 Widget Base (`lv_obj`)

Il più fondamentale di tutti è il Widget Base, su cui si basano tutti gli altri widget. Da una prospettiva object-oriented, si può pensare al Widget Base come alla classe widget da cui ereditano tutti gli altri.

Con questo meccanismo, tutti i widget possiedono le funzionalità del Widget Base. Pertanto, le funzioni e le funzionalità del Widget Base possono essere utilizzate anche con gli altri Widget. Ad esempio: `lv_obj_set_width(slider, 100)`.

Sebbene si tratti solo di un semplice rettangolo, il Widget Base è quello più potente, poiché supporta tutte le funzionalità che altri widget possono utilizzare.

Il Widget Base può avere:

- *Figli*
- *Posizione e Dimensione*
- *Layout*
- *Stili*
- *Eventi*
- *Data binding*
- *Flag*
- *Parti e Stati*
- E molto altro

Un widget Slider, ad esempio, utilizza queste funzionalità per realizzare un comportamento simile a quello di uno slider, ma si basa sullo stesso stile, evento, layout e altri concetti supportati dal Widget Base.

Tutte queste funzionalità comuni e versatili vengono presentate nel capitolo [Funzionalità Comuni dei Widget](#).

In termini HTML, lo si pensi come un `<div>`, un elemento costitutivo molto versatile che, con un po' di CSS e JS, può diventare praticamente qualsiasi cosa.

6.2 Texture 3D (lv_3dtexture)

6.2.1 Panoramica

I widget texture 3D vengono utilizzati per incorporare la primitiva "texture" di una libreria grafica 3D esterna in un'interfaccia utente LVGL.

6.2.2 Parti e Stili

La dimensione deve essere impostata su quella effettiva della primitiva texture. Anche l'opacità può essere modificata.

6.2.3 Utilizzo

Richiede l'abilitazione di una "unità draw" in grado di eseguire il tipo di task `LV_DRAW_TASK_TYPE_3D` per il disegno.

OpenGL è il primo back-end grafico 3D supportato. È necessario abilitare quanto segue.

- `LV_USE_3DTEXTURE`
- `LV_USE_OPENGL`
- `LV_USE_DRAW_OPENGL`

Consultare la *Documentazione del driver LVGL OpenGL ES* per creare una finestra e una texture di visualizzazione.

Il repository `lv_example_3dtexture <https://github.com/lvgl/lv_example_3dtexture>` è una dimostrazione di come utilizzare il widget `3dtexture` per caricare modelli glTF con OpenGL e visualizzarli in LVGL.

```
lv_obj_t * tex = lv_3dtexture_create(parent);
/* Render something to the texture. You can replace it with your code. */
lv_3dtexture_id_t gltf_texture = render_gltf_model_to_opengl_texture(path, w, h, color);
lv_3dtexture_set_src(tex, gltf_texture);
lv_obj_set_size(tex, w, h);
lv_obj_set_style_opa(tex, opa, 0);
```

Il tipo effettivo di `lv_3dtexture_id_t` dipende dal back-end grafico 3D attivo. Con OpenGL, il tipo è `unsigned int`.

6.2.4 Eventi

La Texture 3D non ha una gestione speciale degli eventi.

6.2.5 Esempio

Vedere il repository `lv_example_3dtexture <https://github.com/lvgl/lv_example_3dtexture>`.

6.2.6 API

`lv_3dtexture.h`

`lv_3dtexture_private.h`

6.3 Immagine di Animazione (lv_animimg)

6.3.1 Panoramica

L'immagine di animazione è simile al normale Widget 'Image'. L'unica differenza è che invece di un'immagine sorgente, si imposta un array di più immagini sorgente che forniscono i "fotogrammi" in un'animazione.

È possibile specificare una durata e un numero di ripetizioni.

6.3.2 Parti e Stili

- `LV_PART_MAIN` Un rettangolo di background che utilizza *tipiche proprietà di stile del background* e l'immagine stessa che utilizza le proprietà di stile dell'immagine.

6.3.3 Utilizzo

Sorgenti delle immagini

Per impostare le sorgenti delle immagini animate, utilizzare `lv_animimg_set_src(animimg, dsc[], num)`.

Per impostare la sorgente delle immagini per la riproduzione al contrario dell'immagine animata, si usa `lv_animimg_set_src_reverse(animimg, dsc[], num)`.

Utilizzo dell'animazione interna

Per casi d'uso più avanzati, l'animazione utilizzata internamente dall'immagine può essere recuperata utilizzando `lv_animimg_get_anim(animimg)`. In questo modo, è possibile utilizzare le funzioni *Animation*, ad esempio, per sovrascrivere i valori dell'animazione utilizzando `lv_anim_set_values(anim, start, end)` o per impostare una callback sull'evento di completamento dell'animazione.

6.3.4 Eventi

Nessun evento speciale viene inviato dai Widget Animation-Image.

Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.3.5 Tasti

Nessun *Tasto* viene elaborato dai Widget Animation-Image.

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.3.6 Esempio

Immagine animata semplice

```
#include "../../lv_examples.h"
#if LV_USE_ANIMIMG && LV_BUILD_EXAMPLES
LV_IMAGE_DECLARE(animimg001);
LV_IMAGE_DECLARE(animimg002);
LV_IMAGE_DECLARE(animimg003);

static const lv_image_dsc_t * anim_imgs[3] = {
    &animimg001,
    &animimg002,
    &animimg003,
};

void lv_example_animimg_1(void)
{
    lv_obj_t * animimg0 = lv_animimg_create(lv_screen_active());
    lv_obj_center(animimg0);
    lv_animimg_set_src(animimg0, (const void **) anim_imgs, 3);
    lv_animimg_set_duration(animimg0, 1000);
    lv_animimg_set_repeat_count(animimg0, LV_ANIM_REPEAT_INFINITE);
    lv_animimg_start(animimg0);
}
#endif
```

6.3.7 API

`lv_animimage.h`

`lv_animimage_private.h`

6.4 Arco (lv_arc)

6.4.1 Panoramica

L'Arc è costituito da un arco sullo sfondo e da uno in primo piano. L'arco in primo piano (indicatore) può essere regolato tramite "touch".

6.4.2 Parti e Stili

- `LV_PART_MAIN` Disegna uno sfondo utilizzando le tipiche proprietà di stile del background e un arco utilizzando le proprietà di stile dell'Arc. Le dimensioni e la posizione dell'Arco rispetteranno le proprietà di stile `padding`.
- `LV_PART_INDICATOR` Disegna un altro arco utilizzando le proprietà di stile `Arc`. I suoi valori di padding vengono interpretati in relazione all'Arco sullo sfondo.
- `LV_PART_KNOB` Disegna una maniglia all'estremità dell'indicatore utilizzando tutte le proprietà di sfondo e i valori di padding. Con padding zero, la dimensione della manopola è uguale alla larghezza dell'indicatore. Un padding maggiore lo rende più grande, un padding minore lo rende più piccolo.

6.4.3 Utilizzo

Valore e intervallo

È possibile impostare un nuovo valore utilizzando `lv_arc_set_value(arc, new_value)`. Il valore viene interpretato in un "range" (valori minimo e massimo) che può essere modificato con `lv_arc_set_range(arc, min, max)`. L'intervallo di default è 0..100.

L'Arco indicatore viene disegnato sull'Arco della parte principale. Quindi, se il valore è impostato sul massimo, l'indicatore coprirà l'intero Arco "di sfondo". Per impostare l'angolo iniziale e finale dell'Arco sullo sfondo, utilizzare una di queste funzioni:

- `lv_arc_set_bg_start_angle(arc, angle)`
- `lv_arc_set_bg_end_angle(arc, angle)`
- `lv_arc_set_bg_angles(arc, start_angle, end_angle)`

Zero gradi si trova al centro a destra (ore 3) del Widget e i gradi aumentano in senso orario. I valori degli angoli devono essere compresi tra [0 e 360].

Rotazione

È possibile aggiungere un offset alla posizione a 0 gradi con `lv_arc_set_rotation(arc, deg)`.

Modalità

L'Arco può essere una delle seguenti modalità:

- `LV_ARC_MODE_NORMAL` L'arco indicatore viene disegnato in senso orario dal valore minimo al valore corrente.
- `LV_ARC_MODE_REVERSE` L'Arco indicatore viene disegnato in senso antiorario dal valore massimo al valore corrente.
- `LV_ARC_MODE_SYMMETRICAL` L'Arco indicatore viene disegnato dal punto centrale al valore corrente.

La modalità può essere impostata tramite `lv_arc_set_mode(arc, LV_ARC_MODE_...)` e non ha effetto finché l'angolo non viene impostato tramite `lv_arc_set_value()` o il valore dell'arco non viene modificato tramite l'input del puntatore (dito, mouse, ecc.).

Velocità di variazione

Quando il valore dell'Arco viene modificato tramite l'input del puntatore (dito, mouse, ecc.), la velocità di variazione è limitata in base al suo *change rate* [velocità di variazione]. La velocità di variazione è definita in gradi/secondo e può essere impostata con `lv_arc_set_change_rate(arc, rate)`

Offset della manopola

Modificando l'offset del "knob" [manopola] è possibile spostarne la posizione rispetto alla fine dell'Arco. L'offset della manopola può essere impostato con `lv_arc_set_knob_offset(arc, offset_angle)` e sarà visibile solo se lo è `LV_PART_KNOB`.

Impostazione dell'indicatore tramite programmazione

È possibile impostare l'angolo dell'indicatore direttamente con una qualsiasi di queste funzioni:

- `lv_arc_set_start_angle(arc, start_angle)`
- `lv_arc_set_end_angle(arc, end_angle)`
- `lv_arc_set_angles(arc, start_angle, end_angle)`

Quando vengono utilizzati, "value" e "mode" vengono ignorati.

In altre parole, le impostazioni di angolo e valore sono indipendenti. Si consiglia di utilizzare esclusivamente uno dei due metodi. Combinarli potrebbe causare comportamenti indesiderati.

Per rendere l'arco non regolabile, rimuovere lo stile della manopola e rendere il Widget non cliccabile:

```
lv_obj_remove_style(arc, NULL, LV_PART_KNOB);
lv_obj_remove_flag(arc, LV_OBJ_FLAG_CLICKABLE);
```

Area interattiva

Per default `LV_OBJ_FLAG_ADV_HITTEST` è disabilitato, il che significa che l'intera area dell'Arco è interattiva. Come di consueto, `lv_obj_set_ext_click_area()` può essere utilizzato per aumentare di un numero specificato di pixel l'area che risponderà all'input del puntatore (tocco, mouse, ecc.) all'esterno dell'Arco.

Se `LV_OBJ_FLAG_ADV_HITTEST` è abilitato, l'Arco sarà sensibile solo nell'intervallo tra gli angoli di sfondo iniziale e finale e sull'Arco stesso (non all'interno dell'Arco). In questo caso, `ext_click_area` fa sì che l'area sensibile si estenda sia verso l'interno che verso l'esterno. Inoltre, a ciascun angolo viene applicata una tolleranza di `lv_dpx(50)` pixel, estendendo l'intervallo di hit-test per tutta la lunghezza dell'Arco.

Posizionare un altro Widget sulla manopola

Un altro Widget può essere posizionato in base alla posizione corrente dell'Arco per seguirne il valore corrente (angolo). Per fare ciò, si usa `lv_arc_align_obj_to_angle(arc, widget_to_align, radius_offset)`.

Analogamente, `lv_arc_rotate_obj_to_angle(arc, widget_to_rotate, radius_offset)` può essere utilizzato per ruotare il Widget in base al valore corrente dell'Arco.

Un tipico caso d'uso è chiamare queste funzioni nell'evento `VALUE_CHANGED` dell'Arco.

Data binding

Per familiarizzare con "observer", "subject" e data binding in generale, visitare la pagina [Observer](#).

Questo metodo di sottoscrizione a un Subject intero influisce direttamente sul valore intero di un Widget Arc. Si noti che si tratta di un binding bidirezionale (Subject <==> Widget), quindi l'interazione diretta di un utente finale con il Widget Arc aggiorna il valore del Subject e viceversa.

Supporta soggetti interi e, quando `LV_USE_FLOAT` è abilitato, subject float.

- `lv_arc_bind_value(arc, &subject)`

6.4.4 Eventi

- `LV_EVENT_VALUE_CHANGED` viene inviato quando l'Arco viene premuto/trascinato su un nuovo valore.

Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.4.5 Tasti

- `LV_KEY_RIGHT/UP` Aumenta il valore di uno.
- `LV_KEY_LEFT/DOWN` Diminuisce il valore di uno.

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.4.6 Esempio

Arco semplice

```
#include "../../lv_examples.h"
#if LV_USE_ARC && LV_BUILD_EXAMPLES

static void value_changed_cb(lv_event_t * e);

void lv_example_arc_1(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());

    /*Create an Arc*/
    lv_obj_t * arc = lv_arc_create(lv_screen_active());
    lv_obj_set_size(arc, 150, 150);
    lv_arc_set_rotation(arc, 135);
    lv_arc_set_bg_angles(arc, 0, 270);
    lv_arc_set_value(arc, 10);
    lv_obj_center(arc);
    lv_obj_add_event_cb(arc, value_changed_cb, LV_EVENT_VALUE_CHANGED, label);

    /*Manually update the label for the first time*/
    lv_obj_send_event(arc, LV_EVENT_VALUE_CHANGED, NULL);
}

static void value_changed_cb(lv_event_t * e)
{
    lv_obj_t * arc = lv_event_get_target_obj(e);
    lv_obj_t * label = (lv_obj_t *)lv_event_get_user_data(e);

    lv_label_set_text_fmt(label, "%" LV_PRIId32 "%%", lv_arc_get_value(arc));

    /*Rotate the label to the current position of the arc*/
    lv_arc_rotate_obj_to_angle(arc, label, 25);
}

#endif
```

Loader con Arc

```
#include "../../lv_examples.h"
#if LV_USE_ARC && LV_BUILD_EXAMPLES

static void set_angle(void * obj, int32_t v)
{
    lv_arc_set_value((lv_obj_t *)obj, v);
}

/**
 * Create an arc which acts as a loader.
 */
void lv_example_arc_2(void)
{
    /*Create an Arc*/
    lv_obj_t * arc = lv_arc_create(lv_screen_active());
    lv_arc_set_rotation(arc, 270);
    lv_arc_set_bg_angles(arc, 0, 360);
    lv_obj_remove_style(arc, NULL, LV_PART_KNOB); /*Be sure the knob is not displayed*/
    lv_obj_remove_flag(arc, LV_OBJ_FLAG_CLICKABLE); /*To not allow adjusting by click*/
    lv_obj_center(arc);

    lv_anim_t a;
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_anim_init(&a);
lv_anim_set_var(&a, arc);
lv_anim_set_exec_cb(&a, set_angle);
lv_anim_set_duration(&a, 1000);
lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE); /*Just for the demo*/
lv_anim_set_repeat_delay(&a, 500);
lv_anim_set_values(&a, 0, 100);
lv_anim_start(&a);

}

#endif

```

Grafico a torta con sezioni cliccabili tramite Archi

```

#include "../../lv_examples.h"

#if LV_USE_ARC && LV_BUILD_EXAMPLES

#define CHART_SIZE 150
#define SLICE_OFFSET 20

typedef struct {
    int start_angle;
    int end_angle;
    int mid_angle;
    lv_point_t home;
    bool out;
} slice_info_t;

typedef struct {
    lv_obj_t * obj;
    int start_x;
    int start_y;
    int end_x;
    int end_y;
} slice_anim_data_t;

static float angle_accum = 0.0f;
static slice_info_t * active_info = NULL;
static lv_obj_t * active_arc = NULL;

static void anim_move_cb(void * var, int32_t v)
{
    slice_anim_data_t * d = (slice_anim_data_t *) var;

    int32_t x = d->start_x + ((d->end_x - d->start_x) * v) / 100;
    int32_t y = d->start_y + ((d->end_y - d->start_y) * v) / 100;
    lv_obj_set_pos(d->obj, x, y);
}

static void anim_cleanup_cb(lv_anim_t * a)
{
    lv_free(a->var);
}

static void arc_click_cb(lv_event_t * e)
{
    lv_obj_t * arc = lv_event_get_target_obj(e);
    slice_info_t * info = (slice_info_t *)lv_event_get_user_data(e);

    int32_t x_off = (SLICE_OFFSET * lv_trigo_cos(info->mid_angle)) >> LV_TRIGO_SHIFT;
    int32_t y_off = (SLICE_OFFSET * lv_trigo_sin(info->mid_angle)) >> LV_TRIGO_SHIFT;

    if(active_info && active_info != info && active_info->out) {
        slice_anim_data_t * anim_back = (slice_anim_data_t *) lv_malloc(sizeof(slice_anim_data_t));
        anim_back->obj = active_arc;
        anim_back->start_x = lv_obj_get_x(active_arc) - SLICE_OFFSET;
        anim_back->start_y = lv_obj_get_y(active_arc) - SLICE_OFFSET;
        anim_back->end_x = active_info->home.x;
        anim_back->end_y = active_info->home.y;

        active_info->out = false;

        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, anim_back);
        lv_anim_set_exec_cb(&a, anim_move_cb);
        lv_anim_set_duration(&a, 200);
        lv_anim_set_values(&a, 0, 100);
        lv_anim_set_deleted_cb(&a, anim_cleanup_cb);
        lv_anim_start(&a);
    }

    int target_x, target_y;
    if(info->out) {
        target_x = info->home.x;
        target_y = info->home.y;
        info->out = false;
        active_info = NULL;
        active_arc = NULL;
    }
    else {
        target_x = info->home.x + x_off;
        target_y = info->home.y + y_off;
        info->out = true;
        active_info = info;
        active_arc = arc;
    }

    slice_anim_data_t * anim_data = (slice_anim_data_t *) lv_malloc(sizeof(slice_anim_data_t));
    anim_data->obj = arc;
    anim_data->start_x = lv_obj_get_x(arc) - SLICE_OFFSET;
    anim_data->start_y = lv_obj_get_y(arc) - SLICE_OFFSET;
    anim_data->end_x = target_x;
    anim_data->end_y = target_y;

    lv_anim_t a;

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_anim_init(&a);
lv_anim_set_var(&a, anim_data);
lv_anim_set_exec_cb(&a, anim_move_cb);
lv_anim_set_duration(&a, 200);
lv_anim_set_values(&a, 0, 100);
lv_anim_set_deleted_cb(&a, anim_cleanup_cb);
lv_anim_start(&a);
}

static void create_slice(lv_obj_t * parent, int percentage, lv_color_t color)
{
    if(percentage <= 0) return;

    float slice_angle = (percentage * 360.0f) / 100.0f;
    int start = (int)(angle_accum + 0.5f);
    angle_accum += slice_angle;
    int end = (int)(angle_accum + 0.5f);
    if(end > 360) end = 360;

    lv_obj_t * arc = lv_arc_create(parent);
    lv_obj_set_size(arc, CHART_SIZE, CHART_SIZE);
    lv_obj_center(arc);

    lv_arc_set_mode(arc, LV_ARC_MODE_NORMAL);
    lv_arc_set_bg_start_angle(arc, start);
    lv_arc_set_bg_end_angle(arc, end);

    lv_obj_set_style_arc_width(arc, CHART_SIZE / 2, LV_PART_MAIN);
    lv_obj_set_style_arc_width(arc, 0, LV_PART_INDICATOR);

    lv_obj_set_style_arc_color(arc, color, LV_PART_MAIN);
    lv_obj_set_style_arc_rounded(arc, false, LV_PART_MAIN);
    lv_obj_remove_style(arc, NULL, LV_PART_KNOB);
    lv_obj_add_flag(arc, LV_OBJ_FLAG_ADV_HITTEST);

    lv_obj_t * label = lv_label_create(arc);
    lv_label_set_text_fmt(label, "%d%%", percentage);
    int mid_angle = start + ((end - start) / 2);
    int radius = CHART_SIZE / 4;
    int x_offset = (radius * lv_trigo_cos(mid_angle)) >> LV_TRIGO_SHIFT;
    int y_offset = (radius * lv_trigo_sin(mid_angle)) >> LV_TRIGO_SHIFT;

    lv_obj_align(label, LV_ALIGN_CENTER, x_offset, y_offset);

    slice_info_t * info = (slice_info_t *) lv_malloc(sizeof(slice_info_t));
    info->start_angle = start;
    info->end_angle = end;
    info->mid_angle = mid_angle;
    info->out        = false;
    info->home.x = lv_obj_get_x(arc);
    info->home.y = lv_obj_get_y(arc);
    lv_obj_add_event_cb(arc, arc_click_cb, LV_EVENT_CLICKED, info);
}

void lv_example_arc_3(void)
{
    /* Root container: flex row */
    lv_obj_t * root = lv_obj_create(lv_screen_active());
    lv_obj_set_size(root, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_center(root);
    lv_obj_set_flex_flow(root, LV_FLEX_FLOW_ROW);
    lv_obj_set_flex_align(root, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    lv_obj_set_style_pad_all(root, 0, LV_PART_MAIN);
    lv_obj_set_style_border_width(root, 0, LV_PART_MAIN);
    lv_obj_set_style_border_color(root, lv_palette_main(LV_PALETTE_RED), LV_PART_MAIN);
    lv_obj_set_style_bg_opa(root, LV_OPA_TRANS, LV_PART_MAIN);
    lv_obj_remove_flag(root, LV_OBJ_FLAG_SCROLLABLE);

    /* Slices container */
    lv_obj_t * slices_container = lv_obj_create(root);
    lv_obj_set_size(slices_container, CHART_SIZE + 2 * SLICE_OFFSET, CHART_SIZE + 2 * SLICE_OFFSET);
    lv_obj_set_style_pad_all(slices_container, 0, LV_PART_MAIN);
    lv_obj_set_style_margin_all(slices_container, 0, LV_PART_MAIN);
    lv_obj_set_style_border_width(slices_container, 0, LV_PART_MAIN);
    lv_obj_set_style_border_color(slices_container, lv_palette_main(LV_PALETTE_BLUE), LV_PART_MAIN);
    lv_obj_set_style_bg_opa(slices_container, LV_OPA_TRANS, LV_PART_MAIN);
    lv_obj_remove_flag(slices_container, LV_OBJ_FLAG_SCROLLABLE);

    /* Create slices */
    angle_accum = 0.0f;
    create_slice(slices_container, 12, lv_palette_main(LV_PALETTE_RED));
    create_slice(slices_container, 18, lv_palette_main(LV_PALETTE_BLUE));
    create_slice(slices_container, 26, lv_palette_main(LV_PALETTE_GREEN));
    create_slice(slices_container, 24, lv_palette_main(LV_PALETTE_ORANGE));
    create_slice(slices_container, 20, lv_palette_main(LV_PALETTE_BLUE_GREY));
}

#endif

```

6.4.7 API

lv_arc.h
lv_arc_private.h
lv_arclabel.h
lv_arclabel_private.h

6.5 Label Arco (lv_arclabel)

6.5.1 Panoramica

La Arc Label è un widget specializzato progettato per visualizzare il testo lungo un arco. Consente un posizionamento e uno stile flessibili del testo, rendendolo adatto ad applicazioni in cui il testo deve seguire un percorso curvilineo, come in indicatori, quadranti o interfacce personalizzate. Il Widget supporta diverse configurazioni, tra cui l'allineamento del testo, la direzione, la regolazione del raggio e la personalizzazione del colore.

6.5.2 Parti e Stili

- *LV_PART_MAIN* Rappresenta la parte principale dell'Arc Label, incluso il percorso dell'arco e il testo visualizzato lungo di esso. L'aspetto del testo e dell'arco può essere personalizzato utilizzando le tipiche proprietà di stile del testo e dello sfondo.

6.5.3 Utilizzo

Gestione del Testo

- Il testo può essere impostato utilizzando `lv_arclabel_set_text(arclabel, "Your text")`.
- Per il testo formattato, si usa `lv_arclabel_set_text_fmt(arclabel, "Formatted %s", "text")`.
- Il testo statico può essere impostato con `lv_arclabel_set_text_static(arclabel, static_text)`, che evita l'allocazione dinamica della memoria.

Configurazione dell'Angolo

- Impostare l'angolo iniziale dell'arco con `lv_arclabel_set_angle_start(arclabel, angle)`.
- Definire la dimensione dell'arco (ampiezza angolare) utilizzando `lv_arclabel_set_angle_size(arclabel, size)`.
- Gli angoli sono misurati in gradi, partendo dall'asse x positivo (ore 3) e aumentando in senso orario.

Direzione

Specificare la direzione del testo lungo l'arco utilizzando `lv_arclabel_set_dir(arclabel, LV_ARCLABEL_DIR_CLOCKWISE)` o `lv_arclabel_set_dir(arclabel, LV_ARCLABEL_DIR_COUNTERCLOCKWISE)`.

- *LV_ARCLABEL_DIR_CLOCKWISE* Il testo scorre in senso orario lungo l'arco.
- *LV_ARCLABEL_DIR_COUNTERCLOCKWISE* Il testo scorre in senso antiorario lungo l'arco.

Allineamento

Regolare l'allineamento verticale del testo con `lv_arclabel_set_text_vertical_align(arclabel, LV_ARCLABEL_TEXT_ALIGN_CENTER)`. Impostare l'allineamento orizzontale utilizzando `lv_arclabel_set_text_horizontal_align(arclabel, LV_ARCLABEL_TEXT_ALIGN_CENTER)`.

Sia l'allineamento verticale che quello orizzontale utilizzano la stessa logica.

- *LV_ARCLABEL_TEXT_ALIGN_DEFAULT* Utilizza l'allineamento di default.
- *LV_ARCLABEL_TEXT_ALIGN.LEADING* Allinea il testo al bordo iniziale dell'arco.
- *LV_ARCLABEL_TEXT_ALIGN.CENTER* Centra il testo lungo l'arco.
- *LV_ARCLABEL_TEXT_ALIGN.TRAILING* Allinea il testo al bordo finale dell'arco.

Raggio e Offset del Centro

- Impostare il raggio dell'arco con `lv_arclabel_set_radius(arclabel, radius)`.
- Regolare l'offset del centro dell'arco usando `lv_arclabel_set_center_offset_x(arclabel, x)` e `lv_arclabel_set_center_offset_y(arclabel, y)`.

Colore e Ricolorazione

Abilitare la ricolorazione del testo con `lv_arclabel_set_recolor(arclabel, true)`. Questo consente di colorare parti del testo in modo diverso utilizzando i comandi di colore incorporati nella stringa di testo.

Comportamento Interattivo

Per default, Arc Label non è cliccabile. Per renderla interattiva, è necessario aggiungere una gestione eventi personalizzata, poiché non eredita la cliccabilità per default.

6.5.4 Eventi

L'Arc Label eredita principalmente gli eventi dalla classe dell'oggetto base. Non definisce eventi specifici, oltre a quelli comuni a tutti i widget. È possibile associare gestori eventi personalizzati per rispondere alle interazioni, se necessario.

6.5.5 Tasti

L'Arc Label non definisce associazioni specifiche di tasti oltre a quelle ereditate dalla classe dell'oggetto base. La navigazione e l'interazione tramite tastiera richiederebbero un'implementazione aggiuntiva.

6.5.6 Esempio

Semplice Arc Label

```
#include "../../lv_examples.h"
#if LV_USE_ARCLABEL && LV_BUILD_EXAMPLES
static const char * ARCLABEL_TEXT =
    "I'm on an #FA7C45 ARC! Centered with #12c2E9 C##B668E8 0##c471ed L##B654E5 0##C84AB2 F##DB417A F##f64659 U##ff8888 L# feature!\n";
void lv_exemple_arclabel_1(void)
{
    lv_obj_t * arclabel_inner = NULL;
    lv_obj_t * arclabel_outer = NULL;
    lv_obj_t * arclabel_slogan_1 = NULL;
    lv_obj_t * arclabel_slogan_2 = NULL;

    lv_obj_set_style_bg_color(lv_screen_active(), lv_color_black(), LV_PART_MAIN);

    arclabel_inner = lv_arclabel_create(lv_screen_active());
    lv_obj_set_size(arclabel_inner, 200, 200);
    lv_obj_set_style_text_color(arclabel_inner, lv_color_white(), LV_PART_MAIN);
    lv_arclabel_set_text_static(arclabel_inner, ARCLABEL_TEXT);
    lv_arclabel_set_angle_start(arclabel_inner, 180);
    lv_arclabel_set_radius(arclabel_inner, LV_PCT(80));
    lv_arclabel_set_recolor(arclabel_inner, true);
    lv_arclabel_set_text_vertical_align(arclabel_inner, LV_ARCLABEL_TEXT_ALIGN_TRAILING);
    lv_arclabel_set_dir(arclabel_inner, LV_ARCLABEL_DIR_COUNTER_CLOCKWISE);
    lv_arclabel_set_text_horizontal_align(arclabel_inner, LV_ARCLABEL_TEXT_ALIGN_CENTER);
    lv_obj_center(arclabel_inner);

    arclabel_outer = lv_arclabel_create(lv_screen_active());
    lv_obj_set_size(arclabel_outer, 200, 200);
    lv_obj_set_style_text_letter_space(arclabel_outer, 2, LV_PART_MAIN);
    lv_obj_set_style_text_color(arclabel_outer, lv_color_hex(0x888888), LV_PART_MAIN);
    lv_arclabel_set_angle_start(arclabel_outer, -180);
    lv_arclabel_set_text_static(arclabel_outer, ARCLABEL_TEXT);
    lv_arclabel_set_radius(arclabel_outer, LV_PCT(100));
    lv_arclabel_set_recolor(arclabel_outer, true);
    lv_arclabel_set_text_vertical_align(arclabel_outer, LV_ARCLABEL_TEXT_ALIGN_LEADING);
    lv_arclabel_set_dir(arclabel_outer, LV_ARCLABEL_DIR_CLOCKWISE);
    lv_arclabel_set_text_horizontal_align(arclabel_outer, LV_ARCLABEL_TEXT_ALIGN_CENTER);
    lv_obj_center(arclabel_outer);

    arclabel_slogan_1 = lv_arclabel_create(lv_screen_active());
    lv_obj_set_size(arclabel_slogan_1, 300, 200);
    lv_obj_set_style_text_letter_space(arclabel_slogan_1, 2, LV_PART_MAIN);
    lv_obj_set_style_text_color(arclabel_slogan_1, lv_palette_main(LV_PALETTE_AMBER), LV_PART_MAIN);
    lv_arclabel_set_text_static(arclabel_slogan_1, "STAY HUNGRY");
    lv_arclabel_set_offset(arclabel_slogan_1, 30);
    lv_arclabel_set_radius(arclabel_slogan_1, 150);
    lv_arclabel_set_recolor(arclabel_slogan_1, true);
    lv_arclabel_set_text_vertical_align(arclabel_slogan_1, LV_ARCLABEL_TEXT_ALIGN_TRAILING);
    lv_arclabel_set_text_horizontal_align(arclabel_slogan_1, LV_ARCLABEL_TEXT_ALIGN_CENTER);
    lv_arclabel_set_dir(arclabel_slogan_1, LV_ARCLABEL_DIR_COUNTER_CLOCKWISE);
    lv_obj_center(arclabel_slogan_1);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

arclabel_slogan_2 = lv_arclabel_create(lv_screen_active());
lv_obj_set_size(arclabel_slogan_2, 300, 200);
lv_obj_set_style_text_letter_space(arclabel_slogan_2, 2, LV_PART_MAIN);
lv_obj_set_style_text_color(arclabel_slogan_2, lv_palette_main(LV_PALETTE_AMBER), LV_PART_MAIN);
lv_arclabel_set_text_static(arclabel_slogan_2, "STAY FOOLISH");
lv_arclabel_set_offset(arclabel_slogan_2, 30);
lv_arclabel_set_radius(arclabel_slogan_2, 150);
lv_arclabel_set_angle_start(arclabel_slogan_2, 180);
lv_arclabel_set_recolor(arclabel_slogan_2, true);
lv_arclabel_set_text_vertical_align(arclabel_slogan_2, LV_ARCLABEL_TEXT_ALIGN_TRAILING);
lv_arclabel_set_text_horizontal_align(arclabel_slogan_2, LV_ARCLABEL_TEXT_ALIGN_CENTER);
lv_arclabel_set_dir(arclabel_slogan_2, LV_ARCLABEL_DIR_COUNTER_CLOCKWISE);
lv_obj_center(arclabel_slogan_2);

#if LV_FONT_MONTserrat_18
lv_obj_set_style_text_font(arclabel_inner, &lv_font_montserrat_18, LV_PART_MAIN);
lv_obj_set_style_text_font(arclabel_outer, &lv_font_montserrat_18, LV_PART_MAIN);
#endif
#if LV_FONT_MONTserrat_24
lv_obj_set_style_text_font(arclabel_slogan_1, &lv_font_montserrat_24, LV_PART_MAIN);
lv_obj_set_style_text_font(arclabel_slogan_2, &lv_font_montserrat_24, LV_PART_MAIN);
#endif
}
#endif

```

6.5.7 API

[lv_arclabel.h](#)

[lv_arclabel_private.h](#)

6.6 Bar (lv_bar)

6.6.1 Panoramica

Il Widget "Bar" [barra] ha uno sfondo e un indicatore. La lunghezza dell'indicatore rispetto allo sfondo indica il valore corrente della Bar.

È possibile impostare sia il valore iniziale che quello finale della Bar. La modifica del valore iniziale su un valore diverso dal valore minimo nel suo intervallo regola la posizione iniziale dell'indicatore.

6.6.2 Parti e Stili

- [LV_PART_MAIN](#) Lo sfondo della Bar. Utilizza le *tipiche proprietà di stile del background*. L'aggiunta di un padding rende l'indicatore più piccolo o più grande. La proprietà di stile [anim_time](#) imposta il tempo di animazione se i valori sono impostati con [LV_ANIM_ON](#).
- [LV_PART_INDICATOR](#) L'indicatore della Bar; utilizza anch'esso le *tipiche proprietà di stile del background*.

6.6.3 Utilizzo

Orientamento e Dimensioni

- Per orientamento, larghezza e altezza, è sufficiente impostare le proprietà di stile width e height;
- [lv_bar_set_orientation](#)(bar, orientation) per sovrascrivere l'orientamento causato da width e height. I valori validi per orientation sono:
 - [LV_BAR_ORIENTATION_AUTO](#)
 - [LV_BAR_ORIENTATION_HORIZONTAL](#)
 - [LV_BAR_ORIENTATION_VERTICAL](#)

Valore e intervallo

È possibile impostare un nuovo valore con [lv_bar_set_value](#)(bar, new_value, LV_ANIM_ON / OFF). Il valore viene interpretato in un intervallo (valori minimo e massimo) che può essere modificato con [lv_bar_set_range](#)(bar, min, max). L'intervallo di default è 0..100 e la direzione di disegno di default è da sinistra a destra in modalità orizzontale e dal basso verso l'alto in modalità verticale. Se il valore minimo è maggiore del valore massimo, ad esempio 100..0, la direzione di disegno viene invertita.

Il nuovo valore in `lv_bar_set_value()` può essere impostato con o senza animazione a seconda dell'ultimo parametro (LV_ANIM_ON/OFF).

Modalità

La Bar può essere in una delle seguenti modalità:

- `LV_BAR_MODE_NORMAL` Una Bar normale come descritto sopra
- `LV_BAR_MODE_SYMMETRICAL` Disegna l'indicatore dal valore zero al valore corrente. Richiede un valore minimo negativo e un valore massimo positivo, ad esempio [-100..100].
- `LV_BAR_MODE_RANGE` Consente anche di impostare il valore iniziale con `lv_bar_set_start_value(bar, new_value, LV_ANIM_ON / OFF)`. Il valore iniziale deve essere inferiore al valore finale.

Data binding

Per familiarizzare con "observer", "subject" e data binding in generale, visitare la pagina [Observer](#).

Questo metodo di sottoscrizione a un Subject intero influisce direttamente sul valore intero di un Widget Bar. Si noti che si tratta di un binding unidirezionale (Subject ==> Widget), quindi quando il Subject cambia, anche la Bar verrà aggiornata.

Supporta Subject interi e float.

- `lv_bar_bind_value(bar, &subject)`

6.6.4 Eventi

I Widget Bar non inviano eventi speciali.

Ulteriori Informazioni

Scoprire di più sugli [Eventi](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

6.6.5 Tasti

I Widget Bar non elaborano *tasti*.

Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

6.6.6 Esempio

Barra semplice

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

void lv_example_bar_1(void)
{
    lv_obj_t * bar1 = lv_bar_create(lv_screen_active());
    lv_obj_set_size(bar1, 200, 20);
    lv_obj_center(bar1);
    lv_bar_set_value(bar1, 70, LV_ANIM_OFF);
}

#endif
```

Stile di una barra

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/***
 * Example of styling the bar
 */
void lv_example_bar_2(void)
{
    static lv_style_t style_bg;
    static lv_style_t style_indic;

    lv_style_init(&style_bg);
    lv_style_set_border_color(&style_bg, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_border_width(&style_bg, 2);
    lv_style_set_pad_all(&style_bg, 6); /*To make the indicator smaller*/
    lv_style_set_radius(&style_bg, 6);
    lv_style_set_anim_duration(&style_bg, 1000);

    lv_style_init(&style_indic);
    lv_style_set_bg_opa(&style_indic, LV_OPA_COVER);
    lv_style_set_bg_color(&style_indic, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_radius(&style_indic, 3);

    lv_obj_t * bar = lv_bar_create(lv_screen_active());
    lv_obj_remove_style_all(bar); /*To have a clean start*/
    lv_obj_add_style(bar, &style_bg, 0);
    lv_obj_add_style(bar, &style_indic, LV_PART_INDICATOR);

    lv_obj_set_size(bar, 200, 20);
    lv_obj_center(bar);
    lv_bar_set_value(bar, 100, LV_ANIM_ON);
}

#endif
```

Indicatore di temperatura

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

static void set_temp(void * bar, int32_t temp)
{
    lv_bar_set_value((lv_obj_t *)bar, temp, LV_ANIM_ON);
}

/***
 * A temperature meter example
 */
void lv_example_bar_3(void)
{
    static lv_style_t style_indic;

    lv_style_init(&style_indic);
    lv_style_set_bg_opa(&style_indic, LV_OPA_COVER);
    lv_style_set_bg_color(&style_indic, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_bg_grad_color(&style_indic, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_bg_grad_dir(&style_indic, LV_GRAD_DIR_VER);

    lv_obj_t * bar = lv_bar_create(lv_screen_active());
    lv_obj_add_style(bar, &style_indic, LV_PART_INDICATOR);
    lv_obj_set_size(bar, 20, 200);
    lv_obj_center(bar);
    lv_bar_set_range(bar, -20, 40);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_exec_cb(&a, set_temp);
    lv_anim_set_duration(&a, 3000);
    lv_anim_set_reverse_duration(&a, 3000);
    lv_anim_set_var(&a, bar);
    lv_anim_set_values(&a, -20, 40);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);
}

#endif
```

Motivo a strisce e valore dell'intervallo

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/***
 * Bar with stripe pattern and ranged value
 */
void lv_example_bar_4(void)
{
    LV_IMAGE_DECLARE(img_skew_strip);
    static lv_style_t style_indic;

    lv_style_init(&style_indic);
    lv_style_set_bg_image_src(&style_indic, &img_skew_strip);
    lv_style_set_bg_image_tiled(&style_indic, true);
    lv_style_set_bg_image_opa(&style_indic, LV_OPA_30);

    lv_obj_t * bar = lv_bar_create(lv_screen_active());
    lv_obj_add_style(bar, &style_indic, LV_PART_INDICATOR);

    lv_obj_set_size(bar, 260, 20);
    lv_obj_center(bar);
    lv_bar_set_mode(bar, LV_BAR_MODE_RANGE);
    lv_bar_set_value(bar, 90, LV_ANIM_OFF);
    lv_bar_set_start_value(bar, 20, LV_ANIM_OFF);
```

(continues on next page)

(continua dalla pagina precedente)

```
}
#endif
```

Barra con direzione di base LTR e RTL

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/***
 * Bar with LTR and RTL base direction
 */
void lv_example_bar_5(void)
{
    lv_obj_t * label;

    lv_obj_t * bar_ltr = lv_bar_create(lv_screen_active());
    lv_obj_set_size(bar_ltr, 200, 20);
    lv_bar_set_value(bar_ltr, 70, LV_ANIM_OFF);
    lv_obj_align(bar_ltr, LV_ALIGN_CENTER, 0, -30);

    label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Left to Right base direction");
    lv_obj_align_to(label, bar_ltr, LV_ALIGN_OUT_TOP_MID, 0, -5);

    lv_obj_t * bar_rtl = lv_bar_create(lv_screen_active());
    lv_obj_set_style_base_dir(bar_rtl, LV_BASE_DIR_RTL, 0);
    lv_obj_set_size(bar_rtl, 200, 20);
    lv_bar_set_value(bar_rtl, 70, LV_ANIM_OFF);
    lv_obj_align(bar_rtl, LV_ALIGN_CENTER, 0, 30);

    label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Right to Left base direction");
    lv_obj_align_to(label, bar_rtl, LV_ALIGN_OUT_TOP_MID, 0, -5);
}

#endif
```

"Drawer" personalizzato per visualizzare il valore corrente

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

#define MAX_VALUE 100
#define MIN_VALUE 0

static void set_value(void * bar, int32_t v)
{
    lv_bar_set_value((lv_obj_t *)bar, v, LV_ANIM_OFF);
}

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);

    lv_draw_label_dsc_t label_dsc;
    lv_draw_label_dsc_init(&label_dsc);
    label_dsc.font = LV_FONT_DEFAULT;

    char buf[8];
    lv_snprintf(buf, sizeof(buf), "%d", (int)lv_bar_get_value(obj));

    lv_point_t txt_size;
    lv_text_get_size(&txt_size, buf, label_dsc.font, label_dsc.letter_space, label_dsc.line_space, LV_COORD_MAX,
                    label_dsc.flag);

    lv_area_t txt_area;
    txt_area.x1 = 0;
    txt_area.x2 = txt_size.x + 1;
    txt_area.y1 = 0;
    txt_area.y2 = txt_size.y - 1;

    lv_area_t indic_area;
    lv_obj_get_coords(obj, &indic_area);
    lv_area_set_width(&indic_area, lv_area_get_width(&indic_area) * lv_bar_get_value(obj) / MAX_VALUE);

    /*If the indicator is long enough put the text inside on the right*/
    if(lv_area_get_width(&indic_area) > txt_size.x + 20) {
        lv_area_align(&indic_area, &txt_area, LV_ALIGN_RIGHT_MID, -10, 0);
        label_dsc.color = lv_color_white();
    }
    /*If the indicator is still short put the text out of it on the right*/
    else {
        lv_area_align(&indic_area, &txt_area, LV_ALIGN_OUT_RIGHT_MID, 10, 0);
        label_dsc.color = lv_color_black();
    }
    label_dsc.text = buf;
    label_dsc.text_local = true;
    lv_layer_t * layer = lv_event_get_layer(e);
    lv_draw_label(layer, &label_dsc, &txt_area);
}

/***
 * Custom drawer on the bar to display the current value
 */
void lv_example_bar_6(void)
{
    lv_obj_t * bar = lv_bar_create(lv_screen_active());
    lv_bar_set_range(bar, MIN_VALUE, MAX_VALUE);
    lv_obj_set_size(bar, 200, 20);
    lv_obj_center(bar);
    lv_obj_add_event_cb(bar, event_cb, LV_EVENT_DRAW_MAIN_END, NULL);

    lv_anim_t a;
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_anim_init(&a);
lv_anim_set_var(&a, bar);
lv_anim_set_values(&a, 0, 100);
lv_anim_set_exec_cb(&a, set_value);
lv_anim_set_duration(&a, 4000);
lv_anim_set_reverse_duration(&a, 4000);
lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
lv_anim_start(&a);

}

#endif

```

Barra con direzione opposta

```

#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/**
 * Bar with opposite direction
 */
void lv_example_bar_7(void)
{
    lv_obj_t * bar_tob = lv_bar_create(lv_screen_active());
    lv_obj_set_size(bar_tob, 20, 100);
    lv_bar_set_range(bar_tob, 100, 0);
    lv_bar_set_value(bar_tob, 70, LV_ANIM_OFF);
    lv_obj_align(bar_tob, LV_ALIGN_CENTER, 0, 10);

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "From top to bottom");
    lv_obj_align_to(label, bar_tob, LV_ALIGN_OUT_TOP_MID, 0, -5);
}

#endif

```

6.6.7 API

lv_barcode.h
lv_barcode_private.h
lv_bar.h
lv_bar_private.h

6.7 Pulsante (lv_button)

6.7.1 Panoramica

I "Button" [pulsanti] non presentano nuove funzionalità rispetto al *Widget Base (lv_obj)*. Sono utili per scopi semanticici e hanno impostazioni di default leggermente diverse.

I pulsanti, per default, differiscono dal Widget Base nei seguenti modi:

- Non "scrollable"
- Aggiunti al gruppo di default
- Altezza e larghezza di default impostate su `LV_SIZE_CONTENT`

6.7.2 Parti e Stili

- `LV_PART_MAIN` Lo sfondo del pulsante; utilizza le *tipiche proprietà di stile del background*.

6.7.3 Utilizzo

Non ci sono nuove funzionalità rispetto al *Base Widget*.

6.7.4 Eventi

- `LV_EVENT_VALUE_CHANGED` quando il flag `LV_OBJ_FLAG_CHECKABLE` è abilitato e si clicca sul Widget. L'evento si verifica durante la transizione da/verso lo stato selezionato.

💡 Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.7.5 Tasti

Si noti che lo stato di `LV_KEY_ENTER` viene tradotto in `LV_EVENT_PRESSED`, `LV_EVENT_PRESSING` e `LV_EVENT_RELEASED` ecc.

💡 Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.7.6 Esempio

Pulsanti semplici

```
#include "../../lv_examples.h"
#if LV_USE_BUTTON && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);

    if(code == LV_EVENT_CLICKED) {
        LV_LOG_USER("Clicked");
    }
    else if(code == LV_EVENT_VALUE_CHANGED) {
        LV_LOG_USER("Toggled");
    }
}

void lv_example_button_1(void)
{
    lv_obj_t * label;

    lv_obj_t * btn1 = lv_button_create(lv_screen_active());
    lv_obj_add_event_cb(btn1, event_handler, LV_EVENT_ALL, NULL);
    lv_obj_align(btn1, LV_ALIGN_CENTER, 0, -40);
    lv_obj_remove_flag(btn1, LV_OBJ_FLAG_PRESS_LOCK);

    label = lv_label_create(btn1);
    lv_label_set_text(label, "Button");
    lv_obj_center(label);

    lv_obj_t * btn2 = lv_button_create(lv_screen_active());
    lv_obj_add_event_cb(btn2, event_handler, LV_EVENT_ALL, NULL);
    lv_obj_align(btn2, LV_ALIGN_CENTER, 0, 40);
    lv_obj_add_flag(btn2, LV_OBJ_FLAG_CHECKABLE);
    lv_obj_set_height(btn2, LV_SIZE_CONTENT);

    label = lv_label_create(btn2);
    lv_label_set_text(label, "Toggle");
    lv_obj_center(label);
}
#endif
```

Pulsanti con stili

```
#include "../../lv_examples.h"
#if LV_USE_BUTTON && LV_BUILD_EXAMPLES

/***
 * Style a button from scratch
 */
void lv_example_button_2(void)
{
    /*Init the style for the default state*/
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_radius(&style, 3);

    lv_style_set_bg_opa(&style, LV_OPA_100);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_bg_grad_color(&style, lv_palette_darken(LV_PALETTE_BLUE, 2));
    lv_style_set_bg_grad_dir(&style, LV_GRAD_DIR_VER);

    lv_style_set_border_opa(&style, LV_OPA_40);
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_GREY));

    lv_style_set_shadow_width(&style, 8);
    lv_style_set_shadow_color(&style, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_shadow_offset_y(&style, 8);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_set_outline_opa(&style, LV_OPA_COVER);
lv_style_set_outline_color(&style, lv_palette_main(LV_PALETTE_BLUE));

lv_style_set_text_color(&style, lv_color_white());
lv_style_set_pad_all(&style, 10);

/*Init the pressed style*/
static lv_style_t style_pr;
lv_style_init(&style_pr);

/*Add a large outline when pressed*/
lv_style_set_outline_width(&style_pr, 30);
lv_style_set_outline_opa(&style_pr, LV_OPA_TRANSP);

lv_style_set_translate_y(&style_pr, 5);
lv_style_set_shadow_offset_y(&style_pr, 3);
lv_style_set_bg_color(&style_pr, lv_palette_darken(LV_PALETTE_BLUE, 2));
lv_style_set_bg_grad_color(&style_pr, lv_palette_darken(LV_PALETTE_BLUE, 4));

/*Add a transition to the outline*/
static lv_style_transition_dsc_t trans;
static lv_style_prop_t props[] = {LV_STYLE_OUTLINE_WIDTH, LV_STYLE_OUTLINE_OPA, 0};
lv_style_transition_dsc_init(&trans, props, lv_anim_path_linear, 300, 0, NULL);

lv_style_set_transition(&style_pr, &trans);

lv_obj_t * btn1 = lv_button_create(lv_screen_active());
lv_obj_remove_style_all(btn1); /*Remove the style coming from the theme*/
lv_obj_add_style(btn1, &style, 0);
lv_obj_add_style(btn1, &style_pr, LV_STATE_PRESSED);
lv_obj_set_size(btn1, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
lv_obj_center(btn1);

lv_obj_t * label = lv_label_create(btn1);
lv_label_set_text(label, "Button");
lv_obj_center(label);
}
#endif

```

Pulsante gommoso

```

#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_BUTTON

/***
 * Create a style transition on a button to act like a gum when clicked
 */
void lv_example_button_3(void)
{
    /*Properties to transition*/
    static lv_style_prop_t props[] = {
        LV_STYLE_TRANSFORM_WIDTH, LV_STYLE_TRANSFORM_HEIGHT, LV_STYLE_TEXT_LETTER_SPACE, 0
    };

    /*Transition descriptor when going back to the default state.
     *Add some delay to be sure the press transition is visible even if the press was very short*/
    static lv_style_transition_dsc_t transition_dsc_def;
    lv_style_transition_dsc_init(&transition_dsc_def, props, lv_anim_path_overshoot, 250, 100, NULL);

    /*Transition descriptor when going to pressed state.
     *No delay, go to presses state immediately*/
    static lv_style_transition_dsc_t transition_dsc_pr;
    lv_style_transition_dsc_init(&transition_dsc_pr, props, lv_anim_path_ease_in_out, 250, 0, NULL);

    /*Add only the new transition to be default state*/
    static lv_style_t style_def;
    lv_style_init(&style_def);
    lv_style_set_transition(&style_def, &transition_dsc_def);

    /*Add the transition and some transformation to the presses state.*/
    static lv_style_t style_pr;
    lv_style_init(&style_pr);
    lv_style_set_transform_width(&style_pr, 10);
    lv_style_set_transform_height(&style_pr, -10);
    lv_style_set_text_letter_space(&style_pr, 10);
    lv_style_set_transition(&style_pr, &transition_dsc_pr);

    lv_obj_t * btn1 = lv_button_create(lv_screen_active());
    lv_obj_align(btn1, LV_ALIGN_CENTER, 0, -80);
    lv_obj_add_style(btn1, &style_pr, LV_STATE_PRESSED);
    lv_obj_add_style(btn1, &style_def, 0);

    lv_obj_t * label = lv_label_create(btn1);
    lv_label_set_text(label, "Gum");
}
#endif

```

6.7.7 API

[lv_api_map_v9_0.h](#)

[lv_button.h](#)

[lv_button_private.h](#)

[lv_buttonmatrix.h](#)

[lv_buttonmatrix_private.h](#)

6.8 Matrice di Pulsanti (lv_buttonmatrix)

6.8.1 Panoramica

Il Widget "Button Matrix" [Matrice Pulsanti] è un modo semplice per visualizzare più pulsanti in righe e colonne, semplice perché i pulsanti non vengono effettivamente creati, ma solo disegnati virtualmente al volo. Con la Button Matrix, ogni pulsante utilizza solo otto byte di memoria aggiuntivi invece dei circa 100-150 byte di un normale Widget [Button](#) più i circa 100 byte per il Widget [Label](#).

I nuovi Widget Button Matrix vengono aggiunti al gruppo di default (se impostato). Inoltre, Button Matrix è un Widget modificabile: consente di selezionare e cliccare sui pulsanti anche con encoder e navigazione da tastiera.

6.8.2 Parti e Stili

- [LV_PART_MAIN](#) Il background della Button Matrix, utilizza le tipiche proprietà di stile del background. `pad_row` e `pad_column` impostano lo spazio tra i pulsanti.
- [LV_PART_ITEMS](#) Tutti i pulsanti utilizzano le tipiche proprietà di stile del testo e dello sfondo, ad eccezione di traslazioni e trasformazioni.

6.8.3 Utilizzo

Mappa dei pulsanti

Il numero di pulsanti, le loro posizioni e il testo sono controllati da un array di stringhe descrittive, chiamato *map*, passato a [lv_buttonmatrix_set_map](#)(`btn_matrix`, `my_map`). La dichiarazione di map dovrebbe essere simile a `const char * map[] = {"button1", "button2", "button3", NULL};`. Si noti che l'ultimo elemento deve essere `NULL` o una stringa vuota ("")!

Utilizzare "\n" nella map per inserire un **line break** [andata-a-capo]. Ad esempio `{"button1", "button2", "\n", "button3", ""}`. La larghezza dei pulsanti di ogni riga viene calcolata automaticamente. Quindi, nell'esempio, la prima riga avrà 2 pulsanti, ciascuno con larghezza pari al 50%, e una seconda riga con 1 pulsante con larghezza pari al 100%.

Nota

Il numero di pulsanti non include né gli elementi di "newline" né l'elemento di chiusura dell'array.

Larghezza dei pulsanti

La larghezza dei pulsanti può essere impostata in proporzione alla larghezza degli altri pulsanti nella stessa riga con [lv_buttonmatrix_set_button_width](#)(`btn_matrix`, `button_id`, `width`). Ad esempio, in una riga con due pulsanti: `buttonA, width = 1` e `buttonB, width = 2`, `buttonA` avrà una larghezza pari al 33% e `buttonB` una larghezza pari al 66%. Questo è simile al funzionamento della proprietà "flex-grow" in CSS. La larghezza deve essere compresa tra [1 e 15], con il valore di default pari a 1.

Comportamento dei pulsanti

Il comportamento di ogni pulsante può essere personalizzato con i seguenti flag di controllo:

- [LV_BUTTONMATRIX_CTRL_HIDDEN](#): Nasconde il pulsante; continua a occupare il suo spazio nel layout.
- [LV_BUTTONMATRIX_CTRL_NO_REPEAT](#): Non emette eventi [LV_EVENT_LONG_PRESSED_REPEAT](#) mentre il pulsante è premuto a lungo.
- [LV_BUTTONMATRIX_CTRL_DISABLED](#): Disabilita il pulsante come [LV_STATE_DISABLED](#) nei Widget normali.
- [LV_BUTTONMATRIX_CTRL_CHECKABLE](#): Abilita l'attivazione/disattivazione di [LV_STATE_CHECKED](#) quando cliccato.
- [LV_BUTTONMATRIX_CTRL_CHECKED](#): Imposta il pulsante come selezionato. Utilizzerà gli stili [LV_STATE_CHECKED](#).

- *LV_BUTTONMATRIX_CTRL_CLICK_TRIG*: 1: Abilita l'invio di `LV_EVENT_VALUE_CHANGE` al CLICK, 0: invia `LV_EVENT_VALUE_CHANGE` al PRESS.
- *LV_BUTTONMATRIX_CTRL_POPOVER*: Mostra il testo del pulsante in un pop-over quando viene premuto.
- *LV_BUTTONMATRIX_CTRL_RECOLOR*: Abilita la ricolorazione del testo con `#color`
- *LV_BUTTONMATRIX_CTRL_CUSTOM_1*: Flag personalizzato di libero utilizzo
- *LV_BUTTONMATRIX_CTRL_CUSTOM_2*: Flag personalizzato di libero utilizzo

Per default, questi flag sono disabilitati.

Per impostare e cancellare i flag di controllo di un pulsante, utilizzare

- `lv_buttonmatrix_set_button_ctrl(btn_matrix, button_id, LV_BUTTONMATRIX_CTRL_...)` e
- `lv_buttonmatrix_clear_button_ctrl(btn_matrix, button_id, LV_BUTTONMATRIX_CTRL_...)`

rispettivamente. `button_id` è un indice del pulsante a base zero (0 = primo pulsante). I valori di `LV_BUTTONMATRIX_CTRL_...` possono essere combinati tramite OR bit a bit quando vengono passati a queste funzioni.

Per impostare e cancellare lo stesso attributo di controllo per tutti i pulsanti in una Button Matrix, usare

- `lv_buttonmatrix_set_button_ctrl_all(btn_matrix, LV_BUTTONMATRIX_CTRL_...)` e
- `lv_buttonmatrix_clear_button_ctrl_all(btn_matrix, LV_BUTTONMATRIX_CTRL_...)`

rispettivamente.

Per impostare una mappa di controllo per una Button Matrix (simile a *Button map*), si usa

- `lv_buttonmatrix_set_ctrl_map(btn_matrix, ctrl_map)`.

Un elemento di `ctrl_map` dovrebbe apparire come `ctrl_map[0] = width | LV_BUTTONMATRIX_CTRL_NO_REPEAT | LV_BUTTONMATRIX_CTRL_CHECKABLE`. Il numero di elementi deve essere uguale a quello dei pulsanti.

Uno selezionato

La funzione "One-checked" [selezione unica] può essere abilitata con `lv_buttonmatrix_set_one_checked(btn_matrix, true)` per consentire la selezione di un solo pulsante alla volta.

6.8.4 Eventi

- *LV_EVENT_VALUE_CHANGED*: Inviato quando un pulsante viene premuto/rilasciato o ripetuto dopo una pressione prolungata. Il parametro event è impostato sull'ID del pulsante premuto/rilasciato.

`lv_buttonmatrix_get_selected_button(btn_matrix)` restituisce l'indice del pulsante rilasciato più di recente (il pulsante con il focus) oppure `LV_BUTTONMATRIX_BUTTON_NONE` se non è stato trovato alcun pulsante di questo tipo.

`lv_buttonmatrix_get_button_text(btn_matrix, button_id)` restituisce un puntatore al testo del pulsante specificato dall'indice a base zero `button_id`.

Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.8.5 Tasti

- `LV_KEY_RIGHT/UP/LEFT/RIGHT` Per navigare tra i pulsanti e selezionarne uno
- `LV_KEY_ENTER` Per premere/rilasciare il pulsante selezionato

Si noti che una pressione prolungata sulla Button Matrix con un encoder può significare entrare/uscire dalla modalità di modifica e che una semplice pressione prolungata di un pulsante può anche significare ripeterlo. Per evitare questa contraddizione, aggiungere `lv_buttonmatrix_set_button_ctrl_all(btn_matrix, LV_BUTTONMATRIX_CTRL_CLICK_TRIG | LV_BUTTONMATRIX_CTRL_NO_REPEAT)` alla Button Matrix se utilizzata con un encoder. In questo modo la funzione di ripetizione viene disattivata, quindi il pulsante non verrà attivato quando si esce dalla modalità di modifica.

1 Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.8.6 Esempio

Matrice di pulsanti semplice

```
#include "../../lv_examples.h"
#if LV_USE_BUTTONMATRIX && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        uint32_t id = lv_buttonmatrix_get_selected_button(obj);
        const char * txt = lv_buttonmatrix_get_button_text(obj, id);
        LV_UNUSED(txt);
        LV_LOG_USER("%s was pressed\n", txt);
    }
}

static const char * btm_map[] = {"1", "2", "3", "4", "5", "\n",
                                "6", "7", "8", "9", "0", "\n",
                                "Action1", "Action2", ""};

void lv_example_buttonmatrix_1(void)
{
    lv_obj_t * btm1 = lv_buttonmatrix_create(lv_screen_active());
    lv_buttonmatrix_set_map(btm1, btm_map);
    lv_buttonmatrix_set_button_width(btm1, 10, 2); /*Make "Action1" twice as wide as "Action2"*/
    lv_buttonmatrix_set_button_ctrl(btm1, 10, LV_BUTTONMATRIX_CTRL_CHECKABLE);
    lv_buttonmatrix_set_button_ctrl(btm1, 11, LV_BUTTONMATRIX_CTRL_CHECKED);
    lv_obj_align(btm1, LV_ALIGN_CENTER, 0, 0);
    lv_obj_add_event_cb(btm1, event_handler, LV_EVENT_ALL, NULL);
}

#endif
```

Pulsanti personalizzati

```
#include "../../lv_examples.h"
#if LV_USE_BUTTONMATRIX && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    /*When the button matrix draws the buttons...*/
    if(base_dsc->part == LV_PART_ITEMS) {
        bool pressed = false;
        if(lv_buttonmatrix_get_selected_button(obj) == base_dsc->id1 && lv_obj_has_state(obj, LV_STATE_PRESSED)) {
            pressed = true;
        }

        /*Change the draw descriptor of the 2nd button*/
        if(base_dsc->id1 == 1) {
            lv_draw_fill_dsc_t * fill_draw_dsc = lv_draw_task_get_fill_dsc(draw_task);
            if(fill_draw_dsc) {
                fill_draw_dsc->radius = 0;
                if(pressed) fill_draw_dsc->color = lv_palette_darken(LV_PALETTE_BLUE, 3);
                else fill_draw_dsc->color = lv_palette_main(LV_PALETTE_BLUE);
            }
            lv_draw_box_shadow_dsc_t * box_shadow_draw_dsc = lv_draw_task_get_box_shadow_dsc(draw_task);
            if(box_shadow_draw_dsc) {
                box_shadow_draw_dsc->width = 6;
                box_shadow_draw_dsc->ofs_x = 3;
                box_shadow_draw_dsc->ofs_y = 3;
            }
        }
        lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
        if(label_draw_dsc) {
            label_draw_dsc->color = lv_color_white();
        }
    }

    /*Change the draw descriptor of the 3rd button*/
    else if(base_dsc->id1 == 2) {
        lv_draw_fill_dsc_t * fill_draw_dsc = lv_draw_task_get_fill_dsc(draw_task);
        if(fill_draw_dsc) {
            fill_draw_dsc->radius = LV_RADIUS_CIRCLE;
            if(pressed) fill_draw_dsc->color = lv_palette_darken(LV_PALETTE_RED, 3);
            else fill_draw_dsc->color = lv_palette_main(LV_PALETTE_RED);
        }
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_draw_box_shadow_dsc_t * box_shadow_draw_dsc = lv_draw_task_get_box_shadow_dsc(draw_task);
if(box_shadow_draw_dsc) {
    box_shadow_draw_dsc->radius = LV_RADIUS_CIRCLE;
}
} else if(base_dsc->id1 == 3) {
    lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
    if(label_draw_dsc) {
        label_draw_dsc->opa = 0;
    }
    if(lv_draw_task_get_type(draw_task) == LV_DRAW_TASK_TYPE_FILL) {
        LV_IMAGE_DECLARE(img_star);
        lv_image_header_t header;
        lv_result_t res = lv_image_decoder_get_info(&img_star, &header);
        if(res != LV_RESULT_OK) return;

        lv_area_t a;
        a.x1 = 0;
        a.x2 = header.w - 1;
        a.y1 = 0;
        a.y2 = header.h - 1;
        lv_area_t draw_task_area;
        lv_task_get_area(draw_task, &draw_task_area);
        lv_area_align(&draw_task_area, &a, LV_ALIGN_CENTER, 0, 0);

        lv_draw_image_dsc_t img_draw_dsc;
        lv_draw_image_dsc_init(&img_draw_dsc);
        img_draw_dsc.src = &img_star;
        img_draw_dsc.recolor = lv_color_black();
        if(pressed) img_draw_dsc.recolor.opa = LV_OPA_30;

        lv_draw_image(base_dsc->layer, &img_draw_dsc, &a);
    }
}
}

/**
 * Add custom drawer to the button matrix to customize buttons one by one
 */
void lv_example_buttonmatrix_2(void)
{
    lv_obj_t * btm = lv_buttonmatrix_create(lv_screen_active());
    lv_obj_add_event_cb(btm, event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(btm, LV_OB_FLAG_SEND_DRAW_TASK_EVENTS);
    lv_obj_center(btm);
}

#endif

```

Paginazione

```

#include "../../lv_examples.h"
#if LV_USE_BUTTONMATRIX && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    uint32_t id = lv_buttonmatrix_get_selected_button(obj);
    bool prev = id == 0;
    bool next = id == 6;
    if(prev || next) {
        /*Find the checked button*/
        uint32_t i;
        for(i = 1; i < 7; i++) {
            if(lv_buttonmatrix_has_button_ctrl(obj, i, LV_BUTTONMATRIX_CTRL_CHECKED)) break;
        }

        if(prev && i > 1) i--;
        else if(next && i < 5) i++;

        lv_buttonmatrix_set_button_ctrl(obj, i, LV_BUTTONMATRIX_CTRL_CHECKED);
    }
}

/**
 * Make a button group (pagination)
 */
void lv_example_buttonmatrix_3(void)
{
    static lv_style_t style_bg;
    lv_style_init(&style_bg);
    lv_style_set_pad_all(&style_bg, 0);
    lv_style_set_pad_gap(&style_bg, 0);
    lv_style_set_clip_corner(&style_bg, true);
    lv_style_set_radius(&style_bg, LV_RADIUS_CIRCLE);
    lv_style_set_border_width(&style_bg, 0);

    static lv_style_t style_btn;
    lv_style_init(&style_btn);
    lv_style_set_radius(&style_btn, 0);
    lv_style_set_border_width(&style_btn, 1);
    lv_style_set_border_opa(&style_btn, LV_OPA_50);
    lv_style_set_border_color(&style_btn, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_border_side(&style_btn, LV_BORDER_SIDE_INTERNAL);
    lv_style_set_radius(&style_btn, 0);

    static const char * map[] = {LV_SYMBOL_LEFT, "1", "2", "3", "4", "5", LV_SYMBOL_RIGHT, ""};

    lv_obj_t * btm = lv_buttonmatrix_create(lv_screen_active());
    lv_buttonmatrix_set_map(btm, map);
    lv_obj_add_style(btm, &style_bg, 0);
    lv_obj_add_style(btm, &style_btn, LV_PART_ITEMS);
    lv_obj_add_event_cb(btm, event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_set_size(btm, 225, 35);

    /*Allow selecting on one number at time*/

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_buttonmatrix_set_button_ctrl_all(btnm, LV_BUTTONMATRIX_CTRL_CHECKABLE);
lv_buttonmatrix_clear_button_ctrl(btnm, 0, LV_BUTTONMATRIX_CTRL_CHECKABLE);
lv_buttonmatrix_clear_button_ctrl(btnm, 6, LV_BUTTONMATRIX_CTRL_CHECKABLE);

lv_buttonmatrix_set_one_checked(btnm, true);
lv_buttonmatrix_set_button_ctrl(btnm, 1, LV_BUTTONMATRIX_CTRL_CHECKED);

lv_obj_center(btnm);

}

#endif

```

6.8.7 API

lv_buttonmatrix.h

lv_buttonmatrix_private.h

6.9 Calendar (lv_calendar)

6.9.1 Panoramica

Il Widget "Calendar" è un calendario classico che può:

- mostrare i giorni di qualsiasi mese in una matrice 7x7;
- mostrare il nome dei giorni;
- evidenziare il giorno corrente (today);
- evidenziare eventuali date definite dall'utente.

Il Calendar viene aggiunto al gruppo di default (se impostato). Il Calendar è un Widget modificabile che consente di selezionare e cliccare sulle date con la navigazione tramite encoder o tastiera, nonché con dispositivi di input di tipo puntatore.

Per rendere il Calendar flessibile, per default non mostra l'anno o il mese corrente. Al suo posto, sono disponibili delle "intestazioni" opzionali che possono essere allegate al calendario.

6.9.2 Parti e Stili

Il Widget Calendar utilizza il *Button Matrix* per organizzare i giorni in una matrice.

- *LV_PART_MAIN* Sfondo del Calendar. Utilizza le *tipiche proprietà di stile del background*.
- *LV_PART_ITEMS* Si riferisce a date e nomi dei giorni. I flag di controllo del "Button matrix" sono impostati per differenziare i pulsanti e viene aggiunto un evento personalizzato per modificare le proprietà dei pulsanti come segue:
 - I nomi dei giorni non hanno bordo, non hanno sfondo e sono visualizzati in grigio.
 - I giorni del mese precedente e successivo hanno il flag *LV_BUTTONMATRIX_CTRL_DISABLED*
 - "today" [oggi] ha un bordo più spesso con il colore primario del tema; i giorni evidenziati hanno una certa opacità con il colore primario del tema.

6.9.3 Utilizzo

Alcune funzioni utilizzano il tipo *lv_calendar_date_t* una struttura con i campi year, month e day.

Data corrente

Per impostare la data corrente (today), utilizzare la funzione *lv_calendar_set_today_date(calendar, year, month, day)*. month deve essere compreso tra 1 e 12 e day tra 1 e 31.

Mese visualizzato

Per impostare la data visualizzata, si usa `lv_calendar_set_month_shown(calendar, year, month)`

Giorni evidenziati

L'elenco delle date evidenziate deve essere memorizzato in un array `lv_calendar_date_t` e applicato al Calendario chiamando `lv_calendar_set_highlighted_dates(calendar, highlighted_dates, date_num)`. Verrà salvato solo il puntatore dell'array, quindi l'array deve avere uno scope statico o globale.

Nomi dei giorni

I nomi dei giorni possono essere modificati con `lv_calendar_set_day_names(calendar, day_names)` dove `day_names` è simile a `const char * day_names[7] = {"Su", "Mo", ...};`; Viene salvato solo il puntatore dei nomi dei giorni, quindi l'array dovrebbe avere un ambito statico o globale.

Elenco anni personalizzato

Impostare un elenco di anni personalizzato con `lv_calendar_header_dropdown_set_year_list(calendar, years_list)` dove `years_list` è un puntatore all'elenco degli anni. Può essere una stringa costante come `static const char * years = "2023\n2022\n2021\n2020\n2019";`, oppure può essere generato dinamicamente in un buffer. Il calendario memorizza questi dati in un Widget "Drop-Down List" tramite `lv_dropdown_set_options()`, in modo che il puntatore stringa passato possa essere fornito da una variabile locale o da un buffer e non debba persistere oltre la chiamata.

Calendario cinese

Il calendario cinese è uno strumento culturale tradizionale che integra elementi come il calendario lunare, i termini solari e le festività tradizionali. È ampiamente utilizzato nella vita sociale cinese, aiutando le persone a comprendere le date del calendario lunare, a organizzare le attività festive e a ereditare l'eccellente cultura tradizionale della nazione cinese. Che si tratti di famiglia, lavoro o istruzione, il calendario cinese svolge un ruolo insostituibile, consentendo alle persone di comprendere e apprezzare meglio il fascino della cultura tradizionale cinese.

Per utilizzare il calendario cinese, si usa `lv_calendar_set_chinese_mode(calendar, true)` per abilitarlo.

6.9.4 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato se si clicca su una data. `lv_calendar_get_pressed_date(calendar, &date)` per impostare date sulla data attualmente premuta. Restituisce `LV_RESULT_OK` se è presente una data premuta valida; in caso contrario restituisce `LV_RESULT_INVALID`.

Ulteriori Informazioni

Scoprire di più sugli [Eventi](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

6.9.5 Tasti

- `LV_KEY_DOWN/UP/LEFT/RIGHT` Per navigare tra i pulsanti fino alle date
- `LV_KEY_ENTER` Per premere/rilasciare la data selezionata

Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

6.9.6 Intestazioni

Da LVGL v8.1 in poi, l'intestazione viene aggiunta direttamente al widget Calendar e l'API delle intestazioni è stata modificata.

Pulsanti freccia

`lv_calendar_add_header_arrow(calendar)` crea un'intestazione che contiene una freccia sinistra e una freccia destra ai lati e un testo tra le frecce che mostra l'anno e il mese correnti.

Drop-down

`lv_calendar_add_header_dropdown(calendar)` crea un'intestazione che contiene 2 "Drop-Down List" per l'anno e il mese.

6.9.7 Esempio

Calendario con intestazione

```
#include "../../lv_examples.h"
#if LV_USE_CALENDAR && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = (lv_obj_t *)lv_event_get_current_target(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        lv_calendar_date_t date;
        if(lv_calendar_get_pressed_date(obj, &date)) {
            LV_LOG_USER("Clicked date: %02d.%02d.%d", date.day, date.month, date.year);
        }
    }
}

void lv_example_calendar_1(void)
{
    lv_obj_t * calendar = lv_calendar_create(lv_screen_active());
    lv_obj_set_size(calendar, 185, 230);
    lv_obj_align(calendar, LV_ALIGN_CENTER, 0, 27);
    lv_obj_add_event_cb(calendar, event_handler, LV_EVENT_ALL, NULL);

    lv_calendar_set_today_date(calendar, 2021, 02, 23);
    lv_calendar_set_month_shown(calendar, 2021, 02);

    /*Highlight a few days*/
    static lv_calendar_date_t highlighted_days[3];      /*Only its pointer will be saved so should be static*/
    highlighted_days[0].year = 2021;
    highlighted_days[0].month = 02;
    highlighted_days[0].day = 6;

    highlighted_days[1].year = 2021;
    highlighted_days[1].month = 02;
    highlighted_days[1].day = 11;

    highlighted_days[2].year = 2022;
    highlighted_days[2].month = 02;
    highlighted_days[2].day = 22;

    lv_calendar_set_highlighted_dates(calendar, highlighted_days, 3);
}

#endif LV_USE_CALENDAR_HEADER_DROPDOWN
lv_calendar_add_header_dropdown(calendar);
#elif LV_USE_CALENDAR_HEADER_ARROW
lv_calendar_add_header_arrow(calendar);
#endif
}

#endif
```

Calendario cinese

```
#include "../../lv_examples.h"
#if LV_USE_CALENDAR && LV_USE_CALENDAR_CHINESE && LV_BUILD_EXAMPLES

void lv_example_calendar_2(void)
{
    lv_obj_t * calendar = lv_calendar_create(lv_screen_active());
    lv_obj_set_size(calendar, 300, 300);
    lv_obj_align(calendar, LV_ALIGN_TOP_MID, 0, 0);

    lv_calendar_set_today_date(calendar, 2024, 03, 22);
    lv_calendar_set_month_shown(calendar, 2024, 03);

    #if LV_USE_CALENDAR_HEADER_DROPDOWN
    lv_calendar_add_header_dropdown(calendar);
    #elif LV_USE_CALENDAR_HEADER_ARROW
    lv_calendar_add_header_arrow(calendar);
    #endif

    lv_calendar_set_chinese_mode(calendar, true);
    lv_obj_set_style_text_font(calendar, &lv_font_source_han_sans_sc_14_cjk, LV_PART_MAIN);
}
```

(continues on next page)

(continua dalla pagina precedente)

```
#else
void lv_example_calendar_2(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "chinese calendar is not enabled");
    lv_obj_center(label);
}

#endif
```

6.9.8 API

lv_api_map_v9_1.h
lv_calendar.h
lv_calendar_chinese.h
lv_calendar_header_arrow.h
lv_calendar_header_dropdown.h
lv_calendar_private.h

6.10 Canvas (lv_canvas)

6.10.1 Panoramica

Un Canvas eredita da *Image* e lo estende, consentendo all'utente di disegnare qualsiasi cosa. Rettangoli, testo, immagini, linee, archi, ecc. possono essere disegnati utilizzando l'ampio motore di disegno di LVGL.

6.10.2 Parti e Stili

- *LV_PART_MAIN* Utilizza le proprietà *tipiche del background* e lo stile dell'immagine.

6.10.3 Utilizzo

Buffer

Il Canvas necessita di un buffer in cui memorizzare l'immagine disegnata. Per assegnare un buffer a un Canvas, si usa *lv_canvas_set_buffer(canvas, buffer, width, height, LV_COLOR_FORMAT_...)*. Dove buffer è un buffer statico (non una semplice variabile locale) per contenere l'immagine del Canvas. Ad esempio, per un buffer ARGB8888 100x50: `static uint8_t buffer[100 * 50 * 4]`.

Oppure si può usare `static uint8_t buffer[LV_CANVAS_BUF_SIZE(width, height, bits_per_pixel, stride_in_bytes)]`.

Il Canvas supporta tutti i formati colore come *LV_COLOR_FORMAT_ARGB8888* o *LV_COLOR_FORMAT_I2*. Consultare l'elenco completo nella sezione *Formati di colore*.

Colori indicizzati

Per i formati di colore indicizzati (*LV_COLOR_FORMAT_I1/2/4/8*), la paletta deve essere popolata per tutti gli indici di palette che verranno utilizzati tramite *lv_canvas_set_palette(canvas, index, color)*. Ad esempio, il seguente comando imposta i pixel con *index==3* sul rosso.

```
lv_canvas_set_palette(canvas, 3, lv_color_hex(0xff0000))
```

Disegno

Per impostare il colore di un singolo pixel sul Canvas, si usa *lv_canvas_set_px(canvas, x, y, color, opa)*. Con i formati di colore indicizzati (*LV_COLOR_FORMAT_I1/2/4/8*) si passa l'indice del colore come argomento *color* utilizzando il canale *blu* nel valore *color* ad esempio *lv_color_make(0, 0, index)*.

`lv_canvas_fill_bg(canvas, lv_color_hex(0x00ff00), LV_OPA_50)` riempie l'intero Canvas di blu con un'opacità del 50%. Si noti che, se il formato colore corrente non supporta i colori (ad esempio `LV_COLOR_FORMAT_A8`), il colore verrà ignorato. Allo stesso modo, se l'opacità non è supportata (ad esempio `LV_COLOR_FORMAT_RGB565`), verrà ignorata.

Un array di pixel può essere copiato sul Canvas con `lv_canvas_copy_buf(canvas, buffer_to_copy, x, width, height)`. Il formato colore del buffer e del Canvas devono corrispondere.

Per disegnare qualcosa sul Canvas, utilizzare direttamente le funzioni di disegno di LVGL. Vedere gli esempi per maggiori dettagli.

Le funzioni di disegno possono disegnare in qualsiasi formato colore supportato da LVGL. In genere, questo significa `LV_COLOR_FORMAT_RGB565`, `LV_COLOR_FORMAT_RGB888`, `LV_COLOR_FORMAT_XRGB888` e `LV_COLOR_FORMAT_ARGB8888`.

6.10.4 Eventi

I Widget Canvas non inviano eventi speciali.

Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.10.5 Tasti

I Widget Canvas non elaborano alcun *Tasto*.

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.10.6 Esempio

Disegno sul Canvas e rotazione

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 200
#define CANVAS_HEIGHT 150

void lv_example_canvas_1(void)
{
    lv_draw_rect_dsc_t rect_dsc;
    lv_draw_rect_dsc_init(&rect_dsc);
    rect_dsc.radius = 10;
    rect_dsc.bg_opa = LV_OPA_COVER;
    rect_dsc.bg_grad.dir = LV_GRAD_DIR_VER;
    rect_dsc.bg_grad.stops[0].color = lv_palette_main(LV_PALETTE_RED);
    rect_dsc.bg_grad.stops[0].opa = LV_OPA_100;
    rect_dsc.bg_grad.stops[1].color = lv_palette_main(LV_PALETTE_BLUE);
    rect_dsc.bg_grad.stops[1].opa = LV_OPA_50;
    rect_dsc.border_width = 2;
    rect_dsc.border_opa = LV_OPA_90;
    rect_dsc.border_color = lv_color_white();
    rect_dsc.shadow_width = 5;
    rect_dsc.shadow_offset_x = 5;
    rect_dsc.shadow_offset_y = 5;

    lv_draw_label_dsc_t label_dsc;
    lv_draw_label_dsc_init(&label_dsc);
    label_dsc.color = lv_palette_main(LV_PALETTE_ORANGE);
    label_dsc.text = "Some text on text canvas";
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf_16bpp, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_RGB565);
    LV_DRAW_BUF_INIT_STATIC(draw_buf_16bpp);

    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf_16bpp);
    lv_obj_center(canvas);
    lv_canvas_fill_bg(canvas, lv_palette_lighten(LV_PALETTE_GREY, 3), LV_OPA_COVER);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_area_t coords_rect = {30, 20, 100, 70};
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_draw_rect(&layer, &rect_dsc, &coords_rect);

lv_area_t coords_text = {40, 80, 100, 120};
lv_draw_label(&layer, &label_dsc, &coords_text);

lv_canvas_finish_layer(canvas, &layer);

/*Test the rotation. It requires another buffer where the original image is stored.
*So use previous canvas as image and rotate it to the new canvas*/
LV_DRAW_BUF_DEFINE_STATIC(draw_buf_32bpp, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
LV_DRAW_BUF_INIT_STATIC(draw_buf_32bpp);

/*Create a canvas and initialize its palette*/
canvas = lv_canvas_create(lv_screen_active());
lv_canvas_set_draw_buf(canvas, &draw_buf_32bpp);
lv_canvas_fill_bg(canvas, lv_color_hex3(0xcccc), LV_OPA_COVER);
lv_obj_center(canvas);

lv_canvas_fill_bg(canvas, lv_palette_lighten(LV_PALETTE_GREY, 1), LV_OPA_COVER);

lv_canvas_init_layer(canvas, &layer);
lv_image_dsc_t img;
lv_draw_buf_to_image(&draw_buf_16bpp, &img);
lv_draw_image_dsc_t img_dsc;
lv_draw_image_dsc_init(&img_dsc);
img_dsc.rotation = 120;
img_dsc.src = &img;
img_dsc.pivot.x = CANVAS_WIDTH / 2;
img_dsc.pivot.y = CANVAS_HEIGHT / 2;

lv_area_t coords_img = {0, 0, CANVAS_WIDTH - 1, CANVAS_HEIGHT - 1};
lv_draw_image(&layer, &img_dsc, &coords_img);

lv_canvas_finish_layer(canvas, &layer);
}

#endif

```

Canvas trasparente con chroma key

```

#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 80
#define CANVAS_HEIGHT 40

/***
 * Create a transparent canvas with transparency
 */
void lv_example_canvas_2(void)
{
    lv_obj_set_style_bg_color(lv_screen_active(), lv_palette_lighten(LV_PALETTE_RED, 5), 0);

    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_obj_center(canvas);

    /*Red background (There is no dedicated alpha channel in indexed images so LV_OPA_COVER is ignored)*/
    lv_canvas_fill_bg(canvas, lv_palette_main(LV_PALETTE_BLUE), LV_OPA_COVER);

    /*Create hole on the canvas*/
    int32_t x;
    int32_t y;
    for(y = 10; y < 20; y++) {
        for(x = 5; x < 75; x++) {
            lv_canvas_set_px(canvas, x, y, lv_palette_main(LV_PALETTE_BLUE), LV_OPA_50);
        }
    }

    for(y = 20; y < 30; y++) {
        for(x = 5; x < 75; x++) {
            lv_canvas_set_px(canvas, x, y, lv_palette_main(LV_PALETTE_BLUE), LV_OPA_20);
        }
    }

    for(y = 30; y < 40; y++) {
        for(x = 5; x < 75; x++) {
            lv_canvas_set_px(canvas, x, y, lv_palette_main(LV_PALETTE_BLUE), LV_OPA_0);
        }
    }
}
#endif

```

Disegnare un rettangolo sul canvas

```

#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw a rectangle to the canvas
 */
void lv_example_canvas_3(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

```

(continues on next page)

(continua dalla pagina precedente)

```

/*Create a canvas and initialize its palette*/
lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
lv_canvas_set_draw_buf(canvas, &draw_buf);

lv_canvas_fill_bg(canvas, lv_color_hex3(0xcccc), LV_OPA_COVER);
lv_obj_center(canvas);

lv_layer_t layer;
lv_canvas_init_layer(canvas, &layer);

lv_draw_rect_dsc_t dsc;
lv_draw_rect_dsc_init(&dsc);
dsc.bg_color = lv_palette_main(LV_PALETTE_RED);
dsc.border_color = lv_palette_main(LV_PALETTE_BLUE);
dsc.border_width = 3;
dsc.outline_color = lv_palette_main(LV_PALETTE_GREEN);
dsc.outline_width = 2;
dsc.outline_paa = 2;
dsc.outline_opa = LV_OPA_50;
dsc.radius = 5;
dsc.border_width = 3;

lv_area_t coords = {10, 10, 40, 30};

lv_draw_rect(&layer, &dsc, &coords);

lv_canvas_finish_layer(canvas, &layer);
}
#endif

```

Disegnare un'etichetta sul canvas

```

#include "../lv_examples.h"
#if LV_USE_CANVAS && LV_FONT_MONTserrat_18 && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/**
 * Draw a text to the canvas
 */
void lv_example_canvas_4(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(&draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_label_dsc_t dsc;
    lv_draw_label_dsc_init(&dsc);
    dsc.color = lv_palette_main(LV_PALETTE_RED);
    dsc.font = &lv_font_montserrat_18;
    dsc.decor = LV_TEXT_DECOR_UNDERLINE;
    dsc.text = "Hello";

    lv_area_t coords = {10, 10, 30, 60};

    lv_draw_label(&layer, &dsc, &coords);

    lv_canvas_finish_layer(canvas, &layer);
}
#endif

```

Disegnare un arco sul canvas

```

#include "../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/**
 * Draw an arc to the canvas
 */
void lv_example_canvas_5(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(&draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_arc_dsc_t dsc;
    lv_draw_arc_dsc_init(&dsc);
    dsc.color = lv_palette_main(LV_PALETTE_RED);
    dsc.center.x = 25;
    dsc.center.y = 25;

```

(continues on next page)

(continua dalla pagina precedente)

```
dsc.width = 10;
dsc.radius = 15;
dsc.start_angle = 0;
dsc.end_angle = 220;

lv_draw_arc(&layer, &dsc);

lv_canvas_finish_layer(canvas, &layer);

}
#endif
```

Disegnare un'immagine sul canvas

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw an image to the canvas
 */
void lv_example_canvas_6(void)
{
    /*Create a buffer for the canvas*/
    static uint8_t cbuf[LV_CANVAS_BUF_SIZE(CANVAS_WIDTH, CANVAS_HEIGHT, 32, LV_DRAW_BUF_STRIDE_ALIGN)];

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_buffer(canvas, cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    LV_IMAGE_DECLARE(img_star);
    lv_draw_image_dsc_t dsc;
    lv_draw_image_dsc_init(&dsc);
    dsc.src = &img_star;

    lv_area_t coords = {10, 10, 10 + img_star.header.w - 1, 10 + img_star.header.h - 1};

    lv_draw_image(&layer, &dsc, &coords);

    lv_canvas_finish_layer(canvas, &layer);
}
#endif
```

Disegnare una linea sul canvas

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw a line to the canvas
 */
void lv_example_canvas_7(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(&draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_line_dsc_t dsc;
    lv_draw_line_dsc_init(&dsc);
    dsc.color = lv_palette_main(LV_PALETTE_RED);
    dsc.width = 4;
    dsc.round_end = 1;
    dsc.round_start = 1;
    dsc.p1.x = 15;
    dsc.p1.y = 15;
    dsc.p2.x = 35;
    dsc.p2.y = 10;
    lv_draw_line(&layer, &dsc);

    lv_canvas_finish_layer(canvas, &layer);
}
#endif
```

Disegnare un'immagine vettoriale sul canvas

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#if LV_USE_VECTOR_GRAPHIC
```

(continues on next page)

(continua dalla pagina precedente)

```
#define CANVAS_WIDTH 150
#define CANVAS_HEIGHT 150

/***
 * Draw a path to the canvas
 */
void lv_example_canvas_8(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_vector_dsc_t * dsc = lv_draw_vector_dsc_create(&layer);
    lv_vector_path_t * path = lv_vector_path_create(LV_VECTOR_PATH_QUALITY_MEDIUM);

    lv_fpoint_t pts[] = {{10, 10}, {130, 130}, {10, 130}};
    lv_vector_path_move_to(path, &pts[0]);
    lv_vector_path_line_to(path, &pts[1]);
    lv_vector_path_line_to(path, &pts[2]);
    lv_vector_path_close(path);

    lv_draw_vector_dsc_set_fill_color(dsc, lv_color_make(0x00, 0x80, 0xff));
    lv_draw_vector_dsc_add_path(dsc, path);

    lv_draw_vector(dsc);
    lv_vector_path_delete(path);
    lv_draw_vector_dsc_delete(dsc);

    lv_canvas_finish_layer(canvas, &layer);
}
#else

void lv_example_canvas_8(void)
{
    /*fallback for online examples*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Vector graphics is not enabled");
    lv_obj_center(label);
}

#endif /*LV_USE_VECTOR_GRAPHIC*/
#endif
```

Disegnare un triangolo sul canvas

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 150
#define CANVAS_HEIGHT 150

/***
 * Draw a triangle to the canvas
 */
void lv_example_canvas_9(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_triangle_dsc_t tri_dsc;
    lv_draw_triangle_dsc_init(&tri_dsc);
    tri_dsc.p[0].x = 10;
    tri_dsc.p[0].y = 10;
    tri_dsc.p[1].x = 100;
    tri_dsc.p[1].y = 30;
    tri_dsc.p[2].x = 50;
    tri_dsc.p[2].y = 100;

    tri_dsc.grad.stops_count = 2;
    tri_dsc.grad.dir = LV_GRAD_DIR_VER;
    tri_dsc.grad.stops[0].color = lv_color_hex(0xffff00);
    tri_dsc.grad.stops[0].frac = 64; /*Start at 25%*/
    tri_dsc.grad.stops[0].opa = LV_OPA_COVER;
    tri_dsc.grad.stops[1].color = lv_color_hex(0x0000ff);
    tri_dsc.grad.stops[1].opa = LV_OPA_TRANSP;
    tri_dsc.grad.stops[1].frac = 3 * 64; /*End at 75%*/

    tri_dsc.opa = 128; /*Set the overall opacity to 50%*/

    lv_draw_triangle(&layer, &tri_dsc);

    lv_canvas_finish_layer(canvas, &layer);
}

#endif
```

Disegnare Effetti di Lettere Fantasiose

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 300
#define CANVAS_HEIGHT 200

static void timer_cb(lv_timer_t * timer)
{
    static int16_t counter = 0;
    const char * string = "lol- I'm wawvvvvving>>>";
    const int16_t string_len = lv_strlen(string);

    lv_obj_t * canvas = (lv_obj_t *)lv_timer_get_user_data(timer);
    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_canvas_fill_bg(canvas, lv_color_white(), LV_OPA_COVER);

    lv_draw_letter_dsc_t letter_dsc;
    lv_draw_letter_dsc_init(&letter_dsc);
    letter_dsc.color = lv_color_hex(0xff0000);
    letter_dsc.font = lv_font_get_default();

    {
#define CURVE2_X(t) (t * 2 + 10)
#define CURVE2_Y(t) (lv_trigo_sin((t) * 5) * 40 / 32767 + CANVAS_HEIGHT / 2)

        int32_t pre_x = CURVE2_X(-1);
        int32_t pre_y = CURVE2_Y(-1);
        for(int16_t i = 0; i < string_len; i++) {
            const int16_t angle = (int16_t)(i * 5);
            const int32_t x = CURVE2_X(angle);
            const int32_t y = CURVE2_Y(angle + counter / 2);
            const lv_point_t point = { .x = x, .y = y };

            letter_dsc.unicode = (uint32_t)string[i % string_len];
            letter_dsc.rotation = lv_atan2(y - pre_y, x - pre_x) * 10;
            letter_dsc.color = lv_color_hsv_to_rgb(i * 10, 100, 100);
            lv_draw_letter(&layer, &letter_dsc, &point);

            pre_x = x;
            pre_y = y;
        }
    }

    lv_canvas_finish_layer(canvas, &layer);
    counter++;
}

void lv_example_canvas_10(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_obj_set_size(canvas, CANVAS_WIDTH, CANVAS_HEIGHT);

    lv_obj_center(canvas);

    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_timer_create(timer_cb, 16, canvas);
}
#endif
```

Disegnare Effetti di Lettere Fantasiose 2

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 300
#define CANVAS_HEIGHT 200

static void timer_cb(lv_timer_t * timer)
{
    static int32_t counter = 0;
    const char * string = "windstorrrrrrrrrrrrrm->>";
    const int16_t string_len = lv_strlen(string);

    lv_obj_t * canvas = (lv_obj_t *) lv_timer_get_user_data(timer);
    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_canvas_fill_bg(canvas, lv_color_white(), LV_OPA_COVER);

    lv_draw_letter_dsc_t letter_dsc;
    lv_draw_letter_dsc_init(&letter_dsc);
    letter_dsc.color = lv_color_hex(0xff0000);
    letter_dsc.font = lv_font_get_default();

    {
#define CURVE2_X(t) ((t) * 2 + lv_trigo_cos((t) * 5) * 40 / 32767 - 10)
#define CURVE2_Y(t, T) ((t) * lv_trigo_sin(((t) + (T)) * 5) * 40 / 32767 / 80 + CANVAS_HEIGHT / 2)

        int32_t pre_x = CURVE2_X(-1);
        int32_t pre_y = CURVE2_Y(-1, 0);
        for(int16_t i = 0; i < string_len; i++) {
            const int32_t angle = i * 5;
            const int32_t x = CURVE2_X(angle);
            const int32_t y = CURVE2_Y(angle + 30, counter / 2);

            letter_dsc.unicode = (uint32_t)string[i % string_len];
            letter_dsc.rotation = lv_atan2(y - pre_y, x - pre_x) * 10;
            letter_dsc.color = lv_color_hsv_to_rgb(i * 10, 100, 100);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_point_t p = (lv_point_t) {
    .x = x, .y = y
};
lv_draw_letter(&layer, &letter_dsc, &p);

pre_x = x;
pre_y = y;
}

lv_canvas_finish_layer(canvas, &layer);

counter++;
}

void lv_example_canvas_11(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_obj_set_size(canvas, CANVAS_WIDTH, CANVAS_HEIGHT);

    lv_obj_center(canvas);

    lv_canvas_set_draw_buf(canvas, &draw_buf);

    lv_timer_create(timer_cb, 16, canvas);
}
#endif

```

6.10.7 API

lv_canvas.h

lv_canvas_private.h

6.11 Grafici (lv_chart)

6.11.1 Panoramica

I Widget "Chart" [grafici] vengono utilizzati per visualizzare i dati.

I Chart hanno:

- 0 o più linee di divisione di sfondo (orizzontali e verticali),
- 4 assi interni, ognuno dei quali può essere utilizzato per specificare la scala per una serie di dati,
- un `point_count` (10 per default) che si applica a tutte le serie di dati aggiunte,
- 0 o più serie di dati (è possibile aggiungerle o rimuoverle in qualsiasi momento),
- 0 o più cursori (è possibile aggiungerli o rimuoverli in qualsiasi momento),
- modalità di aggiornamento (modifica il comportamento di `lv_chart_set_next_value()` se utilizzato).

I Chart possono:

- visualizzare assi flessibili con tacche e testo
- mostrare posizioni precise di punti o altre posizioni sul grafico con i cursori
- mostrare o nascondere singole serie di dati
- mostrare o nascondere singoli punti dati
- mostrare o nascondere i cursori
- eseguire scroll e zoom

6.11.2 Nozioni di Base sui Chart

Ogni chart ha i seguenti attributi (oltre agli attributi presenti in *tutti i Widget*):

Tipo (regola il modo in cui vengono disegnate le serie di dati di un grafico)

- Può essere LINE (il default), BAR, STACKED, SCATTER o NONE.

- È possibile modificare il tipo di grafico in qualsiasi momento durante la sua implementazione.

Linee di Divisione Orizzontali e Verticali

- rispettivamente 3 e 5 di default
- può essere qualsiasi valore non negativo, incluso 0

2 assi Y e 2 assi X (questi ultimi vengono utilizzati con i grafici SCATTER [a dispersione])

- Tutti e 4 gli assi sono inclusi automaticamente in ogni grafico (non è necessario crearli).
- I loro intervalli predefiniti sono [0..100]. Se è necessario un intervallo diverso, impostarlo prima di disegnare il grafico.
- Un asse si "utilizza" associandolo a una serie di dati, cosa che avviene al momento della creazione della serie di dati (maggiori dettagli di seguito). È possibile associare più serie di dati a ciascun asse.

Conteggio punti (numero di punti dati in tutte le serie di dati aggiunte al grafico)

- il default è 10
- Se si forniscono array di valori dati personalizzati, ogni array fornito deve contenere almeno questo numero di valori.
- Per i grafici LINE, BAR, STACKED, questo è il numero di punti sull'asse X.
- I grafici LINE, BAR, STACKED richiedono un solo array di valori dati per fornire i valori Y per ogni punto dati.
- Per i grafici a dispersione, questo è il numero di punti a dispersione nella serie di dati.
- I grafici a dispersione hanno array di valori dati separati sia per i valori X che per i valori Y.

Qualsiasi numero di serie di dati

- Dopo la creazione, un grafico inizialmente non contiene alcuna serie di dati. Si devono aggiungere.
- È possibile aggiungere e rimuovere serie di dati in qualsiasi momento durante la vita di un grafico.
- Quando viene creata una serie di dati, questa viene fornita con array di valori pre-allocati in base al tipo di grafico e al `point_count`. (Tutti i tipi di grafico utilizzano un array di valori Y. Anche i grafici di tipo SCATTER utilizzano un array di valori X). Tutti i valori Y così allocati vengono impostati su `LV_CHART_POINT_NONE`, il che fa sì che il punto venga nascosto.
- Per far sì che i punti vengano disegnati sul grafico, è necessario impostare i loro valori Y su un valore diverso da `LV_CHART_POINT_NONE`.
- È possibile nascondere un punto impostando il suo valore Y su `LV_CHART_POINT_NONE`.
- Si può indicare a una serie di dati di utilizzare invece un array di valori fornito. In tal caso:
 - Gli array di valori pre-allocati vengono automaticamente liberati.
 - Quella serie di dati continuerà a utilizzare *l'array fornito* da quel momento in poi.
 - I valori nell'array devono rimanere disponibili per tutta la durata di vita di quella serie di dati.
 - È necessario assicurarsi che ogni array fornito contenga almeno `point_count int32_t` elementi.
 - La gestione della durata di vita di tutti gli array di valori forniti è a discrezione dell'utente.

Un numero qualsiasi di cursori

- Dopo la creazione di un grafico, inizialmente non contiene cursori. È necessario aggiungerli se si desidera utilizzarli.
- È possibile aggiungere, mostrare, nascondere o rimuovere i cursori in qualsiasi momento durante la vita di un grafico.

Modalità di aggiornamento

- Vedere di seguito

Layer di Chart

Quando si disegna un grafico, alcuni elementi appaiono sopra altri, in questo ordine, dal basso verso l'alto:

- Lo sfondo del grafico (con linee di divisione opzionali)
- Ogni serie di dati:
 - Le prime serie di dati aggiunte appaiono in alto.
 - Per un grafico a dispersione, all'interno di ogni serie, i punti successivi nella sequenza appariranno sopra i punti precedenti nella sequenza in caso di sovrapposizione.
- Ogni cursore (se presente):
 - Il cursore aggiunto più di recente appare in alto.

6.11.3 Parti e Stili

- *LV_PART_MAIN* Lo sfondo del grafico. Utilizza le proprietà *tipiche del background* e stile della linea (per le linee di divisione). Il *Padding* riduce l'area della serie. Per i grafici BAR e STACKED *pad_column* imposta lo spazio tra le barre nella stessa serie di dati.
- *LV_PART_SCROLLBAR* Una scrollbar utilizzata se il grafico è ingrandito. Vedere la documentazione del *Widget Base (lv_obj)*.
- *LV_PART_ITEMS* Si riferisce alle serie di dati LINE o BAR.
 - Grafico a LINEE: le proprietà *line* sono utilizzate dalle linee. *width*, *height*, *bg_color* e *radius* vengono utilizzate per impostare l'aspetto dei punti sulla linea.
 - Grafico a barre: le tipiche proprietà di sfondo vengono utilizzate per definire lo stile delle barre. *pad_column* imposta lo spazio tra le colonne nella stessa serie di dati.
- *LV_PART_INDICATOR* Si riferisce ai punti sui grafici a LINEE e SCATTER (piccoli cerchi o quadrati [con angoli eventualmente arrotondati]).
- *LV_PART_CURSOR* Le proprietà *Line* vengono utilizzate per definire lo stile dei cursori. *width*, *height*, *bg_color* e *radius* vengono utilizzati per impostare l'aspetto del "punto" del cursore, indicandone la posizione. Se *width* o *height* sono impostati su 0, vengono disegnate solo le linee del cursore.

6.11.4 Dettagli

Tipo di Chart

Un "chart" [grafico] può essere di uno dei seguenti tipi:

- *LV_CHART_TYPE_NONE*: Non visualizza alcun dato. Può essere utilizzato per nascondere i dati del grafico.
- *LV_CHART_TYPE_LINE*: Disegna linee tra i punti dati. I punti dati possono anche essere visualizzati se i relativi stili *width*, *height*, *bg_color* e *radius* (per *LV_PART_ITEMS*) sono impostati e sia *width* che *height* hanno valori diversi da zero.
- *LV_CHART_TYPE_BAR*: Disegna barre verticali singole per ogni punto di ogni serie.
- *LV_CHART_TYPE_STACKED*: Disegna barre verticali sovrapposte in cui più serie di dati vengono visualizzate come segmenti all'interno di una singola barra per ogni punto dati. Supporta solo valori positivi.
- *LV_CHART_TYPE_SCATTER*: il grafico X/Y traccia i punti e, facoltativamente, le linee tra i punti se i valori dello stile di larghezza della linea per *LV_PART_ITEMS* sono diversi da zero e il valore Y del punto è diverso da *LV_CHART_POINT_NONE*. (Il disegno di singoli punti su un grafico a dispersione può essere soppresso se i loro valori Y sono impostati su *LV_CHART_POINT_NONE*).

I grafici iniziano la loro vita come grafici a linee. È possibile modificare il tipo di un grafico con *lv_chart_set_type(chart, LV_CHART_TYPE_...)*.

Serie di dati

È possibile aggiungere un numero qualsiasi di serie di dati a un grafico utilizzando

```
lv_chart_add_series(chart, color, axis).
```

Questo alloca (e restituisce un puntatore a) una struttura `lv_chart_series_t` che memorizza `color` e `axis` specificati e viene preallocata con un array di valori Y `chart->point_cnt int32_t`, tutti impostati su `LV_CHART_POINT_NONE`. (Un grafico a dispersione include anche un array preassegnato dello stesso numero di valori X).

`axis` specifica quale asse viene utilizzato per scalare i valori e può essere uno dei seguenti:

- `LV_CHART_AXIS_PRIMARY_Y`: Asse sinistro
- `LV_CHART_AXIS_SECONDARY_Y`: Asse destro
- `LV_CHART_AXIS_PRIMARY_X`: Asse inferiore
- `LV_CHART_AXIS_SECONDARY_X`: Asse superiore

Quando si aggiunge una serie di dati a un grafico a dispersione, è possibile utilizzare l'opzione "bit per bit" OPPURE l'asse Y selezionato (primario o secondario) con uno dei valori dell'asse X.

Se si desidera che il grafico utilizzi il proprio array di valori Y invece di quello fornito, lo si può fare con

```
lv_chart_set_series_ext_y_array(chart, series, value_array).
```

Si è responsabili di garantire che l'array fornito contenga almeno `chart->point_cnt` elementi.

`value_array` dovrebbe apparire così: `int32_t * value_array[num_points]`. Solo il puntatore dell'array viene salvato nella serie, quindi, il suo contenuto deve rimanere disponibile per tutta la durata della serie, ovvero l'array deve essere globale, statico o allocato dinamicamente.

Nota

Chiamare `lv_chart_refresh(chart)` quando i dati di un grafico sono cambiati per segnalare che il grafico deve essere nuovamente visualizzato al successivo aggiornamento del display. Non è necessario farlo se si utilizzano gli array di valori forniti e si impostano i valori con le funzioni `lv_chart_set_...value_...()`. Vedere di seguito per ulteriori informazioni su queste funzioni.

Un puntatore all'array di valori Y di una serie può essere ottenuto con `lv_chart_get_series_y_array(chart, series)`. Questo vale sia che si utilizzi l'array di valori Y fornito sia che se ne fornisca uno proprio.

Per i grafici di tipo SCATTER,

- Anche `lv_chart_set_series_ext_x_array(chart, series, value_array)` e
- `lv_chart_get_series_x_array(chart, series)`

Possono essere utilizzati.

Modifica dei dati

Sono disponibili diverse opzioni per impostare i valori Y per una serie di dati:

1. Impostare i valori a livello di codice nell'array, ad esempio `ser1->points[3] = 7` e aggiornare il grafico con `lv_chart_refresh(chart)`.
2. Utilizzare `lv_chart_set_series_value_by_id(chart, series, id, value)` dove `id` è l'indice a partire da zero del punto che si desidera aggiornare.
3. Usare `lv_chart_set_next_value(chart, series, value)`. (Vedere le *Modalità di aggiornamento* di seguito.)
4. Impostare tutti i punti su un singolo valore Y con `lv_chart_set_all_values(chart, series, value)`.

Usare `LV_CHART_POINT_NONE` come valore per fare in modo che la libreria salti il disegno di quel punto, colonna o punto di dispersione.

Per i grafici di tipo SCATTER,

- `lv_chart_set_series_value_by_id2(chart, series, id, x_value, y_value)` e
- `lv_chart_set_next_value2(chart, series, x_value, y_value)`

Possono essere utilizzati.

Modalità di aggiornamento

`lv_chart_set_next_value()` può comportarsi in due modi a seconda della *modalità di aggiornamento*:

- `LV_CHART_UPDATE_MODE_SHIFT`: Sposta i vecchi dati a sinistra e aggiunge quelli nuovi a destra.
- `LV_CHART_UPDATE_MODE_CIRCULAR`: Aggiunge i nuovi dati in modo circolare, come un diagramma ECG.

La modalità di aggiornamento può essere modificata con `lv_chart_set_update_mode(chart, LV_CHART_UPDATE_MODE_...)`.

Numero di punti

Il numero di punti nella serie può essere modificato con `lv_chart_set_point_count(chart, point_num)`. Il valore di default è 10. Nota: questo influenza sul numero di punti elaborati quando un array di valori esterni viene assegnato a una serie; quindi è necessario assicurarsi che qualsiasi array esterno fornito contenga almeno `point_num` elementi.

Gestione di un numero elevato di punti

Nei grafici LINE, se il numero di punti è maggiore dei pixel in orizzontale, il grafico disegnerà solo linee verticali per rendere efficace la visualizzazione di grandi quantità di dati. Se, ad esempio, ci sono 10 punti per pixel, LVGL cerca il valore più piccolo e quello più grande e disegna delle linee verticali tra di essi per garantire che non vengano trascurati picchi.

Range verticale

È possibile specificare i valori minimo e massimo in direzione Y con `lv_chart_set_axis_range(chart, axis, min, max)`. `axis` può essere `LV_CHART_AXIS_PRIMARY_Y` (asse Y sinistro) o `LV_CHART_AXIS_SECONDARY_Y` (asse Y destro).

Il valore dei punti verrà scalato proporzionalmente. L'intervallo di default è 0..100.

Linee di divisione

Il numero di linee di divisione orizzontali e verticali può essere modificato tramite `lv_chart_set_div_line_count(chart, hdiv_num, vdiv_num)`. Le impostazioni di default sono 3 linee di divisione orizzontali e 5 verticali. Se è presente un bordo visibile su un lato e non è presente alcun "padding" [riempimento], la linea di divisione verrà disegnata sopra il bordo e in questo caso non verrà disegnata per non nascondere il bordo del grafico.

Ignorare il punto di inizio di default per le serie

Per far sì che un grafico inizi da un punto diverso da quello predefinito, ovvero `point[0]` della serie, è possibile impostare un indice alternativo con la funzione `lv_chart_set_x_start_point(chart, series, id)` dove `id` la nuova posizione dell'indice a base zero da cui iniziare il grafico.

Si noti che `LV_CHART_UPDATE_MODE_SHIFT` modifica anche `start_point`.

Segni di graduazione ed etichette

Con l'aiuto di `Scale`, è possibile aggiungere scale verticali e orizzontali in modo molto flessibile. Vedere l'[esempio 2](#) qui sotto per saperne di più.

Zoom

Per ingrandire il grafico, è sufficiente inserirlo in un contenitore padre e impostare la larghezza e/o l'altezza del grafico su un valore maggiore. In questo modo, il grafico sarà "scrollable" [scorrevole] nel suo contenitore padre, ovvero il contenitore padre fornisce la "finestra di visualizzazione" scorrevole.

Cursore

Inizialmente, a un nuovo cursore viene assegnata la posizione `LV_CHART_POINT_NONE`, che lo nasconde. Per visualizzare il cursore, la sua posizione deve essere impostata da programma utilizzando una delle funzioni seguenti.

È possibile nascondere un cursore senza rimuoverlo dal grafico utilizzando `lv_chart_set_cursor_point()` passando `LV_CHART_POINT_NONE` come ID del punto.

È possibile aggiungere un cursore con `lv_chart_cursor_t * c1 = lv_chart_add_cursor(chart, color, dir);`. I possibili valori di `dir` sono i valori di enumerazione di `lv_dir_t`: `LV_DIR_NONE/RIGHT/UP/LEFT/DOWN/HOR/VER/ALL` o i loro valori in OR bit a bit per indicare al grafico in quale direzione tracciare le linee.

`lv_chart_set_cursor_pos(chart, cursor, &point)` imposta la posizione del cursore su un punto arbitrario del grafico. `&point` è un puntatore a una variabile `lv_point_t`. Ad esempio, `lv_point_t point = {10, 20}`. Se si scorre il grafico, il cursore si sposta con esso.

`lv_chart_get_point_pos_by_id(chart, series, id, &point_out)` ottiene le coordinate di un dato punto del grafico. Questo è utile per posizionare il cursore su quel punto dati.

`lv_chart_set_cursor_point(chart, cursor, series, point_id)` posiziona il cursore sul punto dati specificato sul grafico. Se la posizione del punto cambia (tramite un nuovo valore o tramite scorrimento), il cursore si sposta con il punto. Vedere un esempio di utilizzo di questa funzione *qui*.

6.11.5 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato quando viene premuto un nuovo punto sul grafico. `lv_chart_get_pressed_point(chart)` restituisce l'indice a partire da zero del punto premuto.

Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.11.6 Tasti

Nessun *tasto* viene elaborato dai Widget Chart.

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.11.7 Esempio

Grafico a linee

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

/***
 * A very basic line chart
 */
void lv_example_chart_1(void)
{
    /*Create a chart*/
    lv_obj_t * chart;
    chart = lv_chart_create(lv_screen_active());
    lv_obj_set_size(chart, 200, 150);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_center(chart);
lv_chart_set_type(chart, LV_CHART_TYPE_LINE); /*Show lines and points too*/

/*Add two data series*/
lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_GREEN), LV_CHART_AXIS_PRIMARY_Y);
lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_SECONDARY_Y);
int32_t * ser2_y_points = lv_chart_get_series_y_array(chart, ser2);

uint32_t i;
for(i = 0; i < 10; i++) {
    /*Set the next points on 'ser1'*/
    lv_chart_set_next_value(chart, ser1, (int32_t)lv_rand(10, 50));

    /*Directly set points on 'ser2'*/
    ser2_y_points[i] = (int32_t)lv_rand(50, 90);
}

lv_chart_refresh(chart); /*Required after direct set*/
}

#endif

```

Tick degli assi ed etichette con scrolling

```

#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

/**
 * Use lv_scale to add ticks to a scrollable chart
 */
void lv_example_chart_2(void)
{
    /*Create a container*/
    lv_obj_t * main_cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(main_cont, 200, 150);
    lv_obj_center(main_cont);

    /*Create a transparent wrapper for the chart and the scale.
     *Set a large width, to make it scrollable on the main container*/
    lv_obj_t * wrapper = lv_obj_create(main_cont);
    lv_obj_remove_style_all(wrapper);
    lv_obj_set_size(wrapper, lv_pct(300), lv_pct(100));
    lv_obj_set_flex_flow(wrapper, LV_FLEX_FLOW_COLUMN);

    /*Create a chart on the wrapper
     *Set it's width to 100% to fill the large wrapper*/
    lv_obj_t * chart = lv_chart_create(wrapper);
    lv_obj_set_width(chart, lv_pct(100));
    lv_obj_set_flex_grow(chart, 1);
    lv_chart_set_type(chart, LV_CHART_TYPE_BAR);
    lv_chart_set_axis_range(chart, LV_CHART_AXIS_PRIMARY_Y, 0, 100);
    lv_chart_set_axis_range(chart, LV_CHART_AXIS_SECONDARY_Y, 0, 400);
    lv_chart_set_point_count(chart, 12);
    lv_obj_set_style_radius(chart, 0, 0);

    /*Create a scale also with 100% width*/
    lv_obj_t * scale_bottom = lv_scale_create(wrapper);
    lv_scale_set_mode(scale_bottom, LV_SCALE_MODE_HORIZONTAL_BOTTOM);
    lv_obj_set_size(scale_bottom, lv_pct(100), 25);
    lv_scale_set_total_tick_count(scale_bottom, 12);
    lv_scale_set_major_tick_every(scale_bottom, 1);
    lv_obj_set_style_pad_hor(scale_bottom, lv_chart_get_first_point_center_offset(chart), 0);

    static const char * month[] = {"Jan", "Febr", "March", "Apr", "May", "Jun", "July", "Aug", "Sept", "Oct", "Nov", "Dec", NULL};
    lv_scale_set_text_src(scale_bottom, month);

    /*Add two data series*/
    lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_lighten(LV_PALETTE_GREEN, 2), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_darken(LV_PALETTE_GREEN, 2), LV_CHART_AXIS_PRIMARY_Y);

    /*Set the next points on 'ser1'*/
    uint32_t i;
    for(i = 0; i < 12; i++) {
        lv_chart_set_next_value(chart, ser1, (int32_t)lv_rand(10, 60));
        lv_chart_set_next_value(chart, ser2, (int32_t)lv_rand(50, 90));
    }

    lv_chart_refresh(chart); /*Required after direct set*/
}
#endif

```

Mostrare il valore dei punti premuti

```

#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * chart = lv_event_get_target_obj(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        lv_obj_invalidate(chart);
    }
    if(code == LV_EVENT_REFR_EXT_DRAW_SIZE) {
        int32_t * s = (int32_t *)lv_event_get_param(e);
        *s = LV_MAX(*s, 20);
    }
    else if(code == LV_EVENT_DRAW_POST_END) {
        int32_t id = lv_chart_get_pressed_point(chart);
        if(id == LV_CHART_POINT_NONE) return;

        LV_LOG_USER("Selected point %d", (int)id);
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_chart_series_t * ser = lv_chart_get_series_next(chart, NULL);
int32_t value = 0;
while(ser) {
    lv_point_t p;
    lv_chart_get_point_pos_by_id(chart, ser, id, &p);

    int32_t * y_array = lv_chart_get_series_y_array(chart, ser);
    if(y_array[id] != LV_CHART_POINT_NONE && y_array[id] >= 0) {

        /*Accumulate the values to show the rectangles at the top of each segment*/
        value += y_array[id];

        /*Draw a rectangle above the clicked point*/
        lv_layer_t * layer = lv_event_get_layer(e);
        lv_draw_rect_dsc_t draw_rect_dsc;
        lv_draw_rect_dsc_init(&draw_rect_dsc);
        draw_rect_dsc.bg_color = lv_color_black();
        draw_rect_dsc.bg_opa = LV_OPA_50;
        draw_rect_dsc.radius = 3;

        lv_area_t chart_obj_coords;
        lv_obj_get_coords(chart, &chart_obj_coords);
        lv_area_t rect_area;
        rect_area.x1 = chart_obj_coords.x1 + p.x - 20;
        rect_area.x2 = chart_obj_coords.x1 + p.x + 20;
        rect_area.y1 = chart_obj_coords.y1 + p.y - 10;
        rect_area.y2 = chart_obj_coords.y1 + p.y + 10;
        lv_draw_rect(layer, &draw_rect_dsc, &rect_area);

        /*Draw the value as label to the center of the rectangle*/
        char buf[16];
        lv_sprintf(buf, sizeof(buf), LV_SYMBOL_DUMMY"%d", value);

        lv_draw_label_dsc_t draw_label_dsc;
        lv_draw_label_dsc_init(&draw_label_dsc);
        draw_label_dsc.color = lv_color_white();
        draw_label_dsc.text = buf;
        draw_label_dsc.text_local = 1;
        draw_label_dsc.align = LV_TEXT_ALIGN_CENTER;
        lv_area_t label_area = rect_area;
        lv_area_set_height(&label_area, lv_font_get_line_height(draw_label_dsc.font));
        lv_area_align(&rect_area, &label_area, LV_ALIGN_CENTER, 0, 0);
        lv_draw_label(layer, &draw_label_dsc, &label_area);
    }

    ser = lv_chart_get_series_next(chart, ser);
}
else if(code == LV_EVENT_RELEASED) {
    lv_obj_invalidate(chart);
}
}

/***
 * Show the value of the pressed points
*/
void lv_example_chart_3(void)
{
    /*Create a chart*/
    lv_obj_t * chart;
    chart = lv_chart_create(lv_screen_active());
    lv_obj_set_size(chart, 280, 180);
    lv_obj_set_style_pad_column(chart, 4, 0);
    lv_obj_center(chart);
    lv_chart_set_type(chart, LV_CHART_TYPE_STACKED);

    lv_obj_add_event_cb(chart, event_cb, LV_EVENT_ALL, NULL);
    lv_obj_refresh_ext_draw_size(chart);

    /*Add two data series*/
    lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_GREEN), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser3 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_BLUE), LV_CHART_AXIS_PRIMARY_Y);
    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_chart_set_next_value(chart, ser1, lv_rand(15, 30));
        lv_chart_set_next_value(chart, ser2, lv_rand(15, 30));
        lv_chart_set_next_value(chart, ser3, lv_rand(15, 30));
    }
}
#endif

```

Ricolorare le barre in base al loro valore

```

#include "../../lv_examples.h"

#if LV_USE_CHART && LV_DRAW_SW_COMPLEX && LV_BUILD_EXAMPLES

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);

    if(base_dsc->part != LV_PART_ITEMS) {
        return;
    }

    lv_draw_fill_dsc_t * fill_dsc = lv_draw_task_get_fill_dsc(draw_task);
    if(fill_dsc) {
        lv_obj_t * chart = lv_event_get_target_obj(e);
        int32_t * y_array = lv_chart_get_series_y_array(chart, lv_chart_get_series_next(chart, NULL));
        int32_t v = y_array[base_dsc->i02];

        uint8_t ratio = (uint8_t)(v * 255 / 100);
        fill_dsc->color = lv_color_mix(lv_palette_main(LV_PALETTE_GREEN), lv_palette_main(LV_PALETTE_RED), ratio);
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```
/*
 * Recolor the bars of a chart based on their value
 */
void lv_example_chart_4(void)
{
    /*Create a chart*/
    lv_obj_t * chart = lv_chart_create(lv_screen_active());
    lv_chart_set_type(chart, LV_CHART_TYPE_BAR);
    lv_chart_set_point_count(chart, 24);
    lv_obj_set_style_pad_column(chart, 2, 0);
    lv_obj_set_size(chart, 260, 160);
    lv_obj_center(chart);

    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_color_hex(0xffff0000), LV_CHART_AXIS_PRIMARY_Y);
    lv_obj_add_event_cb(chart, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(chart, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);

    uint32_t i;
    for(i = 0; i < 24; i++) {
        lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 90));
    }
}

#endif
```

Grafico a linee sfumate con linee di divisione personalizzate

```
#include "../../lv_examples.h"

#if LV_USE_CHART && LV_DRAW_SW_COMPLEX && LV_BUILD_EXAMPLES
#include "../../../../lvgl_private.h"

static void hook_division_lines(lv_event_t * e);
static void add_faded_area(lv_event_t * e);
static void draw_event_cb(lv_event_t * e);

/***
 * Add a faded area effect to the line chart and make some division lines ticker
 */
void lv_example_chart_5(void)
{
    /*Create a chart*/
    lv_obj_t * chart = lv_chart_create(lv_screen_active());
    lv_chart_set_type(chart, LV_CHART_TYPE_LINE); /*Show lines and points too*/
    lv_obj_set_size(chart, 200, 150);
    lv_obj_set_style_pad_all(chart, 0, 0);
    lv_obj_set_style_radius(chart, 0, 0);
    lv_obj_center(chart);

    lv_chart_set_div_line_count(chart, 5, 7);

    lv_obj_add_event_cb(chart, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(chart, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);

    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);

    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 80));
    }
}

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);

    if(base_dsc->part == LV_PART_ITEMS && lv_draw_task_get_type(draw_task) == LV_DRAW_TASK_TYPE_LINE) {
        add_faded_area(e);
    }

    /*Hook the division lines too*/
    if(base_dsc->part == LV_PART_MAIN && lv_draw_task_get_type(draw_task) == LV_DRAW_TASK_TYPE_LINE) {
        hook_division_lines(e);
    }
}

static void add_faded_area(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_area_t coords;
    lv_obj_get_coords(obj, &coords);

    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);

    const lv_chart_series_t * ser = lv_chart_get_series_next(obj, NULL);
    lv_color_t ser_color = lv_chart_get_series_color(obj, ser);

    /*Draw a triangle below the line with some opacity gradient*/
    lv_draw_line_dsc_t * draw_line_dsc = (lv_draw_line_dsc_t *)lv_draw_task_get_draw_dsc(draw_task);
    lv_draw_triangle_dsc_t tri_dsc;

    lv_draw_triangle_dsc_init(&tri_dsc);
    tri_dsc.p[0].x = draw_line_dsc->p1.x;
    tri_dsc.p[0].y = draw_line_dsc->p1.y;
    tri_dsc.p[1].x = draw_line_dsc->p2.x;
    tri_dsc.p[1].y = draw_line_dsc->p2.y;
    tri_dsc.p[2].x = draw_line_dsc->p1.y < draw_line_dsc->p2.y ? draw_line_dsc->p1.x : draw_line_dsc->p2.x;
    tri_dsc.p[2].y = LV_MAX(draw_line_dsc->p1.y, draw_line_dsc->p2.y);
    tri_dsc.grad.dir = LV_GRAD_DIR_VER;

    int32_t full_h = lv_obj_get_height(obj);
    int32_t fract_uppter = (int32_t)(LV_MIN(draw_line_dsc->p1.y, draw_line_dsc->p2.y) - coords.y1) * 255 / full_h;
    int32_t fract_lower = (int32_t)(LV_MAX(draw_line_dsc->p1.y, draw_line_dsc->p2.y) - coords.y1) * 255 / full_h;
    tri_dsc.grad.stops[0].color = (lv_opa_t)(255 - fract_uppter);
    tri_dsc.grad.stops[0].opa = (lv_opa_t)(255 - fract_uppter);
```

(continues on next page)

(continua dalla pagina precedente)

```

tri_dsc.grad.stops[0].frac = 0;
tri_dsc.grad.stops[1].color = ser_color;
tri_dsc.grad.stops[1].opa = (lv_opa_t)(255 - fract_lower);
tri_dsc.grad.stops[1].frac = 255;

lv_draw_triangle(base_dsc->layer, &tri_dsc);

/*Draw rectangle below the triangle*/
lv_draw_rect_dsc_t rect_dsc;
lv_draw_rect_dsc_init(&rect_dsc);
rect_dsc.bg_grad.dir = LV_GRAD_DIR_VER;
rect_dsc.bg_grad.stops[0].color = ser_color;
rect_dsc.bg_grad.stops[0].frac = 0;
rect_dsc.bg_grad.stops[0].opa = (lv_opa_t)(255 + fract_lower);
rect_dsc.bg_grad.stops[1].color = ser_color;
rect_dsc.bg_grad.stops[1].frac = 255;
rect_dsc.bg_grad.stops[1].opa = 0;

lv_area_t rect_area;
rect_area.x1 = (int32_t)draw_line_dsc->p1.x;
rect_area.x2 = (int32_t)draw_line_dsc->p2.x - 1;
rect_area.y1 = (int32_t)LV_MAX(draw_line_dsc->p1.y, draw_line_dsc->p2.y);
rect_area.y2 = (int32_t)coords.y2;
lv_draw_rect(base_dsc->layer, &rect_dsc, &rect_area);

}

static void hook_division_lines(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    lv_draw_line_dsc_t * line_dsc = (lv_draw_line_dsc_t *)lv_draw_task_get_draw_dsc(draw_task);

    /*Vertical line*/
    if(line_dsc->p1.x == line_dsc->p2.x) {
        line_dsc->color = lv_palette_lighten(LV_PALETTE_GREY, 1);
        if(base_dsc->id1 == 3) {
            line_dsc->width = 2;
            line_dsc->dash_gap = 0;
            line_dsc->dash_width = 0;
        }
        else {
            line_dsc->width = 1;
            line_dsc->dash_gap = 6;
            line_dsc->dash_width = 6;
        }
    }
    /*Horizontal line*/
    else {
        if(base_dsc->id1 == 2) {
            line_dsc->width = 2;
            line_dsc->dash_gap = 0;
            line_dsc->dash_width = 0;
        }
        else {
            line_dsc->width = 2;
            line_dsc->dash_gap = 6;
            line_dsc->dash_width = 6;
        }

        if(base_dsc->id1 == 1 || base_dsc->id1 == 3) {
            line_dsc->color = lv_palette_main(LV_PALETTE_GREEN);
        }
        else {
            line_dsc->color = lv_palette_lighten(LV_PALETTE_GREY, 1);
        }
    }
}
#endif

```

Mostrare il cursore sul punto cliccato

```

#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

static lv_obj_t * chart;
static lv_chart_series_t * ser;
static lv_chart_cursor_t * cursor;

static void value_changed_event_cb(lv_event_t * e)
{
    uint32_t last_id;
    lv_obj_t * obj = lv_event_get_target_obj(e);

    last_id = lv_chart_get_pressed_point(obj);
    if(last_id != LV_CHART_POINT_NONE) {
        lv_chart_set_cursor_point(obj, cursor, NULL, last_id);
    }
}

/**
 * Show cursor on the clicked point
 */
void lv_example_chart_6(void)
{
    chart = lv_chart_create(lv_screen_active());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_align(chart, LV_ALIGN_CENTER, 0, -10);

    lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_Y, 10, 5, 6, 5, true, 40);
    // lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_X, 10, 5, 10, 1, true, 30);

    lv_obj_add_event_cb(chart, value_changed_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_refresh_ext_draw_size(chart);

    cursor = lv_chart_add_cursor(chart, lv_palette_main(LV_PALETTE_BLUE), (lv_dir_t)(LV_DIR_LEFT | LV_DIR_BOTTOM));
    ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

uint32_t i;
for(i = 0; i < 10; i++) {
    lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 90));
}

//    lv_chart_set_scale_x(chart, 500);

lv_obj_t * label = lv_label_create(lv_screen_active());
lv_label_set_text(label, "Click on a point");
lv_obj_align_to(label, chart, LV_ALIGN_OUT_TOP_MID, 0, -5);
}

#endif

```

Grafico a dispersione

```

#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    if(base_dsc->part == LV_PART_INDICATOR) {
        lv_obj_t * obj = lv_event_get_target_obj();
        lv_chart_series_t * ser = lv_chart_get_series_next(obj, NULL);
        lv_draw_rect_dsc_t * rect_draw_dsc = (lv_draw_rect_dsc_t *)lv_draw_task_get_draw_dsc(draw_task);
        uint32_t cnt = lv_chart_get_point_count(obj);

        /*Make older value more transparent/
        rect_draw_dsc->bg_opa = (lv_opa_t)((LV_OPA_COVER * base_dsc->id2) / (cnt - 1));

        /*Make smaller values blue, higher values red*/
        int32_t * x_array = lv_chart_get_series_x_array(obj, ser);
        int32_t * y_array = lv_chart_get_series_y_array(obj, ser);
        /*dsc->i is the tells drawing order, but we need the ID of the point being drawn.*/
        uint32_t start_point = lv_chart_get_x_start_point(obj, ser);
        uint32_t p_act = (start_point + base_dsc->id2) % cnt; /*Consider start point to get the index of the array*/
        lv_opa_t x_opa = ((x_array[p_act] * LV_OPA_50) / 200);
        lv_opa_t y_opa = ((y_array[p_act] * LV_OPA_50) / 1000);

        rect_draw_dsc->bg_color = lv_color_mix(lv_palette_main(LV_PALETTE_RED),
                                                lv_palette_main(LV_PALETTE_BLUE),
                                                x_opa + y_opa);
    }
}

static void add_data(lv_timer_t * timer)
{
    lv_obj_t * chart = (lv_obj_t *)lv_timer_get_user_data(timer);
    lv_chart_set_next_value2(chart, lv_chart_get_series_next(chart, NULL), (int32_t)lv_rand(0, 200),
                           (int32_t)lv_rand(0, 1000));
}

/***
 * A scatter chart
 */
void lv_example_chart_7(void)
{
    lv_obj_t * chart = lv_chart_create(lv_screen_active());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_align(chart, LV_ALIGN_CENTER, 0, 0);
    lv_obj_add_event_cb(chart, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(chart, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
    lv_obj_set_style_line_width(chart, 0, LV_PART_ITEMS); /*Remove the lines*/

    lv_chart_set_type(chart, LV_CHART_TYPE_SCATTER);

    lv_chart_set_axis_range(chart, LV_CHART_AXIS_PRIMARY_X, 0, 200);
    lv_chart_set_axis_range(chart, LV_CHART_AXIS_PRIMARY_Y, 0, 1000);

    lv_chart_set_point_count(chart, 50);

    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    uint32_t i;
    for(i = 0; i < 50; i++) {
        lv_chart_set_next_value2(chart, ser, (int32_t)lv_rand(0, 200), (int32_t)lv_rand(0, 1000));
    }

    lv_timer_create(add_data, 100, chart);
}
#endif

```

Grafico a linee circolari con spazio

```

#include "../../lv_examples.h"
#if LV_USE_CHART && LV_DRAW_SW_COMPLEX && LV_BUILD_EXAMPLES

static void add_data(lv_timer_t * t)
{
    lv_obj_t * chart = (lv_obj_t *)lv_timer_get_user_data(t);
    lv_chart_series_t * ser = lv_chart_get_series_next(chart, NULL);

    lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 90));

    uint32_t p = lv_chart_get_point_count(chart);
    uint32_t s = lv_chart_get_x_start_point(chart, ser);
    int32_t * a = lv_chart_get_series_y_array(chart, ser);

    a[(s + 1) % p] = LV_CHART_POINT_NONE;
    a[(s + 2) % p] = LV_CHART_POINT_NONE;
    a[(s + 2) % p] = LV_CHART_POINT_NONE;
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_chart_refresh(chart);
}

/**
 * Circular line chart with gap
 */
void lv_example_chart_8(void)
{
    /*Create a stacked_area_chart.obj*/
    lv_obj_t * chart = lv_chart_create(lv_screen_active());
    lv_chart_set_update_mode(chart, LV_CHART_UPDATE_MODE_CIRCULAR);
    lv_obj_set_style_size(chart, 0, 0, LV_PART_INDICATOR);
    lv_obj_set_size(chart, 280, 150);
    lv_obj_center(chart);

    lv_chart_set_point_count(chart, 80);
    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    /*Prefill with data*/
    uint32_t i;
    for(i = 0; i < 80; i++) {
        lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 90));
    }

    lv_timer_create(add_data, 300, chart);
}
#endif

```

6.11.8 API

lv_api_map_v9_1.h

lv_chart.h

lv_chart_private.h

6.12 Checkbox (lv_checkbox)

6.12.1 Panoramica

Il Widget Checkbox viene creato da una "tick box" [casella di controllo] e da un'etichetta. Quando si clicca sulla Checkbox, la casella di controllo viene attivata/disattivata.

6.12.2 Parti e Stili

- *LV_PART_MAIN* Sfondo della Checkbox e utilizza il testo e le *tipiche proprietà di stile del background*. *pad_column* regola la spaziatura tra la tickbox e l'etichetta.
- *LV_PART_INDICATOR* La "tick box" è un quadrato che utilizza le *tipiche proprietà di stile del background*. Per default, la sua dimensione è pari all'altezza del font della parte "main". Le proprietà di "padding" [riempimento] ingrandiscono la tick box nelle rispettive direzioni.

La Checkbox viene aggiunta al gruppo di default (se impostato).

6.12.3 Utilizzo

Text

Il testo può essere modificato con *lv_checkbox_set_text(cb, "New text")* e verrà allocato dinamicamente.

Per impostare un testo statico, utilizzare *lv_checkbox_set_text_static(cb, txt)*. In questo modo, verrà memorizzato solo un puntatore a *txt*. Il buffer di testo fornito deve rimanere disponibile per tutta la durata della Checkbox.

Selezionare, deselectare, disabilitare

È possibile il "check" [selezionare], "un-check" [deselezionare] e disabilitare la Checkbox tramite codice utilizzando la funzione di aggiunta/cancellazione dello stato comune:

```

lv_obj_add_state(cb, LV_STATE_CHECKED); /* Make Checkbox checked */
lv_obj_remove_state(cb, LV_STATE_CHECKED); /* Make Checkbox unchecked */
lv_obj_add_state(cb, LV_STATE_CHECKED); /* Make Checkbox checked */
lv_obj_add_state(cb, LV_STATE_DISABLED); /* Make Checkbox disabled */

```

Per scoprire se la Checkbox è selezionata, si usa `lv_obj_has_state(cb, LV_STATE_CHECKED)`.

6.12.4 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato quando la Checkbox viene attivata/disattivata.

Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.12.5 Tasti

I seguenti *Tasti* vengono elaborati dalla casella di controllo:

- `LV_KEY_RIGHT/UP` Passa allo stato CHECKED se la Checkbox è abilitata
- `LV_KEY_LEFT/DOWN` Passa allo stato non-CHECKED se la Checkbox è abilitata
- `LV_KEY_ENTER` Clicca sulla Checkbox e ne cambia il valore.

Si noti che, come di consueto, lo stato di `LV_KEY_ENTER` viene tradotto in `LV_EVENT_PRESSED/PRESSING/RELEASED` ecc.

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.12.6 Esempio

Checkbox Semplici

```
#include "../../lv_examples.h"
#if LV_USE_CHECKBOX && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        LV_UNUSED(obj);
        const char * txt = lv_checkbox_get_text(obj);
        const char * state = lv_obj_get_state(obj) & LV_STATE_CHECKED ? "Checked" : "Unchecked";
        LV_UNUSED(txt);
        LV_UNUSED(state);
        LV_LOG_USER("%s: %s", txt, state);
    }
}

void lv_example_checkbox_1(void)
{
    lv_obj_set_flex_flow(lv_screen_active(), LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(lv_screen_active(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER);

    lv_obj_t * cb;
    cb = lv_checkbox_create(lv_screen_active());
    lv_checkbox_set_text(cb, "Apple");
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_screen_active());
    lv_checkbox_set_text(cb, "Banana");
    lv_obj_add_state(cb, LV_STATE_CHECKED);
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_screen_active());
    lv_checkbox_set_text(cb, "Lemon");
    lv_obj_add_state(cb, LV_STATE_DISABLED);
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_screen_active());
    lv_obj_add_state(cb, LV_STATE_CHECKED);
    lv_obj_add_state(cb, LV_STATE_DISABLED);
    lv_checkbox_set_text(cb, "Melon\nand a new line");
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    lv_obj_update_layout(cb);
}

#endif
```

Checkbox come radio button

```
#include "../../lv_examples.h"
#if LV_USE_CHECKBOX && LV_BUILD_EXAMPLES

static lv_style_t style_radio;
static lv_style_t style_radio_chk;
static int32_t active_index_1 = 0;
static int32_t active_index_2 = 0;

static void radio_event_handler(lv_event_t * e)
{
    int32_t * active_id = (int32_t *)lv_event_get_user_data(e);
    lv_obj_t * cont = (lv_obj_t *)lv_event_get_current_target(e);
    lv_obj_t * act_cb = lv_event_get_target_obj(e);
    lv_obj_t * old_cb = lv_obj_get_child(cont, *active_id);

    /*Do nothing if the container was clicked*/
    if(act_cb == cont) return;

    lv_obj_remove_state(old_cb, LV_STATE_CHECKED); /*Uncheck the previous radio button*/
    lv_obj_add_state(act_cb, LV_STATE_CHECKED); /*Check the current radio button*/

    *active_id = lv_obj_get_index(act_cb);

    LV_LOG_USER("Selected radio buttons: %d, %d, (%int)active_index_1, (%int)active_index_2");
}

static void radiobutton_create(lv_obj_t * parent, const char * txt)
{
    lv_obj_t * obj = lv_checkbox_create(parent);
    lv_checkbox_set_text(obj, txt);
    lv_obj_add_flag(obj, LV_OBJ_FLAG_EVENT_BUBBLE);
    lv_obj_add_style(obj, &style_radio, LV_PART_INDICATOR);
    lv_obj_add_style(obj, &style_radio_chk, LV_PART_INDICATOR | LV_STATE_CHECKED);
}

/**
 * Checkboxes as radio buttons
 */
void lv_example_checkbox_2(void)
{
    /* The idea is to enable `LV_OBJ_FLAG_EVENT_BUBBLE` on checkboxes and process the
     * `LV_EVENT_CLICKED` on the container.
     * A variable is passed as event user data where the index of the active
     * radiobutton is saved */

    lv_style_init(&style_radio);
    lv_style_set_radius(&style_radio, LV_RADIUS_CIRCLE);

    lv_style_init(&style_radio_chk);
    lv_style_set_bg_image_src(&style_radio_chk, NULL);

    uint32_t i;
    char buf[32];

    lv_obj_t * cont1 = lv_obj_create(lv_screen_active());
    lv_obj_set.flex_flow(cont1, LV_FLEX_FLOW_COLUMN);
    lv_obj.set.size(cont1, lv_pct(40), lv_pct(80));
    lv_obj.add.event_cb(cont1, radio_event_handler, LV_EVENT_CLICKED, &active_index_1);

    for(i = 0; i < 5; i++) {
        lv_snprintf(buf, sizeof(buf), "A %d", (int)i + 1);
        radiobutton_create(cont1, buf);
    }

    /*Make the first checkbox checked*/
    lv_obj_add_state(lv_obj_get_child(cont1, 0), LV_STATE_CHECKED);

    lv_obj_t * cont2 = lv_obj_create(lv_screen_active());
    lv_obj.set.flex_flow(cont2, LV_FLEX_FLOW_COLUMN);
    lv_obj.set.size(cont2, lv_pct(40), lv_pct(80));
    lv_obj.set.x(cont2, lv_pct(50));
    lv_obj.add.event_cb(cont2, radio_event_handler, LV_EVENT_CLICKED, &active_index_2);

    for(i = 0; i < 3; i++) {
        lv_snprintf(buf, sizeof(buf), "B %d", (int)i + 1);
        radiobutton_create(cont2, buf);
    }

    /*Make the first checkbox checked*/
    lv_obj_add_state(lv_obj_get_child(cont2, 0), LV_STATE_CHECKED);
}

#endif
```

6.12.7 API

lv_checkbox.h

lv_checkbox_private.h

6.13 Drop-Down List (lv_dropdown)

6.13.1 Panoramica

La Drop-Down List [elenco a discesa] consente all'utente di selezionare un valore da un elenco.

La Drop-Down List è chiusa per default e visualizza un singolo valore o un testo predefinito. Quando viene attivata (cliccando sulla Drop-Down List), viene creato un elenco da cui l'utente può selezionare un elemento. Quando l'utente seleziona un nuovo valore, l'elenco viene nuovamente eliminato.

La Drop-down List viene aggiunta al gruppo di default (se impostato). La Drop-Down list è un widget modificabile che consente la selezione degli elementi dell'elenco anche tramite codificatore o navigazione da tastiera.

6.13.2 Parti e Stili

Il Widget Drop-Down List è costituito dagli elementi: "button" e "list" (versioni semplificate dei Widget Button e List).

Pulsante

- `LV_PART_MAIN` Background del pulsante, usa le proprietà di stile *tipiche del background* e di text per il testo.
- `LV_PART_INDICATOR` In genere un simbolo a forma di freccia che può essere un'immagine o un testo (ad esempio `LV_SYMBOL`).

Il pulsante passa a `LV_STATE_CHECKED` quando viene aperto.

Lista

- `LV_PART_MAIN` La lista stessa; utilizza le *tipiche proprietà di stile del background*. `max_height` può essere utilizzato per limitare l'altezza della lista.
- `LV_PART_SCROLLBAR` Le proprietà di background, bordo, ombra e larghezza della barra di scorrimento (per la sua larghezza) e il padding destro per la spaziatura a destra.
- `LV_PART_SELECTED` Si riferisce all'opzione attualmente premuta, checked o premuta+checked. Utilizza anche le *tipiche proprietà di stile del background*.

L'elenco viene mostrato/nascosto all'apertura/chiusura. Per aggiungere stili, si usa `lv_dropdown_get_list`(dropdown) per ottenere l'oggetto elenco. Esempio:

```
lv_obj_t * list = lv_dropdown_get_list(dropdown) /* Get list */
lv_obj_add_style(list, &my_style, selector)      /* Add styles to list */
```

In alternativa, il tema può essere esteso con nuovi stili.

6.13.3 Utilizzo

Elementi dell'elenco

Gli elementi dell'elenco vengono passati alla Drop-Down List come elenco "newline-separated" in una stringa come argomento `options` a `lv_dropdown_set_options`(dropdown, options). Ogni elemento dell'elenco deve essere separato da \n. Esempio: "First\nSecond\nThird". Questa stringa viene copiata dalla Drop-Down List, quindi il suo contenuto non deve rimanere disponibile oltre questa chiamata.

La funzione `lv_dropdown_add_option`(dropdown, "New option", pos) inserisce una nuova opzione all'indice pos.

Per risparmiare memoria, le opzioni possono essere impostate anche da una stringa statica (const) con `lv_dropdown_set_options_static`(dropdown, options). In questo caso, il contenuto della stringa delle opzioni deve rimanere disponibile per tutta la durata della Drop-Down List e `lv_dropdown_add_option()` non può essere utilizzato.

È possibile selezionare un'opzione a livello di codice con `lv_dropdown_set_selected`(dropdown, id), dove id è l'indice dell'opzione target.

Prelevare l'opzione selezionata

Per ottenere l'*indice* dell'opzione selezionata, si usa `lv_dropdown_get_selected`(dropdown).

`lv_dropdown_get_selected_str`(dropdown, buf, buf_size) copia il *nome* dell'opzione selezionata in buf.

Direzione

L'elenco può essere creato su qualsiasi lato. Il default `LV_DIR_BOTTOM` può essere modificato con `lv_dropdown_set_dir`(dropdown, `LV_DIR_LEFT`).

Se l'elenco si troverà verticalmente fuori dallo schermo, verrà allineato al bordo.

Simbolo

Un simbolo (in genere una freccia) può essere aggiunto alla Drop-Down List con `lv_dropdown_set_symbol`(dropdown, `LV_SYMBOL_...`)

Se la direzione della Drop-Down List è `LV_DIR_LEFT` il simbolo verrà visualizzato a sinistra, altrimenti a destra.

Mostrare il selezionato

La parte principale può mostrare l'elemento selezionato o un testo statico. Se il testo statico è impostato con `lv_dropdown_set_text`(dropdown, "Some text"), verrà visualizzato indipendentemente dall'elemento selezionato. Se il testo è NULL, l'opzione selezionata viene visualizzata sul pulsante.

Apertura/chiusura programmatica

Per aprire o chiudere programmaticamente la Drop-Down List si usa `lv_dropdown_open`(dropdown) o `lv_dropdown_close`(dropdown).

Data binding

Per familiarizzare con "observer", "subject" e data binding in generale, visitare la pagina [Observer](#).

Questo metodo di sottoscrizione a un Subject intero influisce direttamente sul valore intero di un Widget Drop-Down. Si noti che si tratta di un'associazione bidirezionale (Soggetto <====> Widget), quindi l'interazione diretta di un utente finale con il Widget Drop-Down aggiorna il valore del Subject e viceversa.

Supporta solo Subject interi.

- `lv_dropdown_bind_value`(dropdown, &subject)

6.13.4 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato quando viene selezionata una nuova opzione o l'elenco viene aperto/chiuso.
- `LV_EVENT_CANCEL` Inviato quando l'elenco viene chiuso.
- `LV_EVENT_READY` Inviato quando l'elenco viene aperto.

i Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.13.5 Tasti

- `LV_KEY_RIGHT/DOWN` Seleziona l'elemento successivo dell'elenco.
- `LV_KEY_LEFT/UP` Seleziona l'elemento precedente dell'elenco.

- **LV_KEY_ENTER** Applica l'elemento dell'elenco selezionato (invia l'evento **LV_EVENT_VALUE_CHANGED** e chiude la Drop-Down List).

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.13.6 Esempio

Menù a discesa semplice

```
#include "../../lv_examples.h"
#if LV_USE_DROPDOWN && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_dropdown_get_selected_str(obj, buf, sizeof(buf));
        LV_LOG_USER("Option: %s", buf);
    }
}

void lv_example_dropdown_1(void)
{
    /*Create a normal drop down list*/
    lv_obj_t * dd = lv_dropdown_create(lv_screen_active());
    lv_dropdown_set_options(dd, "Apple\n"
                           "Banana\n"
                           "Orange\n"
                           "Cherry\n"
                           "Grape\n"
                           "Raspberry\n"
                           "Melon\n"
                           "Orangen\n"
                           "Lemon\n"
                           "Nuts");
    lv_obj_align(dd, LV_ALIGN_TOP_MID, 0, 20);
    lv_obj_add_event_cb(dd, event_handler, LV_EVENT_ALL, NULL);
}

#endif
```

Menù a discesa nelle quattro direzioni

```
#include "../../lv_examples.h"
#if LV_USE_DROPDOWN && LV_BUILD_EXAMPLES

/**
 * Create a drop down, up, left and right menus
 */
void lv_example_dropdown_2(void)
{
    static const char * opts = "Apple\n"
                             "Banana\n"
                             "Orange\n"
                             "Melon";
    lv_obj_t * dd;
    dd = lv_dropdown_create(lv_screen_active());
    lv_dropdown_set_options_static(dd, opts);
    lv_obj_align(dd, LV_ALIGN_TOP_MID, 0, 10);

    dd = lv_dropdown_create(lv_screen_active());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_BOTTOM);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_UP);
    lv_obj_align(dd, LV_ALIGN_BOTTOM_MID, 0, -10);

    dd = lv_dropdown_create(lv_screen_active());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_RIGHT);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_RIGHT);
    lv_obj_align(dd, LV_ALIGN_LEFT_MID, 10, 0);

    dd = lv_dropdown_create(lv_screen_active());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_LEFT);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_LEFT);
    lv_obj_align(dd, LV_ALIGN_RIGHT_MID, -10, 0);
}

#endif
```

Menù

```
#include "../../lv_examples.h"
#if LV_USE_DROPDOWN && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * dropdown = lv_event_get_target_obj(e);
    char buf[64];
    lv_dropdown_get_selected_str(dropdown, buf, sizeof(buf));
    LV_LOG_USER("%s' is selected", buf);
}

/** Create a menu from a drop-down list and show some drop-down list features and styling */
void lv_example_dropdown_3(void)
{
    /*Create a drop down list*/
    lv_obj_t * dropdown = lv_dropdown_create(lv_screen_active());
    lv_obj_align(dropdown, LV_ALIGN_TOP_LEFT, 10, 10);
    lv_dropdown_set_options(dropdown, "New project\n"
        "New file\n"
        "Save\n"
        "Save as ...\n"
        "Open project\n"
        "Recent projects\n"
        "Preferences\n"
        "Exit");

    /*Set a fixed text to display on the button of the drop-down list*/
    lv_dropdown_set_text(dropdown, "Menu");

    /*Use a custom image as down icon and flip it when the list is opened*/
    LV_IMAGE_DECLARE(img_caret_down);
    lv_dropdown_set_symbol(dropdown, &img_caret_down);
    lv_obj_set_style_transform_rotation(dropdown, 1800, LV_PART_INDICATOR | LV_STATE_CHECKED);

    /*In a menu we don't need to show the last clicked item*/
    lv_dropdown_set_selected_highlight(dropdown, false);

    lv_obj_add_event_cb(dropdown, event_cb, LV_EVENT_VALUE_CHANGED, NULL);
}
#endif
```

6.13.7 API

lv_api_map_v8.h

lv_dropdown.h

lv_dropdown_private.h

lv_obj_property_names.h

6.14 Image (lv_image)

6.14.1 Panoramica

Le immagini sono Widget che visualizzano immagini da Flash (come array) o da file. Le immagini possono anche visualizzare simboli (LV_SYMBOL_...).

Utilizzando *l'interfaccia di decodifica delle immagini*, è possibile supportare anche formati di immagine personalizzati.

6.14.2 Parti e Stili

- **LV_PART_MAIN** Un rettangolo di background che utilizza le *tipiche proprietà di stile del background* e l'immagine stessa utilizza le proprietà di stile dell'immagine.

6.14.3 Utilizzo

Sorgente dell'immagine

Per garantire la massima flessibilità, la sorgente dell'immagine può essere:

- una variabile nel codice (un array C contenente i pixel).
- un file memorizzato esternamente (ad esempio su una scheda SD).
- un *Simbolo* come testo.

Per impostare la sorgente di un'immagine, si usa `lv_image_set_src(img, src)`.

Per generare un array di pixel da un'immagine PNG, JPG o BMP, si usa il [Tool di conversione di immagini online](#) e si imposta l'immagine convertita come sorgente dell'immagine con il suo puntatore tramite `lv_image_set_src(img1, &converted_img_var)`. Per rendere accessibile la variabile immagine convertita dal file C, dichiararla con `LV_IMAGE_DECLARE(converted_img_var)`.

Per utilizzare file esterni, è necessario convertire i file immagine utilizzando lo strumento di conversione online, ma selezionando il formato di output binario. È inoltre necessario utilizzare il modulo file system di LVGL e registrare un driver con alcune funzioni per le operazioni di base sui file. Per ulteriori informazioni: [File system](#). Quindi impostare l'immagine tradotta come sorgente dell'immagine con `lv_image_set_src(img, "S:folder1/my_img.bin")`.

È anche possibile impostare un simbolo come sorgente dell'immagine, in modo simile a [Label](#). In questo caso, l'immagine verrà visualizzata come testo in base al *font* specificato nello stile. Permette l'uso di "caratteri" monocromatici leggeri al posto di immagini reali. È possibile impostare un simbolo come sorgente dell'immagine con `lv_image_set_src(img1, LV_SYMBOL_OK)`.

Etichetta come immagine

Immagini ed etichette vengono talvolta utilizzate per comunicare la stessa cosa, ad esempio per descrivere la funzione di un pulsante. In questo contesto, immagini ed etichette sono in qualche modo intercambiabili: le immagini possono visualizzare del testo utilizzando la macro `LV_SYMBOL_DUMMY` (che equivale a una stringa C di 3 byte contenente un codice speciale) come prefisso del testo. Ad esempio, `lv_image_set_src(img, LV_SYMBOL_DUMMY "Some text")`.

Trasparenza

Le immagini interne (array di pixel) ed esterne supportano 2 metodi di gestione della trasparenza:

- **Byte alpha:** Un canale alfa viene aggiunto a ogni pixel che ne contiene l'opacità, in genere un byte. È la 'A' nei vari formati colore che contengono un canale alfa, come ARGB8888, ARGB8565, ARGB1555, ecc.
- **Colore trasparente indicizzato:** un indice specifico in una tavolozza di colori serve a segnalare la trasparenza per ogni pixel che lo utilizza.

Palette e indice Alfa

Oltre ai formati colore RGB888 e ARGB8888, sono supportati i seguenti formati:

- **Indicizzato:** L'immagine ha una tavolozza colori e ogni pixel è un indice di tale tavolozza.
- **Indicizzato alfa:** I valori memorizzati nelle posizioni dei pixel sono valori alfa (opacità).

Queste opzioni possono essere selezionate nel convertitore di immagini. Per saperne di più, consultare la sezione [Formati Colore](#).

Recolor

Un colore può essere miscelato con ogni pixel di un'immagine con una determinata intensità. Questo può essere utile per mostrare diversi stati (selezionato, inattivo, premuto, ecc.) di un'immagine senza dover memorizzare più versioni della stessa immagine. Questa funzionalità può essere abilitata nello stile impostando `img_recolor_opa` tra `LV_OPA_TRANS` (nessuna ricolorazione, valore: 0) e `LV_OPA_COVER` (ricolorazione completa, valore: 255). Il valore di default è `LV_OPA_TRANS`, che disabilita questa funzionalità.

Il colore da miscelare è impostato da `img_recolor`.

Offset

Con `lv_image_set_offset_x(img, x_ofs)` e `lv_image_set_offset_y(img, y_ofs)`, è possibile aggiungere un offset all'immagine visualizzata. Utile se le dimensioni del Widget sono inferiori a quelle dell'immagine sorgente. Utilizzando il parametro offset è possibile creare una [Texture atlas](#) o un effetto "immagine in movimento" [Animando](#) l'offset x o y.

Trasformazioni

È possibile ingrandire o ridurre le immagini utilizzando `lv_image_set_scale(img, factor)`. Impostare `factor` a 256 o a `LV_SCALE_NONE` per disabilitare lo zoom. Un valore maggiore ingrandisce le immagini (ad esempio 512 ne raddoppia le dimensioni), un valore minore le rimpicciolisce (ad esempio 128 ne dimezza le dimensioni). Il ridimensionamento frazionario funziona utilizzando un valore proporzionalmente maggiore o minore, ad esempio 281 per un ingrandimento del 10%.

`lv_image_set_scale_x(img, factor)` e `lv_image_set_scale_y(img, factor)` possono essere utilizzati anche per impostare il ridimensionamento orizzontale e verticale in modo indipendente. Possono essere valori diversi.

Per ruotare l'immagine, si usa `lv_image_set_rotation(img, angle_x10)`. L'argomento `angle_x10` è un `int32_t` contenente l'angolo (in gradi) moltiplicato per 10. Questo fornisce una risoluzione di 0,1 gradi. Esempio: 458 significa 45,8°.

Per default il punto di rotazione è il centro dell'immagine. Questo può essere modificato con `lv_image_set_pivot(img, pivot_x, pivot_y)` dove le coordinate (0,0) rappresentano l'angolo in alto a sinistra.

La qualità della trasformazione può essere regolata con `lv_image_set_antialias(img, true)`. L'attivazione dell'anti-aliasing rende le trasformazioni di qualità superiore, ma più lente.

Le trasformazioni richiedono che l'intera immagine sia disponibile. Pertanto, le immagini indicizzate (`LV_COLOR_FORMAT_I1/2/4/8...`) e le immagini solo alfa non possono essere trasformate. In altre parole, le trasformazioni funzionano solo su immagini normali (A)RGB o A8 memorizzate come array C, o su immagini fornite da un *Decoder dell'Immagine* personalizzato che restituisce l'intera immagine.

Si noti che le coordinate reali dei Widget Image non cambiano con una trasformazione. Ciò significa che `lv_obj_get_width / height / x / y()` restituirà le coordinate originali, non ingrandite.

IMPORTANTE: La trasformazione dell'immagine è indipendente dalle proprietà di trasformazione *derivate dagli stili*. Le principali differenze sono che le trasformazioni pure del Widget Image:

- non trasformano gli elementi figlio del Widget Image e
- l'immagine viene trasformata direttamente senza creare un livello intermedio (buffer) per catturare l'immagine del Widget.

Allineamento interno

Per default, la larghezza e l'altezza del Widget "image" sono `LV_SIZE_CONTENT`, il che significa che il widget verrà ridimensionato automaticamente in base alle dimensioni dell'immagine sorgente.

Se la larghezza o l'altezza del Widget sono impostate su un valore diverso, il valore della proprietà `inner_align` (impostato tramite `lv_image_set_inner_align(widget, align)`) determina l'allineamento dell'immagine sorgente all'interno del Widget.

`align` può assumere uno qualsiasi di questi valori:

- `LV_IMAGE_ALIGN_DEFAULT`: Significa in alto a sinistra
- `LV_IMAGE_ALIGN_TOP_LEFT`
- `LV_IMAGE_ALIGN_TOP_MID`
- `LV_IMAGE_ALIGN_TOP_RIGHT`
- `LV_IMAGE_ALIGN_BOTTOM_LEFT`
- `LV_IMAGE_ALIGN_BOTTOM_MID`
- `LV_IMAGE_ALIGN_BOTTOM_RIGHT`
- `LV_IMAGE_ALIGN_LEFT_MID`
- `LV_IMAGE_ALIGN_RIGHT_MID`
- `LV_IMAGE_ALIGN_CENTER`
- `LV_IMAGE_ALIGN_STRETCH`

- *LV_IMAGE_ALIGN_TILE*
- *LV_IMAGE_ALIGN_CONTAIN*
- *LV_IMAGE_ALIGN_COVER*

Qualsiasi valore di *offset* viene applicato dopo l'allineamento della sorgente dell'immagine. Ad esempio, impostando un offset di *y*=-10 con *align* == *LV_IMAGE_ALIGN_CENTER*, la sorgente dell'immagine verrà spostata di 10 pixel verso l'alto rispetto al centro del Widget.

Per ridimensionare o affiancare automaticamente l'immagine, passare uno di questi valori di *align*:

- *LV_IMAGE_ALIGN_STRETCH* Imposta la scala X e Y per riempire l'area del Widget.
- *LV_IMAGE_ALIGN_TILE* Affianca l'immagine per riempire l'area del Widget. L'offset viene applicato per spostare l'affiancamento.
- *LV_IMAGE_ALIGN_CONTAIN* L'immagine mantiene le sue proporzioni, ma viene ridimensionata alla dimensione massima che rientra nell'area del Widget.
- *LV_IMAGE_ALIGN_COVER* L'immagine mantiene le sue proporzioni e riempie l'area del Widget.

Data binding

Per familiarizzare con "observer", "subject" e data binding in generale, visitare la pagina *Observer*.

Questo metodo di sottoscrizione a un Subject puntatore influisce direttamente sul valore sorgente (*src*) di un Widget Image. Si noti che si tratta di un binding unidirezionale (Subject ==> Widget), quindi quando il subject cambia, anche l'Image verrà aggiornata.

Supporta solo subject puntatore.

- *lv_image_bind_src*(img, &subject)

6.14.4 Eventi

I Widget Image non inviano eventi speciali. Per default, i Widget Image vengono creati senza il flag *LV_OBJ_FLAG_CLICKABLE*, ma è possibile aggiungerlo per far sì che un Widget Image rilevi ed emetta eventi *LV_EVENT_CLICKED*.

Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.14.5 Tasti

Nessun *Tasto* viene elaborato dai Widget Image.

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.14.6 Esempio

Immagine da variabile e simbolo

```
#include "../../lv_examples.h"
#if LV_USE_IMAGE && LV_BUILD_EXAMPLES
void lv_example_image_1(void)
{
    LV_IMAGE_DECLARE(img_cogwheel_argb);
    lv_obj_t * img1 = lv_image_create(lv_screen_active());
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_image_set_src(img1, &img_cogwheel_argb;
lv_obj_align(img1, LV_ALIGN_CENTER, 0, 0);

lv_obj_t * img2 = lv_image_create(lv_screen_active());
lv_image_set_src(img2, LV_SYMBOL_OK "Accept");
lv_obj_align_to(img2, img1, LV_ALIGN_OUT_BOTTOM_MID, 0, 20);
}

#endif

```

Ricolorazione dell'immagine

```

#include "../../lv_examples.h"
#if LV_USE_IMAGE && LV_USE_SLIDER && LV_BUILD_EXAMPLES

static lv_obj_t * create_slider(lv_color_t color);
static void slider_event_cb(lv_event_t * e);

static lv_obj_t * red_slider, * green_slider, * blue_slider, * intense_slider;
static lv_obj_t * img1;

/***
 * Demonstrate runtime image re-coloring
 */
void lv_example_image_2(void)
{
    /*Create 4 sliders to adjust RGB color and re-color intensity*/
    red_slider = create_slider(lv_palette_main(LV_PALETTE_RED));
    green_slider = create_slider(lv_palette_main(LV_PALETTE_GREEN));
    blue_slider = create_slider(lv_palette_main(LV_PALETTE_BLUE));
    intense_slider = create_slider(lv_palette_main(LV_PALETTE_GREY));

    lv_slider_set_value(red_slider, LV_OPA_20, LV_ANIM_OFF);
    lv_slider_set_value(green_slider, LV_OPA_90, LV_ANIM_OFF);
    lv_slider_set_value(blue_slider, LV_OPA_60, LV_ANIM_OFF);
    lv_slider_set_value(intense_slider, LV_OPA_50, LV_ANIM_OFF);

    lv_obj_align(red_slider, LV_ALIGN_LEFT_MID, 25, 0);
    lv_obj_align_to(green_slider, red_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);
    lv_obj_align_to(blue_slider, green_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);
    lv_obj_align_to(intense_slider, blue_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);

    /*Now create the actual image*/
    LV_IMAGE_DECLARE(img_cogwheel_argb);
    img1 = lv_image_create(lv_screen_active());
    lv_image_set_src(img1, &img_cogwheel_argb);
    lv_obj_align(img1, LV_ALIGN_RIGHT_MID, -20, 0);

    lv_obj_send_event(intense_slider, LV_EVENT_VALUE_CHANGED, NULL);
}

static void slider_event_cb(lv_event_t * e)
{
    LV_UNUSED(e);

    /*Recolor the image based on the sliders' values*/
    lv_color_t color = lv_color_make((uint8_t)lv_slider_get_value(red_slider),
                                    (uint8_t)lv_slider_get_value(green_slider),
                                    (uint8_t)lv_slider_get_value(blue_slider));
    lv_opa_t intense = (lv_opa_t)lv_slider_get_value(intense_slider);
    lv_obj_set_style_image_recolor_opa(img1, intense, 0);
    lv_obj_set_style_image_recolor(img1, color, 0);
}

static lv_obj_t * create_slider(lv_color_t color)
{
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_slider_set_range(slider, 0, 255);
    lv_obj_set_size(slider, 10, 200);
    lv_obj_set_style_bg_color(slider, color, LV_PART_KNOB);
    lv_obj_set_style_bg_color(slider, lv_color_darken(color, LV_OPA_40), LV_PART_INDICATOR);
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    return slider;
}
#endif

```

Rotazione e zoom

```

#include "../../lv_examples.h"
#if LV_USE_IMAGE && LV_BUILD_EXAMPLES

static void set_angle(void * img, int32_t v)
{
    lv_image_set_rotation((lv_obj_t *)img, v);
}

static void set_scale(void * img, int32_t v)
{
    lv_image_set_scale((lv_obj_t *)img, v);
}

/***
 * Show transformations (zoom and rotation) using a pivot point.
 */
void lv_example_image_3(void)
{
    LV_IMAGE_DECLARE(img_cogwheel_argb);

    /*Now create the actual image*/
    lv_obj_t * img = lv_image_create(lv_screen_active());
    lv_image_set_src(img, &img_cogwheel_argb);
    lv_obj_align(img, LV_ALIGN_CENTER, 50, 50);
    lv_image_set_pivot(img, 0, 0); /*Rotate around the top left corner*/
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_anim_t a;
lv_anim_init(&a);
lv_anim_set_var(&a, img);
lv_anim_set_exec_cb(&a, set_angle);
lv_anim_set_values(&a, 0, 3600);
lv_anim_set_duration(&a, 5000);
lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
lv_anim_start(&a);

lv_anim_set_exec_cb(&a, set_scale);
lv_anim_set_values(&a, 128, 256);
lv_anim_set_reverse_duration(&a, 3000);
lv_anim_start(&a);
}

#endif

```

Offset e stile dell'immagine

```

#include "../../lv_examples.h"
#if LV_USE_IMAGE && LV_BUILD_EXAMPLES

static void ofs_y_anim(void * img, int32_t v)
{
    lv_image_set_offset_y((lv_obj_t *)img, v);
}

/**
 * Image styling and offset
 */
void lv_example_image_4(void)
{
    LV_IMAGE_DECLARE(img_skew_strip);

    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_image_recolor_opa(&style, LV_OPA_COVER);
    lv_style_set_image_recolor(&style, lv_color_black());

    lv_obj_t * img = lv_image_create(lv_screen_active());
    lv_obj_add_style(img, &style, 0);
    lv_image_set_src(img, img_skew_strip);
    lv_obj_set_size(img, 150, 100);
    lv_obj_center(img);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, img);
    lv_anim_set_exec_cb(&a, ofs_y_anim);
    lv_anim_set_values(&a, 0, 100);
    lv_anim_set_duration(&a, 3000);
    lv_anim_set_reverse_duration(&a, 500);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);
}

#endif

```

6.14.7 API

lv_draw_buf.h

lv_draw_image.h

lv_draw_image_private.h

lv_image_decoder.h

lv_image_decoder_private.h

lv_image_dsc.h

lv_api_map_v8.h

lv_api_map_v9_0.h

lv_api_map_v9_1.h

lv_image_cache.h

lv_image_header_cache.h

lv_style.h

lv_image.h

lv_image_private.h

*lv_imagebutton.h**lv_imagebutton_private.h**lv_obj_property_names.h*

6.15 Image Button (`lv_imagebutton`)

6.15.1 Panoramica

"Image Button" [pulsante immagine] è molto simile al semplice Widget 'Button'. L'unica differenza è che visualizza immagini definite dall'utente per ogni stato invece di disegnare un rettangolo. L'elenco degli stati è illustrato di seguito.

È possibile impostare un'immagine a sinistra, a destra e al centro; l'immagine al centro verrà ripetuta per adattarsi alla larghezza del widget.

6.15.2 Parti e Stili

- *LV_PART_MAIN* Si riferisce alle immagini. Se vengono utilizzate le proprietà di stile del background, verrà disegnato un rettangolo dietro l'Image Button.

6.15.3 Utilizzo

Sorgenti delle immagini

Per impostare l'immagine in uno stato, si usa `lv_imagebutton_set_src(imagebutton, LV_IMAGEBUTTON_STATE_... , src_left, src_mid, src_right)`.

Le sorgenti delle immagini funzionano come descritto nel [Widget Image](#) con la differenza che i "Simboli" non sono supportati dall'Image Button. Qualsiasi sorgente può essere NULL. In genere, l'immagine centrale dovrebbe essere una delle sorgenti di immagini impostate.

Se viene specificato solo `src_mid`, la larghezza del widget verrà impostata automaticamente sulla larghezza dell'immagine. Tuttavia, se vengono impostate tutte e tre le sorgenti, la larghezza deve essere impostata dall'utente (ad esempio, utilizzando `lv_obj_set_width`) e l'immagine centrale verrà suddivisa in riquadri per riempire la dimensione specificata.

I possibili stati sono:

- *LV_IMAGEBUTTON_STATE_RELEASED*
- *LV_IMAGEBUTTON_STATE_PRESSED*
- *LV_IMAGEBUTTON_STATE_DISABLED*
- *LV_IMAGEBUTTON_STATE_CHECKED_RELEASED*
- *LV_IMAGEBUTTON_STATE_CHECKED_PRESSED*
- *LV_IMAGEBUTTON_STATE_CHECKED_DISABLED*

Le sorgenti di immagini impostate per lo stato *LV_IMAGEBUTTON_STATE_RELEASED* vengono utilizzate per qualsiasi stato per il quale non siano state impostate sorgenti immagine. Se sono state impostate sorgenti immagine per altri stati, ad esempio *LV_IMAGEBUTTON_STATE_PRESSED*, queste verranno utilizzate al loro posto quando l'Image Button si trova in quello stato.

Impostazione dello Stato Tramite Programmazione

Invece delle normali funzioni `lv_obj_add_state()` e `lv_obj_remove_state()`, si usa `lv_imagebutton_set_state(imagebutton, LV_IMAGEBUTTON_STATE_...)` per impostare lo stato degli Image Button.

6.15.4 Eventi

- ***LV_EVENT_VALUE_CHANGED*** Inviato quando lo stato CHECKED dell'Image Button viene attivato/disattivato. Richiede l'impostazione del flag ***LV_OBJ_FLAG_CHECKABLE*** dell'Image Button.

Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.15.5 Tasti

- **LV_KEY_RIGHT/UP** Passa allo stato CHECKED se ***LV_OBJ_FLAG_CHECKABLE*** è abilitato.
- **LV_KEY_LEFT/DOWN** Passa allo stato non CHECKED se ***LV_OBJ_FLAG_CHECKABLE*** è abilitato.
- ***LV_KEY_ENTER*** Clicca sull'Image Button

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.15.6 Esempio

Pulsante immagine semplice

```
#include "../../lv_examples.h"
#ifndef LV_USE_IMAGEBUTTON && LV_BUILD_EXAMPLES
void lv_example_imagebutton_1(void)
{
    LV_IMAGE_DECLARE(imagebutton_left);
    LV_IMAGE_DECLARE(imagebutton_right);
    LV_IMAGE_DECLARE(imagebutton_mid);

    /*Create a transition animation on width transformation and recolor.*/
    static lv_style_prop_t tr_prop[] = {LV_STYLE_TRANSFORM_WIDTH, LV_STYLE_IMAGE_RECOLOR_OPA, 0};
    static lv_style_transition_dsc_t tr;
    lv_style_transition_dsc_init(&tr, tr_prop, lv_anim_path_linear, 200, 0, NULL);

    static lv_style_t style_def;
    lv_style_init(&style_def);
    lv_style_set_text_color(&style_def, lv_color_white());
    lv_style_set_transition(&style_def, &r);

    /*Darken the button when pressed and make it wider*/
    static lv_style_t style_pr;
    lv_style_init(&style_pr);
    lv_style_set_image_recolor_opa(&style_pr, LV_OPA_30);
    lv_style_set_image_recolor(&style_pr, lv_color_black());
    lv_style_set_transform_width(&style_pr, 20);

    /*Create an image button*/
    lv_obj_t * imagebutton1 = lv_imagebutton_create(lv_screen_active());
    lv_imagebutton_set_src(imagebutton1, LV_IMAGEBUTTON_STATE_RELEASED, &imagebutton_left, &imagebutton_mid,
                          &imagebutton_right);
    lv_obj_add_style(imagebutton1, &style_def, 0);
    lv_obj_add_style(imagebutton1, &style_pr, LV_STATE_PRESSED);

    lv_obj_set_width(imagebutton1, 100);
    lv_obj_align(imagebutton1, LV_ALIGN_CENTER, 0, 0);

    /*Create a label on the image button*/
    lv_obj_t * label = lv_label_create(imagebutton1);
    lv_label_set_text(label, "Button");
    lv_obj_align(label, LV_ALIGN_CENTER, 0, -4);
}
#endif
```

6.15.7 API

lv_imagebutton.h

lv_imagebutton_private.h

6.16 Pinyin IME

Pinyin IME fornisce un'API per fornire il metodo di input Pinyin cinese (input cinese) per un Widget Keyboard, che supporta sia la modalità di input a 26 tasti che a 9 tasti. Si può pensare a `lv_ime_pinyin` come a un plug-in del metodo di input Pinyin per il Widget Keyboard.

Normalmente, un ambiente in cui può essere eseguito `Keyboard (lv_keyboard)`, può eseguire anche `lv_ime_pinyin`. Ci sono due fattori principali che influenzano l'esecuzione: la dimensione del file del font e la dimensione del dizionario.

6.16.1 Utilizzo

Abilitare `LV_USEIME_PINYIN` in `lv_conf.h`.

Per prima cosa, si usa `lv_ime_pinyin_create(lv_screen_active())` per creare un plug-in per il metodo di input Pinyin, poi si usa `lv_ime_pinyin_set_keyboard(pinyin_ime, kb)` per aggiungere il Widget Keyboard creato al plug-in per il metodo di input Pinyin. Si può usare `lv_ime_pinyin_set_dict(pinyin_ime, your_dict)` per usare un dizionario personalizzato. Se non si vuole usare il dizionario nativo, si può disabilitare `LV_IMEPINYIN_USE_DEFAULT_DICT` in `lv_conf.h`, il che può far risparmiare molto spazio di memoria.

Il thesaurus integrato è personalizzato in base alla libreria di font **LV_FONT_SOURCE_HAN_SANS_SC_16_CJK**, che attualmente include più di 1.000 dei radicali CJK più comuni quindi si consiglia di utilizzare un font e un thesaurus personalizzati.

Durante l'utilizzo del plug-in per il metodo di input Pinyin, è possibile modificare la tastiera e il dizionario in qualsiasi momento.

6.16.2 Dizionario Personalizzato

Se non si desidera utilizzare il dizionario Pinyin integrato o se si ritiene che il dizionario fonetico integrato consumi troppa memoria, è possibile utilizzare un dizionario personalizzato.

Personalizzare il dizionario è molto semplice.

Per prima cosa, impostare `LV_IMEPINYIN_USE_DEFAULT_DICT` a 0 in `lv_conf.h`

Quindi, scrivere un dizionario nel seguente formato.

Formato del dizionario

L'ordine di disposizione di ogni sillaba pinyin è molto importante. Se c'è bisogno di personalizzare il dizionario dei sinonimi in base alla tabella sillabica Hanyu Pinyin, si può leggere [qui](#) per saperne di più sulle sillabe Hanyu Pinyin e sulla tabella sillabica.

Poi, si scrive il proprio dizionario secondo il seguente formato:

```
lv_100ask_pinyin_dict_t your_pinyin_dict[] = {
    { "a", "ㄞㄞㄞ" },
    { "ai", "ㄞㄞㄞㄞ" },
    { "an", "ㄞㄞㄞㄞㄞ" },
    { "ang", "ㄞㄞㄞㄞㄞ" },
    { "ao", "ㄞㄞㄞㄞ" },
    { "ba", "ㄞㄞㄞㄞㄞ" },
    { "bai", "ㄞㄞㄞㄞㄞ" },
    /* ..... */
    { "zuo", "ㄞㄞㄞㄞㄞ" },
    {NULL, NULL}
```

L'ultimo elemento deve essere `{null, null}`, altrimenti non funzionerà correttamente.

Applicazione di un nuovo dizionario

Dopo aver scritto un dizionario secondo il formato sopra indicato, è sufficiente chiamare questa funzione per impostarlo e utilizzarlo:

```
lv_obj_t *pinyin_ime = lv_100ask_pinyin_ime_create(lv_screen_active());
lv_100ask_pinyin_ime_set_dict(pinyin_ime, your_pinyin_dict);
```

6.16.3 Modalità

lv_ime_pinyin ha le seguenti modalità:

- *LVIMEPINYINMODEK26*: Modalità di input Pinyin a 26 tasti
 - *LVIMEPINYINMODEK9*: Modalità di input Pinyin a 9 tasti
 - *LVIMEPINYINMODEK9NUMBER*: Modalità tastierino numerico

Il layout della modalità TEXT della tastiera contiene pulsanti per cambiare modalità.

Per impostare manualmente la modalità, si usa `lv_ime_pinyin_set_mode(pinyin_ime, mode)`. La modalità di default è `LVIME_PINYIN_MODE_K26`.

6.16.4 Esempio

Pinyin IME input 26 tasti

Pinyin IME input 9 tasti

```
#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_USE_TEXTAREA && LV_FONT_SOURCE_HAN_SANS_SC_16_CJK && LV_USE_IME_PINYIN && LV_IME_PINYIN_USE_K9_MODE && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * ta = lv_event_get_target_obj(e);
    lv_obj_t * kb = (lv_obj_t *) lv_event_get_user_data(e);

    if(code == LV_EVENT_FOCUSED) {
        if(lv_indev_get_type(lv_indev_active()) != LV_INDEV_TYPE_KEYPAD) {
            lv_keyboard_set_textarea(kb, ta);
            lv_obj_remove_flag(kb, LV_OBJ_FLAG_HIDDEN);
        }
    }
    else if(code == LV_EVENT_READY) {
        lv_obj_add_flag(kb, LV_OBJ_FLAG_HIDDEN);
        lv_obj_remove_state(ta, LV_STATE_FOCUSED);
        lv_indev_reset(NULL, ta); /*To forget the last clicked object to make it focusable again*/
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

}

void lv_example_ime_pinyin_2(void)
{
    lv_obj_t * pinyin_ime = lv_ime_pinyin_create(lv_screen_active());
    lv_obj_set_style_text_font(pinyin_ime, &lv_font_source_han_sans_sc_16_cjk, 0);
    //lv_ime_pinyin_set_dict(pinyin_ime, your_dict); // Use a custom dictionary. If it is not set, the built-in dictionary will be used.

    /* tal */
    lv_obj_t * tal = lv_textarea_create(lv_screen_active());
    lv_textarea_set_one_line(tal, true);
    lv_obj_set_style_text_font(tal, &lv_font_source_han_sans_sc_16_cjk, 0);
    lv_obj_align(tal, LV_ALIGN_TOP_LEFT, 0, 0);

    /*Create a keyboard and add it to ime_pinyin*/
    lv_obj_t * kb = lv_keyboard_create(lv_screen_active());
    lv_keyboard_set_textarea(kb, tal);

    lv_ime_pinyin_set_keyboard(pinyin_ime, kb);
    lv_ime_pinyin_set_mode(pinyin_ime,
        LV_IME_PINYIN_MODE_K9); // Set to 9-key input mode. Default: 26-key input(k26) mode.
    lv_obj_add_event_cb(tal, ta_event_cb, LV_EVENT_ALL, kb);

    /*Get the cand_panel, and adjust its size and position*/
    lv_obj_t * cand_panel = lv_ime_pinyin_get_cand_panel(pinyin_ime);
    lv_obj_set_size(cand_panel, LV_PCT(100), LV_PCT(10));
    lv_obj_align_to(cand_panel, kb, LV_ALIGN_OUT_TOP_MID, 0, 0);

    /*Try using ime_pinyin to output the Chinese below in the tal above*/
    lv_obj_t * cz_label = lv_label_create(lv_screen_active());
    lv_label_set_text(cz_label,
        "?????Embedded System????????????????????????????????");
    lv_obj_set_style_text_font(cz_label, &lv_font_source_han_sans_sc_16_cjk, 0);
    lv_obj_set_width(cz_label, 310);
    lv_obj_align_to(cz_label, tal, LV_ALIGN_OUT_BOTTOM_LEFT, 0, 0);
}

#endif

```

6.16.5 API

lv_ime_pinyin.h

lv_ime_pinyin_private.h

6.17 Keyboard (lv_keyboard)

6.17.1 Panoramica

Il Widget "Keyboard" [tastiera] è una speciale *Matrice di Pulsanti* (*lv_buttonmatrix*) con mappe dei tasti predefinite e altre funzionalità per fornire una tastiera virtuale su schermo per scrivere testo in una *Text Area* (*lv_textarea*).

6.17.2 Parti e Stili

Simile alla Button Matrix, il Widget Keyboard è composto da 2 parti:

- *LV_PART_MAIN* La parte principale. Utilizza le *tipiche proprietà di stile del background*
- *LV_PART_ITEMS* I pulsanti. Utilizza anche le *tipiche proprietà di stile del background* e le proprietà *text*.

6.17.3 Utilizzo

Modalità

Le tastiere hanno le seguenti modalità:

- *LV_KEYBOARD_MODE_TEXT_LOWER* Visualizza lettere minuscole
- *LV_KEYBOARD_MODE_TEXT_UPPER* Visualizza lettere maiuscole
- *LV_KEYBOARD_MODE_SPECIAL* Visualizza caratteri speciali
- *LV_KEYBOARD_MODE_NUMBER* Visualizza numeri, segno +/- e punto decimale
- Da *LV_KEYBOARD_MODE_USER_1* a *LV_KEYBOARD_MODE_USER_4* Modalità definite dall'utente.

I layout delle modalità TEXT contengono "tasti" per cambiare modalità.

Per impostare la modalità da codice, si usa *lv_keyboard_set_mode(kb, mode)*. La modalità di default è *LV_KEYBOARD_MODE_TEXT_UPPER*.

Assegnazione dell'Area di Testo

È possibile assegnare un'*Area di testo* alla tastiera per posizionarvi automaticamente i caratteri cliccati. Per assegnarla, si usa `lv_keyboard_set_textarea(kb, text_area)`.

Pop-over dei Tasti

Per abilitare i pop-over dei tasti alla pressione, come sulle comuni tastiere Android e iOS, si usa `lv_keyboard_set_popovers(kb, true)`. Le mappe di controllo di default sono preconfigurate per mostrare i pop-over solo sui tasti che producono un simbolo (ovvero non sulla barra spaziatrice). Se si utilizza una mappa di controllo personalizzata (vedere di seguito), impostare il flag `LV_BUTTONMATRIX_CTRL_POPOVER` per ogni tasto per cui si desidera visualizzare un pop-over.

Si noti che i pop-over per i tasti nella riga superiore verranno visualizzati al di fuori dei limiti del widget. Per risolvere questo problema, riservare spazio libero extra sopra la Keyboard o assicurarsi che questa venga aggiunta *dopo* eventuali Widget adiacenti al suo limite superiore (posizionandola "sopra" tali Widget) in modo che i pop-over vengano visualizzati sopra di essi.

Al momento, i pop-over sono solo un effetto visivo e non consentono ancora la selezione di caratteri aggiuntivi, come quelli accentati.

Nuova Mappa dei Pulsanti

È possibile specificare una nuova mappa (layout) per la tastiera con `lv_keyboard_set_map(kb, LV_KEYBOARD_MODE_..., kb_map, kb_ctrl)`. Consultare la sezione *Mappa dei pulsanti* del Button Matrix per ulteriori informazioni sulla creazione di nuove mappe.

Tenere presente che l'utilizzo delle seguenti parole chiave nella mappa avrà lo stesso effetto della mappa originale:

- `LV_SYMBOL_OK` Invia `LV_EVENT_READY` alla Text Area assegnata.
- `LV_SYMBOL_CLOSE` o `LV_SYMBOL_KEYBOARD` Inviano `LV_EVENT_CANCEL` all'area di testo assegnata.
- `LV_SYMBOL_BACKSPACE` Cancella il carattere a sinistra.
- `LV_SYMBOL_LEFT` Sposta il cursore a sinistra.
- `LV_SYMBOL_RIGHT` Sposta il cursore a destra.
- `LV_SYMBOL_NEW_LINE` Nuova riga.
- "ABC" Carica la mappa delle maiuscole.
- "abc" Carica la mappa delle minuscole.
- "1#" Carica la mappa dei numeri.

Personalizzazione dei tasti

Ogni tasto può essere personalizzato utilizzando eventi di disegno come `LV_EVENT_DRAW_TASK_ADDED`.

Controllare gli esempi per vedere come può essere utilizzato nella pratica.

6.17.4 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato quando il pulsante viene premuto/rilasciato o ripetuto dopo una pressione prolungata. I dati dell'evento contengono l'ID del pulsante premuto/rilasciato.
- `LV_EVENT_READY`: È stato cliccato il pulsante *Ok*.
- `LV_EVENT_CANCEL`: È stato cliccato il pulsante *Close*.

La tastiera dispone di un **gestore eventi predefinito** chiamato `lv_keyboard_def_event_cb()`, che gestisce la pressione del pulsante, il cambio di mappa, l'invio di eventi all'area di testo assegnata, ecc. È possibile rimuoverlo e sostituirlo con un gestore eventi personalizzato, oppure aggiungerne uno personalizzato.

i Nota

In LVGL v8.0 e versioni successive, l'aggiunta di un gestore eventi alla Keyboard non rimuove il gestore eventi di default. Questo comportamento è diverso dalla versione 7, dove l'aggiunta di un gestore di eventi sostituirebbe quello precedente.

i Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.17.5 Tasti

- *LV_KEY_RIGHT/UP/LEFT/RIGHT* Per navigare tra i pulsanti, selezionando quello desiderato.
- *LV_KEY_ENTER* Per premere/rilasciare il pulsante selezionato.

i Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.17.6 Esempio

Tastiera con area di testo

```
#include "../../lv_examples.h"
#if LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * ta = lv_event_get_target_obj(e);
    lv_obj_t * kb = (lv_obj_t *)lv_event_get_user_data(e);
    if(code == LV_EVENT_FOCUSED) {
        lv_keyboard_set_textarea(kb, ta);
        lv_obj_add_flag(kb, LV_OBJ_FLAG_HIDDEN);
    }

    if(code == LV_EVENT_DEFOCUSSED) {
        lv_keyboard_set_textarea(kb, NULL);
        lv_obj_add_flag(kb, LV_OBJ_FLAG_HIDDEN);
    }
}

void lv_example_keyboard_1(void)
{
    /*Create a keyboard to use it with an of the text areas*/
    lv_obj_t * kb = lv_keyboard_create(lv_screen_active());

    /*Create a text area. The keyboard will write here*/
    lv_obj_t * tal;
    tal = lv_textarea_create(lv_screen_active());
    lv_obj_align(tal, LV_ALIGN_TOP_LEFT, 10, 10);
    lv_obj_add_event_cb(tal, ta_event_cb, LV_EVENT_ALL, kb);
    lv_textarea_set_placeholder_text(tal, "Hello");
    lv_obj_set_size(tal, 140, 80);

    lv_obj_t * ta2;
    ta2 = lv_textarea_create(lv_screen_active());
    lv_obj_align(ta2, LV_ALIGN_TOP_RIGHT, -10, 10);
    lv_obj_add_event_cb(ta2, ta_event_cb, LV_EVENT_ALL, kb);
    lv_obj_set_size(ta2, 140, 80);

    lv_keyboard_set_textarea(kb, tal);

    /*The keyboard will show Arabic characters if they are enabled */
#if LV_USE_ARABIC_PERSIAN_CHARS && LV_FONT_DEJAVU_16_PERSIAN_HEBREW
    lv_obj_set_style_text_font(kb, &lv_font_dejavu_16_persian_hebrew, 0);
    lv_obj_set_style_text_font(tal, &lv_font_dejavu_16_persian_hebrew, 0);
    lv_obj_set_style_text_font(ta2, &lv_font_dejavu_16_persian_hebrew, 0);
#endif
}
#endif
```

Tastiera con mappa personalizzata

```

#include "../../lv_examples.h"
#ifndef LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

void lv_example_keyboard_2(void)
{
    /*Create an AZERTY keyboard map*/
    static const char * kb_map[] = {"A", "Z", "E", "R", "T", "Y", "U", "I", "O", "P", LV_SYMBOL_BACKSPACE, "\n",
                                    "Q", "S", "D", "F", "G", "J", "K", "L", "M", LV_SYMBOL_NEW_LINE, "\n",
                                    "W", "X", "C", "V", "B", "N", "A", "S", "D", "F", "G", "J", "K", "L", "M", LV_SYMBOL_CLOSE, "", "", "", LV_SYMBOL_OK, NULL
    };

    /*Set the relative width of the buttons and other controls*/
    static const lv_buttonmatrix_ctrl_t kb_ctrl[] = {LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_HIDDEN | LV_BUTTONMATRIX_CTRL_WIDTH_2, LV_BUTTONMATRIX_CTRL_WIDTH_2,
    LV_BUTTONMATRIX_CTRL_WIDTH_6, (lv_buttonmatrix_ctrl_t)(LV_BUTTONMATRIX_CTRL_HIDDEN | LV_BUTTONMATRIX_CTRL_WIDTH_2), LV_BUTTONMATRIX_CTRL_WIDTH_2
    };

    /*Create a keyboard and add the new map as USER_1 mode*/
    lv_obj_t * kb = lv_keyboard_create(lv_screen_active());

    lv_keyboard_set_map(kb, LV_KEYBOARD_MODE_USER_1, kb_map, kb_ctrl);
    lv_keyboard_set_mode(kb, LV_KEYBOARD_MODE_USER_1);

    /*Create a text area. The keyboard will write here*/
    lv_obj_t * ta;
    ta = lv_textarea_create(lv_screen_active());
    lv_obj_align(ta, LV_ALIGN_TOP_MID, 0, 10);
    lv_obj_set_size(ta, lv_pct(90), 80);
    lv_obj_add_state(ta, LV_STATE_FOCUSED);

    lv_keyboard_set_textarea(kb, ta);
}

#endif

```

Tastiera con disegno

```

#include "../../lv_examples.h"
#if LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    bool pressed = false;
    if(lv_keyboard_get_selected_button(obj) == base_dsc->idl && lv_obj_has_state(obj, LV_STATE_PRESSED)) {
        pressed = true;
    }

/*When the keyboard draws the buttons...*/
if(base_dsc->part == LV_PART_ITEMS) {
    /*Get a color based on the button's index*/
    lv_palette_t palette = (lv_palette_t)(base_dsc->idl % LV_PALETTE_LAST);
    lv_draw_fill_dsc_t * fill_draw_dsc = lv_draw_task_get_fill_dsc(draw_task);
    if(fill_draw_dsc) {
        fill_draw_dsc->color = pressed ? lv_palette_darken(palette, 3) : lv_palette_main(palette);
    }
    lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
    if(label_draw_dsc) {
        /*For the OK symbol, draw a star instead*/
        if(lv_strcmp(label_draw_dsc->text, LV_SYMBOL_OK) == 0) {
            label_draw_dsc->opa = 0; /*Hide the label*/
            lv_image_DECLARE(img_star);
            lv_image_header_t header;
            lv_result_t res = lv_image_decoder_get_info(&img_star, &header);
            if(res != LV_RESULT_OK) return;

            lv_area_t a;
            lv_area_set(&a, 0, 0, header.w - 1, header.h - 1);
            lv_area_t draw_task_area;
            lv_draw_task_get_area(draw_task, &draw_task_area);
            lv_area_align(&draw_task_area, &a, LV_ALIGN_CENTER, 0, 0);

            lv_draw_image_dsc_t img_draw_dsc;
            lv_draw_image_dsc_init(&img_draw_dsc);
            img_draw_dsc.src = &img_star;
            lv_draw_image(base_dsc->layer, &img_draw_dsc, &a);
        }
    else {
        /*For the other labels just pick an lighter color*/
        label_draw_dsc->color = lv_palette_lighten(palette, 4);
    }
}
}

/** 
 * Add custom drawer to the keyboard to customize the buttons one by one
 */
void lv_example_keyboard_3(void)
{
    lv_obj_t * kb = lv_keyboard_create(lv_screen_active());
    lv_obj_add_event_cb(kb, event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(kb, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
    lv_obj_center(kb);
}

#endif

```

6.17.7 API

lv_api_map_v8.h

lv_keyboard.h

lv_keyboard_private.h

lv_obj_property_names.h

6.18 Label (lv_label)

6.18.1 Panoramica

Una "Label" [etichetta] è il Widget utilizzato per visualizzare il testo.

6.18.2 Parti e Stili

- *LV_PART_MAIN* Utilizza le proprietà *tipiche del background* e *text*. I valori di padding possono essere utilizzati per aggiungere spazio tra il testo e i bordi del background dell'etichetta.
- *LV_PART_SCROLLBAR* La barra di scorrimento che viene visualizzata quando il testo è più grande delle dimensioni del Widget.
- *LV_PART_SELECTED* Indica lo stile del *testo selezionato*. È possibile utilizzare solo le proprietà di stile *text_color* e *bg_color*.

6.18.3 Utilizzo

Impostazione del testo

È possibile impostare il testo su un'etichetta in fase di esecuzione con *lv_label_set_text*(label, "New text"). Questo allocherà un buffer dinamicamente e la stringa fornita verrà copiata in quel buffer. Pertanto, non è necessario mantenere il testo passato a *lv_label_set_text()* nello scope dopo il ritorno della funzione.

Con *lv_label_set_text_fmt*(label, fmt, ...) è possibile utilizzare la formattazione printf per impostare il testo. Esempio: *lv_label_set_text_fmt*(label, "Value: %d", 15).

Le etichette possono visualizzare anche il testo da un buffer di caratteri statico. Per farlo, si usa *lv_label_set_text_static*(label, "Text"). In questo caso, il testo non viene posto nella memoria dinamica e viene utilizzato direttamente il buffer specificato. Ciò significa che il contenuto del buffer dei caratteri *dove* rimanere valido per tutta la durata dell'etichetta o finché non viene impostato un altro buffer tramite una delle funzioni sopra indicate.

Le stringhe const sono sicure da usare con *lv_label_set_text_static()* poiché sono memorizzate nella memoria ROM, che è sempre accessibile.

⚠️ Avvertimento

Non utilizzare stringhe const con *lv_label_set_text_static()* quando l'etichetta viene utilizzata in modalità *LV_LABEL_LONG_DOT* poiché l'etichetta tenterà di modificare la stringa sul posto. Ciò causerà un'eccezione MCU tentando di modificare la memoria di programma (ROM).

⚠️ Attenzione

Se l'etichetta viene aggiornata rapidamente con nuove stringhe (ad esempio > 30X/secondo, come RPM in una dashboard o un valore ADC) e la lunghezza di tali stringhe cambia frequentemente, è consigliabile:

- allocare un buffer di stringhe statico sufficientemente grande da contenere la stringa più grande possibile,
- aggiornare tale buffer con le nuove stringhe solo quando queste apporteranno una differenza visibile per l'utente finale e

- aggiornare l'etichetta con `lv_label_set_text_static(label, buffer)` utilizzando tale buffer.

Motivo: se si utilizza `lv_label_set_text(label, new_text)`, verrà forzata una realloc() di memoria ogni volta che la lunghezza della stringa cambia. Questo sovraccarico della MCU può essere evitato eseguendo quanto sopra.

Impostare il tag di traduzione

Quando si usa il modulo di traduzione di LVGL, è possibile associare un tag di traduzione a un'etichetta direttamente con `lv_label_set_translation_tag(label, tag)`. Dopo aver richiamato questa funzione, le future modifiche alla lingua aggiorneranno automaticamente il testo dell'etichetta per visualizzare la traduzione corrispondente per quel tag nella nuova lingua.

Newline

I caratteri di "newline" [nuova riga] vengono gestiti automaticamente dal Widget Label. È possibile utilizzare \n per creare un'interruzione di riga. Ad esempio: "line1\nline2\n\nline4"

Modalità lunghe

Per default, la larghezza e l'altezza dell'etichetta sono impostate su `LV_SIZE_CONTENT`. In questo modo, la dimensione dell'etichetta viene automaticamente espansa in base alla dimensione del testo + padding + spessore del bordo. In alternativa, se la larghezza o l'altezza vengono impostate esplicitamente (ad esempio, utilizzando `lv_obj_set_width()` o un layout), le righe più larghe della larghezza dell'etichetta possono essere manipolate in base a diverse policy di "modalità lunga". Analogamente, le policy possono essere applicate se l'altezza del testo è maggiore dell'altezza della Label.

- `LV_LABEL_LONG_MODE_WRAP` Manda a capo le righe troppo lunghe. Se l'altezza è `LV_SIZE_CONTENT`, l'altezza dell'etichetta verrà espansa, altrimenti il testo verrà tagliato. (Il default)
- `LV_LABEL_LONG_MODE_DOTS` Sostituisce gli ultimi 3 caratteri dall'angolo in basso a destra dell'etichetta con punti (..)
- `LV_LABEL_LONG_MODE_SCROLL` Se il testo è più largo dell'etichetta, scorrerlo orizzontalmente avanti e indietro. Se è più alto, scorre verticalmente. Lo scorrimento avviene in una sola direzione e quello orizzontale ha la precedenza.
- `LV_LABEL_LONG_MODE_SCROLL_CIRCULAR` Se il testo è più largo dell'etichetta, scorrerlo orizzontalmente in modo continuo. Se è più alto, scorre verticalmente. Lo scorrimento avviene in una sola direzione e quello orizzontale ha la precedenza.
- `LV_LABEL_LONG_MODE_CLIP` Ritaglia semplicemente le parti di testo esterne all'etichetta.

È possibile specificare la modalità lunga con `lv_label_set_long_mode(label, LV_LABEL_LONG_...)`

Si noti che `LV_LABEL_LONG_DOT` manipola il buffer di testo sul posto per aggiungere/rimuovere i punti. Quando si utilizzano `lv_label_set_text()` o `lv_label_set_array_text()`, viene allocato un buffer separato e questo dettaglio di implementazione passa inosservato. Questo non avviene con `lv_label_set_text_static()`. Il buffer che si passa a `lv_label_set_text_static()` deve essere scrivibile per utilizzare `LV_LABEL_LONG_DOT`.

Ricolorazione del testo

Nel testo, è possibile utilizzare comandi per ricoloreare parti del testo. Ad esempio: Write a #ff0000 red# word. Questa funzionalità può essere abilitata individualmente per ogni etichetta tramite la funzione `lv_label_set_recolor(label, en)`. Nel contesto del testo con ritorno a capo automatico, qualsiasi ricolorazione avviata su una riga verrà terminata alla fine della riga in cui la riga è andata a capo, se non era già terminata da un # finale nel testo.

Selezione del testo

Se abilitata tramite `LV_LABEL_TEXT_SELECTION`, è possibile selezionare parte del testo. È simile a quando si utilizza il mouse su un PC per selezionare il testo. L'intero meccanismo (cliccare e selezionare il testo mentre si trascina il dito/mouse) è implementato in `Text Area (lv_textarea)` e il Widget Label consente solo la

selezione programmatica del testo con `lv_label_get_text_selection_start(label, start_char_index)` e `lv_label_get_text_selection_end(label, end_char_index)`.

Allineamento del testo

Per allineare orizzontalmente le righe di un'etichetta, la proprietà di stile `text_align` può essere utilizzata con `lv_obj_set_style_text_align()` o `lv_style_set_text_align()`, passando uno dei valori di enumerazione `LV_TEXT_ALIGN_...`. Si noti che questo ha un effetto visibile solo se:

- la larghezza del Widget Label è maggiore della larghezza della riga di testo più lunga e
- il testo ha più righe con lunghezze diverse.

Testo molto lungo

LVGL può gestire in modo efficiente etichette molto lunghe (ad esempio > 40k caratteri) risparmiando alcuni dati extra (~12 byte) per velocizzare il disegno. Per abilitare questa funzionalità, impostare `LV_LABEL_LONG_TXT_HINT` a 1 in `lv_conf.h`.

Animazioni di scorrimento personalizzate

Alcuni aspetti delle animazioni di scorrimento nelle modalità lunghe `LV_LABEL_LONG_SCROLL` e `LV_LABEL_LONG_SCROLL_CIRCULAR` possono essere personalizzati impostando la proprietà di stile dell'animazione dell'etichetta, utilizzando `lv_style_set_anim()`. Verrà trattato come un template utilizzato per creare le animazioni dello scorrimento.

Simboli

Le etichette possono visualizzare simboli accanto alle lettere (o da soli). Leggere la sezione [Panoramica](#) per saperne di più sui simboli.

Data binding

Per familiarizzare con "observer", "subject" e data binding in generale, visitare la pagina [Observer](#).

Questo metodo di sottoscrizione a un Subject influisce sul `text` di un Widget Label. Il Subject può essere di tipo STRING, POINTER, INTEGER o FLOAT.

Quando si verifica la sottoscrizione, e ogni volta che il valore del Subject viene modificato in seguito, il valore del Subject viene utilizzato per aggiornare il testo della Label in questo modo:

Subject string

La stringa del Subject viene utilizzata per aggiornare direttamente il testo della Label.

Subject pointer

Se durante la sottoscrizione viene passato NULL come argomento `format_string`, si presume che il valore del puntatore del Subject punti a una stringa terminata da NULL e viene utilizzato per aggiornare direttamente il testo dell'etichetta. Vedere [Argomento format_string](#) per altre opzioni.

Subject integer

Il valore intero di Subject viene utilizzato con l'argomento `format_string`. Vedere [Argomento format_string](#) per i dettagli.

Subject float

Il valore float del Subject viene utilizzato con l'argomento `format_string`. Richiede `LV_USE_FLOAT`. Vedere [Argomento format_string](#) per i dettagli.

Si noti che questo è un binding unidirezionale (Subject ==> Widget).

- `lv_label_bind_text(label, &subject, format_string)`

Argomento `format_string`

L'argomento `format_string` è facoltativo e, se fornito, deve contenere esattamente 1 specificatore di formato simile a `printf` ed essere uno dei seguenti:

Subject string o pointer

"%s" per formattare il nuovo valore del puntatore come stringa o "%p" per formattare il puntatore come puntatore (in genere il valore dell'indirizzo del puntatore è scritto con 4, 8 o 16 caratteri esadecimali a seconda della piattaforma).

Subject integer

Specificatore di formato "%d" ("% PRIIdxx --- un equivalente multipiattaforma dove xx può essere 8, 16, 32 o 64, a seconda della piattaforma).

Subject float

Specificatore di formato "%f", ad esempio "%0.2f", per visualizzare due cifre dopo la virgola.

Se viene passato NULL per l'argomento `format_string`:

Subject string o pointer

Gli aggiornamenti si aspettano che il puntatore punti a una stringa terminata da NUL.

Subject integer

La Label visualizzerà semplicemente il numero. Equivalente a "%d".

Subject float

La Label visualizzerà il valore con la stringa di formato "%0.1f".

Esempio: "%d °C"

Come di consueto con le stringhe di formato, %% viene utilizzato per ottenere %. Per esempio %d%%

6.18.4 Eventi

I Widget Label non inviano eventi speciali. Per default, i Widget Label vengono creati senza il flag `LV_OBJ_FLAG_CLICKABLE`, ma è possibile aggiungerlo per far sì che emetta eventi `LV_EVENT_CLICKED`.

Ulteriori Informazioni

Scoprire di più sugli [Eventi](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

6.18.5 Tasti

Nessun *Tasto* viene elaborato dai Widget Label.

Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

6.18.6 Esempio

A capo automatico, ricolorazione e scorrimento

```
#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES

/***
 * Show line wrap, re-color, line align and text scrolling.
 */
void lv_example_label_1(void)
{
    lv_obj_t * labell = lv_label_create(lv_screen_active());
    lv_label_set_long_mode(labell, LV_LABEL_LONG_MODE_WRAP);           /*Break the long lines*/
    lv_label_set_recolor(labell, true);          /*Enable re-coloring by commands in the text*/
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_label_set_text(label1, "#0000ff Re-color# #ffff00 words# #ff0000 of a# label, align the lines to the center "
    "and wrap long text automatically.");
lv_obj_set_width(label1, 150); /*Set smaller width to make the lines wrap*/
lv_obj_set_style_text_align(label1, LV_TEXT_ALIGN_CENTER, 0);
lv_obj_align(label1, LV_ALIGN_CENTER, 0, -40);

lv_obj_t *label2 = lv_label_create(lv_screen_active());
lv_label_set_long_mode(label2, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR); /*Circular scroll*/
lv_obj_set_width(label2, 150);
lv_label_set_text(label2, "It is a circularly scrolling text. ");
lv_obj_align(label2, LV_ALIGN_CENTER, 0, 40);

#endif

```

Ombreggiatura del testo

```

#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES

/***
 * Create a fake text shadow
 */
void lv_example_label_2(void)
{
    /*Create a style for the shadow*/
    static lv_style_t style_shadow;
    lv_style_init(&style_shadow);
    lv_style_set_text_opa(&style_shadow, LV_OPA_30);
    lv_style_set_text_color(&style_shadow, lv_color_black());

    /*Create a label for the shadow first (it's in the background)*/
    lv_obj_t * shadow_label = lv_label_create(lv_screen_active());
    lv_obj_add_style(shadow_label, &style_shadow, 0);

    /*Create the main label*/
    lv_obj_t * main_label = lv_label_create(lv_screen_active());
    lv_label_set_text(main_label, "A simple method to create\n"
                           "shadows on a text.\n"
                           "It even works with\n"
                           "newlines and spaces.");
    lv_label_set_text(shadow_label, lv_label_get_text(main_label));

    /*Set the same text for the shadow label*/
    lv_label_set_text(shadow_label, lv_label_get_text(main_label));

    /*Position the main label*/
    lv_obj_align(main_label, LV_ALIGN_CENTER, 0, 0);

    /*Shift the second label down and to the right by 2 pixel*/
    lv_obj_align_to(shadow_label, main_label, LV_ALIGN_TOP_LEFT, 2, 2
}

#endif

```

Visualizzazione di testi LTR, RTL e in cinese

Diseño de etiquetas con gradientes de color

```
#include "../../lv_examples.h"

#if LV_USE_LABEL && LV_FONT_MONTSEYAT_24 && LV_USE_CANVAS && LV_BUILD_EXAMPLES && LV_DRAW_SW_COMPLEX

#define MASK_WIDTH 150
#define MASK_HEIGHT 60

static void generate_mask(lv_draw_buf_t * mask, int32_t w, int32_t h, const char * txt)
{
    /*Create a "8-bit alpha" canvas and clear it*/
    lv_color_t color = {0, 0, 0, 0};
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
lv_canvas_set_draw_buf(canvas, mask);
lv_canvas_fill_bg(canvas, lv_color_black(), LV_OPA_TRANSP);

lv_layer_t layer;
lv_canvas_init_layer(canvas, &layer);

/*Draw a label to the canvas. The result "image" will be used as mask*/
lv_draw_label_dsc_t label_dsc;
lv_draw_label_dsc_init(&label_dsc);
label_dsc.color = lv_color_white();
label_dsc.align = LV_TEXT_ALIGN_CENTER;
label_dsc.text = txt;
label_dsc.font = &lv_font_montserrat_24;
lv_area_t a = {0, 0, w - 1, h - 1};
lv_draw_label(&layer, &label_dsc, &a);

lv_canvas_finish_layer(canvas, &layer);

lv_obj_delete(canvas);
}

/***
 * Draw label with gradient color
 */
void lv_example_label_4(void)
{
    /* Create the mask of a text by drawing it to a canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(mask, MASK_WIDTH, MASK_HEIGHT, LV_COLOR_FORMAT_L8);
    LV_DRAW_BUF_INIT_STATIC(mask);

    generate_mask(&mask, MASK_WIDTH, MASK_HEIGHT, "Text with gradient");

    /* Create an object from where the text will be masked out.
     * Now it's a rectangle with a gradient but it could be an image too*/
    lv_obj_t * grad = lv_obj_create(lv_screen_active());
    lv_obj_set_size(grad, MASK_WIDTH, MASK_HEIGHT);
    lv_obj_center(grad);
    lv_obj_set_style_bg_color(grad, lv_color_hex(0xffff0000), 0);
    lv_obj_set_style_bg_grad_color(grad, lv_color_hex(0x0000ff), 0);
    lv_obj_set_style_bg_grad_dir(grad, LV_GRAD_DIR_HOR, 0);
    lv_obj_set_style_bitmap_mask_src(grad, &mask, 0);
}
#endif

```

Personalizzazione dell'animazione a scorrimento circolare

```

#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES

/***
 * Show customizing the circular scrolling animation of a label with `LV_LABEL_LONG_MODE_SCROLL_CIRCULAR`
 * long mode.
 */
void lv_example_label_5(void)
{
    static lv_anim_t animation_template;
    static lv_style_t label_style;

    lv_anim_init(&animation_template);
    lv_anim_set_delay(&animation_template, 1000);           /*Wait 1 second to start the first scroll*/
    lv_anim_set_repeat_delay(&animation_template,
                           3000);   /*Repeat the scroll 3 seconds after the label scrolls back to the initial position*/
    lv_anim_set_repeat_count(&animation_template, LV_ANIM_REPEAT_INFINITE);

    /*Initialize the label style with the animation template*/
    lv_style_init(&label_style);
    lv_style_set_anim(&label_style, &animation_template);

    lv_obj_t * label1 = lv_label_create(lv_screen_active());
    lv_label_set_long_mode(label1, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);      /*Circular scroll*/
    lv_obj_set_width(label1, 150);
    lv_label_set_text(label1, "It is a circularly scrolling text. ");
    lv_obj_align(label1, LV_ALIGN_CENTER, 0, 40);
    lv_obj_add_style(label1, &label_style, LV_STATE_DEFAULT);           /*Add the style to the label*/
}
#endif

```

Carattere a spaziatura fissa

```

#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES && LV_FONT_MONTSERAT_20

static bool fix_w_get_glyph_dsc(const lv_font_t * font, lv_font_glyph_dsc_t * dsc, uint32_t letter,
                               uint32_t letter_next)
{
    bool ret = lv_font_get_glyph_dsc_fmt(font, dsc, letter, letter_next);
    if(!ret) return false;

    /* Set a fixed width */
    dsc->adv_w = 20;
    dsc->ofs_x = (dsc->adv_w - dsc->box_w) / 2;
    return true;
}

void lv_example_label_6(void)
{
    /* Clone the original font and override its behavior */
    static lv_font_t mono_font;
    mono_font = lv_font_montserrat_20;
    mono_font.get_glyph_dsc = fix_w_get_glyph_dsc;
}

```

(continues on next page)

(continua dalla pagina precedente)

```

/* Create a label with normal font */
lv_obj_t * label1 = lv_label_create(lv_screen_active());
lv_obj_set_style_text_font(label1, &lv_font_montserrat_20, 0);
lv_label_set_text(label1, "0123.Wabc");

/* Create a label with fixed-width glyph descriptor override */
lv_obj_t * label2 = lv_label_create(lv_screen_active());
lv_obj_set_y(label2, 30);
lv_obj_set_style_text_font(label2, &mono_font, 0);
lv_label_set_text(label2, "0123.Wabc");
}

#endif

```

Assegnare un tag di traduzione a un'etichetta

```

#include "../../lv_examples.h"

#if LV_USE_TRANSLATION && LV_USE_DROPDOWN && LV_USE_LABEL && LV_BUILD_EXAMPLES

/* Arrays are defined 'const' to place them in program space instead of RAM. */
static const char * const tags[] = {"tiger", "lion", "rabbit", "elephant", NULL};
static const char * const languages[] = {"English", "Deutsch", "Español", NULL};

static void add_static_translations(void)
{
    static const char * const translations[] = {
        "The Tiger", "Der Tiger", "El Tigre",
        "The Lion", "Der Löwe", "El León",
        "The Rabbit", "Das Kaninchen", "El Conejo",
        "The Elephant", "Der Elefant", "El Elefante",
    };
    lv_translation_add_static(languages, tags, translations);
}

static void language_change_cb(lv_event_t * e)
{
    static char selected_lang[20];

    lv_obj_t * dropdown = lv_event_get_target_obj(e);
    lv_dropdown_get_selected_str(dropdown, selected_lang, sizeof(selected_lang));
    lv_translation_set_language(selected_lang);
}

/**
 * Use a translation tag in labels
 */
void lv_example_label_7(void)
{
    lv_obj_set_flex_flow(lv_screen_active(), LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(lv_screen_active(), LV_ALIGN_CENTER, LV_ALIGN_CENTER, LV_ALIGN_CENTER);

    add_static_translations();
    const size_t tag_count = sizeof(tags) / sizeof(tags[0]) - 1;
    const size_t lang_count = sizeof(languages) / sizeof(languages[0]) - 1;

    /* Create a dropdown to be able to select the language */
    lv_obj_t * language_dropdown = lv_dropdown_create(lv_screen_active());
    lv_dropdown_clear_options(language_dropdown);

    for(size_t i = 0; i < lang_count; ++i) {
        lv_dropdown_add_option(language_dropdown, languages[i], i);
    }

    lv_obj_add_event_cb(language_dropdown, language_change_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_translation_set_language("English");

    /* Create a label for each tag */
    for(size_t i = 0; i < tag_count; ++i) {
        lv_obj_t * label = lv_label_create(lv_screen_active());
        lv_label_set_translation_tag(label, tags[i]);
    }
}

#endif /*LV_USE_TRANSLATION && LV_USE_DROPDOWN && LV_USE_LABEL && LV_BUILD_EXAMPLES*/

```

6.18.7 API

lv_label.h

lv_label_private.h

lv_obj_property_names.h

6.19 LED (lv_led)

6.19.1 Panoramica

I LED sono Widget rettangolari (o circolari) la cui luminosità può essere regolata. Con una luminosità inferiore, il colore del LED diventa più scuro.

6.19.2 Parti e Stili

- LV_LED_PART_MAIN utilizza le *tipiche proprietà di stile del background*.

6.19.3 Utilizzo

Color

È possibile impostare il colore del LED con `lv_led_set_color(led, lv_color_hex(0xff0080))`. Questo verrà utilizzato come colore di sfondo, colore del bordo e colore dell'ombra.

Luminosità

È possibile impostare la luminosità con `lv_led_set_brightness(led, brightness)`. Il valore di brightness deve essere compreso tra 0 (più scuro) e 255 (più chiaro).

Attiva/Disattiva

Si usano `lv_led_on(led)` e `lv_led_off(led)` per impostare la luminosità su un valore predefinito ON o OFF. `lv_led_toggle(led)` alterna tra lo stato ON e OFF.

È possibile impostare valori di luminosità LED ON e OFF personalizzati definendo le macro LV_LED_BRIGHT_MAX e LV_LED_BRIGHT_MIN nel progetto. I loro valori di default sono 80 e 255. Anche questi devono essere compresi nell'intervallo [0..255].

6.19.4 Eventi

I Widget LED non inviano eventi speciali.

Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.19.5 Tasti

I Widget LED non elaborano alcun *tasto*.

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.19.6 Esempio

LED con stile personalizzato

```
#include "../../lv_examples.h"
#if LV_USE_LED && LV_BUILD_EXAMPLES

/***
 * Create LED's with different brightness and color
 */
void lv_example_led_1(void)
{
    /*Create a LED and switch it OFF*/
    lv_obj_t * led1 = lv_led_create(lv_screen_active());
    lv_obj_align(led1, LV_ALIGN_CENTER, -80, 0);
    lv_led_off(led1);

    /*Copy the previous LED and set a brightness*/
    lv_obj_t * led2 = lv_led_create(lv_screen_active());
    lv_obj_align(led2, LV_ALIGN_CENTER, 0, 0);
    lv_led_set_brightness(led2, 150);
    lv_led_set_color(led2, lv_palette_main(LV_PALETTE_RED));

    /*Copy the previous LED and switch it ON*/
    lv_obj_t * led3 = lv_led_create(lv_screen_active());
```

(continues on next page)

(continua dalla pagina precedente)

```

    lv_obj_align(led3, LV_ALIGN_CENTER, 80, 0);
    lv_led_on(led3);
}

#endif

```

6.19.7 API

lv_led.h

lv_led_private.h

6.20 Line (lv_line)

6.20.1 Panoramica

Il Widget "Line" è in grado di disegnare linee rette tra un insieme di punti.

6.20.2 Parti e Stili

- *LV_PART_MAIN* utilizza le proprietà *tipiche del background* e lo stile della linea.

6.20.3 Utilizzo

Impostazione dei punti

I punti di una linea devono essere memorizzati in un array *lv_point_precise_t* e passati al Widget tramite la funzione *lv_line_set_points*(lines, point_array, point_cnt).

Le loro coordinate possono essere specificate come coordinate pixel grezze (ad esempio {5, 10}) o come percentuale del "bounding box" [riquadro di delimitazione] della linea utilizzando *lv_pct*(x). In quest'ultimo caso, potrebbe essere necessario impostare esplicitamente la larghezza/altezza della Linea utilizzando *lv_obj_set_width()* e *lv_obj_set_height()*, poiché i valori percentuali non espandono automaticamente il riquadro di delimitazione.

Dimensionamento automatico

Per default, la larghezza e l'altezza della Linea sono impostate su *LV_SIZE_CONTENT*. Ciò significa che le sue dimensioni verranno impostate automaticamente in modo da adattarsi a tutti i punti. Se le dimensioni venissero impostate esplicitamente, alcune parti della Linea potrebbero non essere visibili.

Inversione della y

Per default, il punto *y == 0* si trova nella parte superiore del Widget. Potrebbe essere contortintuitivo in alcuni casi, quindi le coordinate y possono essere invertite con *lv_line_set_y_invert*(line, true). In questo caso, *y == 0* sarà in fondo al Widget. L'inversione della y è disabilitata per default.

6.20.4 Eventi

I Widget Line inviano solo *eventi generici*.

Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.20.5 Tasti

Nessun *Tasto* viene elaborato dai Widget Line.

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.20.6 Esempio

Linea semplice

```
#include "../../lv_examples.h"
#if LV_USE_LINE && LV_BUILD_EXAMPLES

void lv_example_line_1(void)
{
    /*Create array for the points of the line*/
    static lv_point_precise_t line_points[] = { {5, 5}, {70, 70}, {120, 10}, {180, 60}, {240, 10} };

    /*Create style*/
    static lv_style_t style_line;
    lv_style_init(&style_line);
    lv_style_set_line_width(&style_line, 8);
    lv_style_set_line_color(&style_line, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_line_rounded(&style_line, true);

    /*Create a line and apply the new style*/
    lv_obj_t * linel;
    linel = lv_line_create(lv_screen_active());
    lv_line_set_points(linel, line_points, 5); /*Set the points*/
    lv_obj_add_style(linel, &style_line, 0);
    lv_obj_center(linel);
}

#endif
```

6.20.7 API

lv_line.h

lv_line_private.h

6.21 List (lv_list)

6.21.1 Panoramica

Il Widget List è fondamentalmente un rettangolo con layout verticale a cui è possibile aggiungere Pulsanti e Testo.

6.21.2 Parti e Stili

Background

- *LV_PART_MAIN* La parte principale della Lista che utilizza le *tipiche proprietà di stile del background*.
- *LV_PART_SCROLLBAR* La scrollbar. Consultare la documentazione di *Widget Base (lv_obj)* per i dettagli.

Pulsanti e Testo

- Consultare la documentazione del *Button* e della *Label*.

6.21.3 Utilizzo

Pulsanti

lv_list_add_button(list, icon, text) aggiunge un pulsante a larghezza intera con un'icona (che può essere un'immagine o un simbolo) e del testo. Questa funzione restituisce un puntatore al pulsante creato, utilizzabile, ad esempio, per aggiungere una callback di evento.

Il testo scorre orizzontalmente se è più lungo del pulsante.

Text

`lv_list_add_text(list, text)` aggiunge una stringa di testo. Questa funzione restituisce un puntatore all'etichetta creata, utilizzabile, ad esempio, per modificarne il testo con una delle funzioni `lv_label_set_text...()`.

6.21.4 Eventi

I Widget List non inviano eventi speciali, ma gli eventi possono essere inviati dai Pulsanti come di consueto.

Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.21.5 Tasti

Nessun *Tasto* viene elaborato dai Widget List.

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.21.6 Esempio

Lista semplice

```
#include "../../lv_examples.h"
#if LV_USE_LIST && LV_BUILD_EXAMPLES
static lv_obj_t * list1;

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_CLICKED) {
        LV_UNUSED(obj);
        LV_LOG_USER("Clicked: %s", lv_list_get_button_text(list1, obj));
    }
}
void lv_example_list_1(void)
{
    /*Create a list*/
    list1 = lv_list_create(lv_screen_active());
    lv_obj_set_size(list1, 180, 220);
    lv_obj_center(list1);

    /*Add buttons to the list*/
    lv_obj_t * btn;
    lv_list_add_text(list1, "File");
    btn = lv_list_add_button(list1, LV_SYMBOL_FILE, "New");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_DIRECTORY, "Open");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_SAVE, "Save");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_CLOSE, "Delete");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_EDIT, "Edit");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_list_add_text(list1, "Connectivity");
    btn = lv_list_add_button(list1, LV_SYMBOL_BLUETOOTH, "Bluetooth");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_GPS, "Navigation");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_USB, "USB");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_BATTERY_FULL, "Battery");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_list_add_text(list1, "Exit");
    btn = lv_list_add_button(list1, LV_SYMBOL_OK, "Apply");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_CLOSE, "Close");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
}

#endif
```

Ordinamento di un elenco utilizzando i pulsanti Su e Giù

```
#include "../../lv_examples.h"
#if LV_USE_LIST && LV_BUILD_EXAMPLES

static lv_obj_t * list1;
static lv_obj_t * list2;

static lv_obj_t * currentButton = NULL;

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_CLICKED) {
        LV_LOG_USER("Clicked: %s", lv_list_get_button_text(list1, obj));

        if(currentButton == obj) {
            currentButton = NULL;
        } else {
            currentButton = obj;
        }
        lv_obj_t * parent = lv_obj_get_parent(obj);
        int32_t i;
        for(i = 0; i < (int32_t)lv_obj_get_child_count(parent); i++) {
            lv_obj_t * child = lv_obj_get_child(parent, i);
            if(child == currentButton) {
                lv_obj_add_state(child, LV_STATE_CHECKED);
            } else {
                lv_obj_remove_state(child, LV_STATE_CHECKED);
            }
        }
    }
}

static void event_handler_top(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_CLICKED) {
        if(currentButton == NULL) return;
        lv_obj_move_to_index(currentButton, 0);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_up(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT)) {
        if(currentButton == NULL) return;
        int32_t index = lv_obj_get_index(currentButton);
        if(index < 0) return;
        lv_obj_move_to_index(currentButton, index - 1);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_center(lv_event_t * e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT)) {
        if(currentButton == NULL) return;

        lv_obj_t * parent = lv_obj_get_parent(currentButton);
        const int32_t pos = (int32_t)lv_obj_get_child_count(parent) / 2;

        lv_obj_move_to_index(currentButton, pos);

        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_dn(lv_event_t * e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT)) {
        if(currentButton == NULL) return;
        const int32_t index = lv_obj_get_index(currentButton);

        lv_obj_move_to_index(currentButton, index + 1);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_bottom(lv_event_t * e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_CLICKED) {
        if(currentButton == NULL) return;
        lv_obj_move_to_index(currentButton, -1);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_swap(lv_event_t * e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT)) {
        uint32_t cnt = lv_obj_get_child_count(list1);
        for(int i = 0; i < 100; i++) {
            lv_obj_t * obj = lv_obj_get_child(list1, (int32_t)lv_rand(0, cnt - 1));
            lv_obj_move_to_index(obj, (int32_t)lv_rand(0, cnt - 1));
            if(currentButton != NULL) {
                lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
            }
        }
    }
}

void lv_example_list_2(void)
{
```

(continues on next page)

(continua dalla pagina precedente)

```

/*Create a list*/
list1 = lv_list_create(lv_screen_active());
lv_obj_set_size(list1, lv_pct(60), lv_pct(100));
lv_obj_set_style(list1, 5, 0);

/*Add buttons to the list*/
lv_obj_t * btn;
int i;
for(i = 0; i < 15; i++) {
    btn = lv_button_create(list1);
    lv_obj_set_width(btn, lv_pct(50));
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_obj_t * lab = lv_label_create(btn);
    lv_label_set_text_fmt(lab, "Item %d", i);
}

/*Select the first button by default*/
currentButton = lv_obj_get_child(list1, 0);
lv_obj_add_state(currentButton, LV_STATE_CHECKED);

/*Create a second list with up and down buttons*/
list2 = lv_list_create(lv_screen_active());
lv_obj_set_size(list2, lv_pct(40), lv_pct(100));
lv_obj_align(list2, LV_ALIGN_TOP_RIGHT, 0, 0);
lv_obj_set_flex_flow(list2, LV_FLEX_FLOW_COLUMN);

btn = lv_list_add_button(list2, NULL, "Top");
lv_obj_add_event_cb(btn, event_handler_top, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);

btn = lv_list_add_button(list2, LV_SYMBOL_UP, "Up");
lv_obj_add_event_cb(btn, event_handler_up, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);

btn = lv_list_add_button(list2, LV_SYMBOL_LEFT, "Center");
lv_obj_add_event_cb(btn, event_handler_center, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);

btn = lv_list_add_button(list2, LV_SYMBOL_DOWN, "Down");
lv_obj_add_event_cb(btn, event_handler_dn, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);

btn = lv_list_add_button(list2, NULL, "Bottom");
lv_obj_add_event_cb(btn, event_handler_bottom, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);
}

#endif

```

6.21.7 API

lv_api_map_v8.h

lv_list.h

6.22 Lottie (lv_lottie)

6.22.1 Panoramica

Il Widget Lottie è in grado di analizzare, rasterizzare e riprodurre le animazioni Lottie.

Le animazioni Lottie sono animazioni vettoriali. Le si consideri una moderna combinazione di SVG e GIF.

Le animazioni possono essere scaricate da diverse fonti, come <https://lottiefiles.com/> oppure se ne possono creare di proprie, ad esempio, utilizzando Adobe After Effects.

Il Widget Lottie si basa su *Canvas* (*lv_canvas*) perché per eseguire il rendering dell'animazione l'utente deve fornire un buffer in cui memorizzare il fotogramma corrente.

6.22.2 Parti e Stili

- *LV_PART_MAIN* Il background dell'animazione Lottie. Si applicano le tipiche proprietà di stile del background, ma di solito viene lasciato trasparente.

6.22.3 Utilizzo

Dipendenze

Il Widget Lottie utilizza la libreria ThorVG che è integrata in LVGL. Per utilizzare le animazioni Lottie, è necessario abilitare *LV_USE_THORVG_INTERNAL* (per utilizzare il ThorVG integrato) o *LV_USE_THORVG_EXTERNAL*

(per linkarlo esternamente) in `lv_conf.h`. Per la grafica vettoriale in generale, è necessario abilitare anche `LV_USE_VECTOR_GRAPHIC`.

Poiché ThorVG è scritto in C++, quando si utilizza `LV_USE_THORVG_INTERNAL` assicurarsi di poter compilare i file cpp.

Impostare un buffer

Per eseguire il rendering dell'animazione, è necessario assegnare un buffer al Widget Lottie. Le animazioni vengono renderizzate in formato ARGB8888, pertanto la dimensione del buffer deve essere pari a `target_width x target_height x 4 byte`.

Per mantenere coerenti le dimensioni del buffer e quelle dell'animazione, la dimensione del Widget (ovvero la dimensione dell'animazione) viene impostata internamente sulle dimensioni del buffer.

Il buffer può essere impostato con `lv_lottie_set_buffer(lottie, w, h, buf)` o `lv_lottie_set_draw_buf(lottie, draw_buf)`.

Quando si utilizza un buffer di disegno, questo deve essere già inizializzato dall'utente con il formato colore `LV_COLOR_FORMAT_ARGB8888_PREMULTIPLIED`.

Impostare una sorgente

`lv_example_lottie_approve.c` contiene un'animazione di esempio. Invece di memorizzare la stringa JSON, viene memorizzato un array esadecimale per i seguenti motivi:

- per evitare l'escape del carattere " nel file JSON e
- alcuni compilatori non supportano stringhe molto lunghe.

`lvgl/scripts/filetohex.py` può essere utilizzato per convertire un file Lottie in un array esadecimale. Ad esempio:

```
./filetohex.py path/to/lottie.json --filter-character --null-terminate > out.txt
```

`--filter-character` filtra i caratteri non ASCII e `--null-terminate` assicura che venga aggiunto uno zero finale per chiudere correttamente la stringa.

Per creare un'animazione dai dati, si usa `lv_lottie_set_src_data(lottie, data, sizeof(data))`

Le animazioni Lottie possono essere aperte da file JSON utilizzando `lv_lottie_set_src_file(lottie, "path/to/file.json")`. Notare che il caricatore Lottie non supporta l'interfaccia del file system di LVGL, ma dovrebbe essere utilizzato un "path normale" senza lettera del driver.

Ottenere l'animazione

```
lv_anim_t * a = lv_lottie_get_anim(lottie)
```

restituisce l'animazione LVGL che controlla quella Lottie. Per default, l'animazione gira all'infinito a 60 FPS, tuttavia l'animazione LVGL può essere regolata liberamente.

6.22.4 Eventi

Nessun evento viene emesso dai Widget di Lottie.

Ulteriori Informazioni

Scoprire di più sugli [Eventi](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

6.22.5 Tasti

Nessun tasto viene elaborato dai Widget Lottie.

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.22.6 Esempio

Caricamento di un'animazione Lottie da un array

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_LOTTIE

/***
 * Load an lottie animation from data
 */
void lv_example_lottie_1(void)
{
    extern const uint8_t lv_example_lottie_approve[];
    extern const size_t lv_example_lottie_approve_size;

    lv_obj_t * lottie = lv_lottie_create(lv_screen_active());
    lv_lottie_set_src_data(lottie, lv_example_lottie_approve, lv_example_lottie_approve_size);

#if LV_DRAW_BUF_ALIGN == 4 && LV_DRAW_BUF_STRIDE_ALIGN == 1
    /*If there are no special requirements, just declare a buffer
     x4 because the Lottie is rendered in ARGB8888_PREMULTIPLIED format*/
    static uint8_t buf[64 * 64 * 4];
    lv_lottie_set_buffer(lottie, 64, 64, buf);
#else
    /*For GPUs and special alignment/stride setting use a draw_buf instead*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, 64, 64, LV_COLOR_FORMAT_ARGB8888_PREMULTIPLIED);
    lv_lottie_set_draw_buf(lottie, &draw_buf);
#endif

    lv_obj_center(lottie);
}

#else

void lv_example_lottie_1(void)
{
    /*fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Lottie cannot be previewed online");
    lv_obj_center(label);
}

#endif /*LV_USE_LOTTIE*/
#endif /*LV_BUILD_EXAMPLES*/
```

Caricamento di un'animazione Lottie da un file

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_LOTTIE

/***
 * Load an lottie animation from file
 */
void lv_example_lottie_2(void)
{
    lv_obj_t * lottie = lv_lottie_create(lv_screen_active());
    lv_lottie_set_src_file(lottie, "lvg/examples/widgets/lottie/lv_example_lottie_approve.json");

#if LV_DRAW_BUF_ALIGN == 4 && LV_DRAW_BUF_STRIDE_ALIGN == 1
    /*If there are no special requirements, just declare a buffer
     x4 because the Lottie is rendered in ARGB8888_PREMULTIPLIED format*/
    static uint8_t buf[64 * 64 * 4];
    lv_lottie_set_buffer(lottie, 64, 64, buf);
#else
    /*For GPUs and special alignment/stride setting use a draw_buf instead*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, 64, 64, LV_COLOR_FORMAT_ARGB8888_PREMULTIPLIED);
    lv_lottie_set_draw_buf(lottie, &draw_buf);
#endif

    lv_obj_center(lottie);
}

#else

void lv_example_lottie_2(void)
{
    /*fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Lottie cannot be previewed online");
    lv_obj_center(label);
}

#endif /*LV_USE_LOTTIE*/
#endif /*LV_BUILD_EXAMPLES*/
```

6.22.7 API

lv_lottie.h

lv_lottie_private.h

6.23 Menu (lv_menu)

6.23.1 Panoramica

Il Widget Menu può essere utilizzato per creare menù multilivello che gestiscono automaticamente la navigazione tra i livelli di menù e consentono all'utente di catturare la navigazione tra le pagine e gli eventi di clic.

6.23.2 Parti e Stili

Il Widget Menu è costituito dai seguenti Widget:

- Contenitore principale: `lv_menu_main_cont`
- Intestazione principale: `lv_menu_main_header_cont`
- Pulsante Back: *Pulsante (lv_button)*
- -__Icona pulsante Back: *Image (lv_image)*
- Pagina Principale: `lv_menu_page`
- Contenitore barra laterale: `lv_menu_sidebar_cont`
- Intestazione barra laterale: `lv_menu_sidebar_header_cont`
- Pulsante Back: *Pulsante (lv_button)*
- -__Icona pulsante Back: *Image (lv_image)*
- Pagina Barra Laterale: `lv_menu_page`

6.23.3 Utilizzo

Creare un Menù

`lv_menu_create(parent)` crea un nuovo Menù vuoto.

Questo crea un Widget Menu con questa gerarchia di oggetti:

- Menu: `lv_menu_t`
 - Hidden Sub-Page Storage: `lv_obj_t`
 - Contenitore principale: `lv_menu_main_cont_class`
 - * Intestazione principale: `lv_menu_main_header_cont_class`
 - Pulsante Back: *Pulsante (lv_button)*
 - -__Icona pulsante Back: *Image (lv_image)*
 - Titolo dell'intestazione principale: *Label (lv_label)* (default hidden)

Nota: non viene creata alcuna barra laterale. Se ne può creare una in seguito, volendo.

Modalità intestazione

Esistono le seguenti modalità di intestazione:

- `LV_MENU_HEADER_TOP_FIXED` L'intestazione è posizionata in alto. (Il default).
- `LV_MENU_HEADER_TOP_UNFIXED` L'intestazione è posizionata in alto e può essere fatta scorrere fuori dalla vista.
- `LV_MENU_HEADER_BOTTOM_FIXED` L'intestazione è posizionata in basso.

È possibile impostare le modalità dell'intestazione con `lv_menu_set_mode_header(menu, LV_MENU_HEADER...)`.

Modalità del pulsante Root back

Esistono le seguenti modalità del pulsante "root back":

- `LV_MENU_ROOT_BACK_BTN_DISABLED` (il default)
- `LV_MENU_ROOT_BACK_BTN_ENABLED`

È possibile impostare le modalità del pulsante "root back" con `lv_menu_set_mode_root_back_button(menu, LV_MENU_ROOT_BACK_BTN...)`.

Creare una Menu Page

`lv_menu_page_create(menu, title)` crea una nuova Menu Page vuoto. È possibile aggiungere qualsiasi Widget alla Page.

Impostare una Menu Page nell'area principale

Una volta creata una Menu Page, è possibile impostarla nell'area principale con `lv_menu_set_page(menu, page)`. NULL per cancellare il main e la cronologia del Menù.

Impostare una Menu Page nella barra laterale

Una volta creata una Menu Page, la si può impostare sulla barra laterale con `lv_menu_set_sidebar_page(menu, page)`. NULL per cancellare la barra laterale.

Collegamento tra Menu Page

Se, ad esempio, è stato creato un oggetto button nella Pagina principale. Quando si clicca sul Widget pulsante, si vuole che si apra una nuova Pagina, si usa `lv_menu_set_load_page_event(menu, btn, new page)`.

Creare un contenitore, una sezione e un separatore del Menù

È possibile creare i seguenti oggetti per semplificare la definizione dello stile del Menu:

- `lv_menu_cont_create(parent_page)` crea un nuovo contenitore vuoto.
- `lv_menu_section_create(parent_page)` crea una nuova sezione vuota.
- `lv_menu_separator_create(parent_page)` crea un separatore.

6.23.4 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato quando viene visualizzata una Page.
 - `lv_menu_get_cur_main_page(menu)` restituisce un puntatore alla Menu Page attualmente visualizzata nel contenitore principale.
 - `lv_menu_get_cur_sidebar_page(menu)` restituisce un puntatore alla Menu Page attualmente visualizzata nel contenitore della barra laterale.
- `LV_EVENT_CLICKED` Inviato quando si clicca su un "back button" in un'intestazione dalla barra principale o laterale. `LV_OBJ_FLAG_EVENT_BUBBLE` è abilitato sui pulsanti, in modo da poter aggiungere eventi al Menu stesso.
 - `lv_menu_back_button_is_root(menu, button)` per verificare se il pulsante è il "root back button".

Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.23.5 Tasti

Nessun *Tasto* viene elaborato dai Widget Menu.

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.23.6 Esempio

Menù semplice

```
#include "../../lv_examples.h"
#if LV_USE_MENU && LV_BUILD_EXAMPLES

void lv_example_menu_1(void)
{
    /*Create a menu object*/
    lv_obj_t * menu = lv_menu_create(lv_screen_active());
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create a sub page*/
    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(sub_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    /*Create a main page*/
    lv_obj_t * main_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 1");

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 2");

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 3 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_page);

    lv_menu_set_page(menu, main_page);
}

#endif
```

Menù semplice con pulsante principale

```
#include "../../lv_examples.h"
#if LV_USE_MENU && LV_USE_MSGBOX && LV_BUILD_EXAMPLES

static void back_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_obj_t * menu = (lv_obj_t *)lv_event_get_user_data(e);

    if(lv_menu_back_button_is_root(menu, obj)) {
        lv_obj_t * mbox1 = lv_msgbox_create(NULL);
        lv_msgbox_add_title(mbox1, "Hello");
        lv_msgbox_add_text(mbox1, "Root back btn click.");
        lv_msgbox_add_close_button(mbox1);
    }
}

void lv_example_menu_2(void)
{
    lv_obj_t * menu = lv_menu_create(lv_screen_active());
    lv_menu_set_mode_root_back_button(menu, LV_MENU_ROOT_BACK_BUTTON_ENABLED);
    lv_obj_add_event_cb(menu, back_event_handler, LV_EVENT_CLICKED, menu);
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create a sub page*/
    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(sub_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    /*Create a main page*/
    lv_obj_t * main_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 1");

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_label_set_text(label, "Item 2");

cont = lv_menu_cont_create(main_page);
label = lv_label_create(cont);
lv_label_set_text(label, "Item 3 (Click me!)");
lv_menu_set_load_page_event(menu, cont, sub_page);

lv_menu_set_page(menu, main_page);
}

#endif

```

Menù semplice con intestazione personalizzata

```

#include "../../lv_examples.h"
#if LV_USE_MENU && LV_BUILD_EXAMPLES

void lv_example_menu_3(void)
{
    /*Create a menu object*/
    lv_obj_t * menu = lv_menu_create(lv_screen_active());
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    /*Modify the header*/
    lv_obj_t * back_btn = lv_menu_get_main_header_back_button(menu);
    lv_obj_t * back_button_label = lv_label_create(back_btn);
    lv_label_set_text(back_button_label, "Back");

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create sub pages*/
    lv_obj_t * sub_1_page = lv_menu_page_create(menu, "Page 1");

    cont = lv_menu_cont_create(sub_1_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    lv_obj_t * sub_2_page = lv_menu_page_create(menu, "Page 2");

    cont = lv_menu_cont_create(sub_2_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    lv_obj_t * sub_3_page = lv_menu_page_create(menu, "Page 3");

    cont = lv_menu_cont_create(sub_3_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    /*Create a main page*/
    lv_obj_t * main_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 1 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_1_page);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 2 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_2_page);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 3 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_3_page);

    lv_menu_set_page(menu, main_page);
}

#endif

```

Menù semplice con pulsante mobile per aggiungere una nuova Menu Page

```

#include "../../lv_examples.h"
#if LV_USE_MENU && LV_BUILD_EXAMPLES

static uint32_t btn_cnt = 1;
static lv_obj_t * main_page;
static lv_obj_t * menu;

static void float_button_event_cb(lv_event_t * e)
{
    LV_UNUSED(e);

    btn_cnt++;

    lv_obj_t * cont;
    lv_obj_t * label;

    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(sub_page);
    label = lv_label_create(cont);
    lv_label_set_text_fmt(label, "Hello, I am hiding inside %" LV_PRIu32, btn_cnt);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text_fmt(label, "Item %" LV_PRIu32, btn_cnt);
    lv_menu_set_load_page_event(menu, cont, sub_page);

    lv_obj_scroll_to_view_recursive(cont, LV_ANIM_ON);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

}

void lv_example_menu_4(void)
{
    /*Create a menu object*/
    menu = lv_menu_create(lv_screen_active());
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create a sub page*/
    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(sub_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding inside the first item");

    /*Create a main page*/
    main_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 1");
    lv_menu_set_load_page_event(menu, cont, sub_page);

    lv_menu_set_page(menu, main_page);

    /*Create floating btn*/
    lv_obj_t * float_btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(float_btn, 50, 50);
    lv_obj_add_flag(float_btn, LV_OBJ_FLAG_FLOATING);
    lv_obj_align(float_btn, LV_ALIGN_BOTTOM_RIGHT, -10, -10);
    lv_obj_add_event_cb(float_btn, float_button_event_cb, LV_EVENT_CLICKED, menu);
    lv_obj_set_style_radius(float_btn, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_bg_image_src(float_btn, LV_SYMBOL_PLUS, 0);
    lv_obj_set_style_text_font(float_btn, lv_theme_get_font_large(float_btn), 0);
}
#endif

```

Menu complesso

```

#include "../../lv_examples.h"
#ifndef LV_USE_MENU && LV_USE_MSGBOX && LV_BUILD_EXAMPLES
    #error "lv_menu example requires LV_USE_MENU & LV_USE_MSGBOX & LV_BUILD_EXAMPLES"
#endif

typedef enum {
    LV_MENU_ITEM_BUILDER_VARIANT_1,
    LV_MENU_ITEM_BUILDER_VARIANT_2
} lv_menu_builder_variant_t;

static void back_event_handler(lv_event_t * e);
static void switch_handler(lv_event_t * e);
static lv_obj_t * root_page;
static lv_obj_t * create_text(lv_obj_t * parent, const char * icon, const char * txt,
                             lv_menu_builder_variant_t builder_variant);
static lv_obj_t * create_slider(lv_obj_t * parent,
                               const char * icon, const char * txt, int32_t min, int32_t max, int32_t val);
static lv_obj_t * create_switch(lv_obj_t * parent,
                               const char * icon, const char * txt, bool chk);

void lv_example_menu_5(void)
{
    lv_obj_t * menu = lv_menu_create(lv_screen_active());

    lv_color_t bg_color = lv_obj_get_style_bg_color(menu, LV_PART_MAIN);
    if(lv_color_brightness(bg_color) > 127) {
        lv_obj_set_style_bg_color(menu, lv_color_darken(lv_obj_get_style_bg_color(menu, LV_PART_MAIN), 10), 0);
    } else {
        lv_obj_set_style_bg_color(menu, lv_color_darken(lv_obj_get_style_bg_color(menu, LV_PART_MAIN), 50), 0);
    }
    lv_menu_set_mode_root_back_button(menu, LV_MENU_ROOT_BACK_BUTTON_ENABLED);
    lv_obj_add_event_cb(menu, back_event_handler, LV_EVENT_CLICKED, menu);
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * section;

    /*Create sub pages*/
    lv_obj_t * sub_mechanics_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_mechanics_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
    lv_menu_separator_create(sub_mechanics_page);
    section = lv_menu_section_create(sub_mechanics_page);
    create_slider(section, LV_SYMBOL_SETTINGS, "Velocity", 0, 150, 120);
    create_slider(section, LV_SYMBOL_SETTINGS, "Acceleration", 0, 150, 50);
    create_slider(section, LV_SYMBOL_SETTINGS, "Weight limit", 0, 150, 80);

    lv_obj_t * sub_sound_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_sound_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
    lv_menu_separator_create(sub_sound_page);
    section = lv_menu_section_create(sub_sound_page);
    create_switch(section, LV_SYMBOL_AUDIO, "Sound", false);

    lv_obj_t * sub_display_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_display_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
    lv_menu_separator_create(sub_display_page);
    section = lv_menu_section_create(sub_display_page);
    create_slider(section, LV_SYMBOL_SETTINGS, "Brightness", 0, 150, 100);

    lv_obj_t * sub_software_info_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_software_info_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
    section = lv_menu_section_create(sub_software_info_page);
    create_text(section, NULL, "Version 1.0", LV_MENU_ITEM_BUILDER_VARIANT_1);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * sub_legal_info_page = lv_menu_page_create(menu, NULL);
lv_obj_set_style_pad_hor(sub_legal_info_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN),
    0);
section = lv_menu_section_create(sub_legal_info_page);
for(uint32_t i = 0; i < 15; i++) {
    create_text(section, NULL,
        "This is a long long long long long long long long text, if it is long enough it may scroll.",
        LV_MENU_ITEM_BUILDER_VARIANT_1);
}

lv_obj_t * sub_about_page = lv_menu_page_create(menu, NULL);
lv_obj_set_style_pad_hor(sub_about_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
lv_menu_separator_create(sub_about_page);
section = lv_menu_section_create(sub_about_page);
cont = create_text(section, NULL, "Software information", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_software_info_page);
cont = create_text(section, NULL, "Legal information", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_legal_info_page);

lv_obj_t * sub_menu_mode_page = lv_menu_page_create(menu, NULL);
lv_obj_set_style_pad_hor(sub_menu_mode_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
lv_menu_separator_create(sub_menu_mode_page);
section = lv_menu_section_create(sub_menu_mode_page);
cont = create_switch(section, LV_SYMBOL_AUDIO, "Sidebar enable", true);
lv_obj_add_event_cb(lv_obj_get_child(cont, 2), switch_handler, LV_EVENT_VALUE_CHANGED, menu);

/*Create a root page*/
root_page = lv_menu_page_create(menu, "Settings");
lv_obj_set_style_pad_hor(root_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
section = lv_menu_section_create(root_page);
cont = create_text(section, LV_SYMBOL_SETTINGS, "Mechanics", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_mechanics_page);
cont = create_text(section, LV_SYMBOL_AUDIO, "Sound", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_sound_page);
cont = create_text(section, LV_SYMBOL_SETTINGS, "Display", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_display_page);

create_text(root_page, NULL, "Others", LV_MENU_ITEM_BUILDER_VARIANT_1);
section = lv_menu_section_create(root_page);
cont = create_text(section, NULL, "About", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_about_page);
cont = create_text(section, LV_SYMBOL_SETTINGS, "Menu mode", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_menu_mode_page);

lv_menu_set_sidebar_page(menu, root_page);

lv_obj_send_event(lv_obj_get_child(lv_obj_get_child(lv_menu_get_cur_sidebar_page(menu), 0), 0), LV_EVENT_CLICKED,
    NULL);
}

static void back_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_obj_t * menu = (lv_obj_t *)lv_event_get_user_data(e);

    if(lv_menu_back_button_is_root(menu, obj)) {
        lv_obj_t * mbox1 = lv_msgbox_create(NULL);
        lv_msgbox_add_title(mbox1, "Hello");
        lv_msgbox_add_text(mbox1, "Root back btn click.");
        lv_msgbox_add_close_button(mbox1);
    }
}

static void switch_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * menu = (lv_obj_t *)lv_event_get_user_data(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        if(lv_obj_has_state(obj, LV_STATE_CHECKED)) {
            lv_menu_set_page(menu, NULL);
            lv_menu_set_sidebar_page(menu, root_page);
            lv_obj_send_event(lv_obj_get_child(lv_obj_get_child(lv_menu_get_cur_sidebar_page(menu), 0), 0), LV_EVENT_CLICKED,
                NULL);
        }
        else {
            lv_menu_set_sidebar_page(menu, NULL);
            lv_menu_clear_history(menu); /* Clear history because we will be showing the root page later */
            lv_menu_set_page(menu, root_page);
        }
    }
}

static lv_obj_t * create_text(lv_obj_t * parent, const char * icon, const char * txt,
    lv_menu_builder_variant_t builder_variant)
{
    lv_obj_t * obj = lv_menu_cont_create(parent);

    lv_obj_t * img = NULL;
    lv_obj_t * label = NULL;

    if(icon) {
        img = lv_image_create(obj);
        lv_image_set_src(img, icon);
    }

    if(txt) {
        label = lv_label_create(obj);
        lv_label_set_text(label, txt);
        lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
        lv_obj_set_flex_grow(label, 1);
    }

    if(builder_variant == LV_MENU_ITEM_BUILDER_VARIANT_2 && icon && txt) {
        lv_obj_add_flag(img, LV_OBJ_FLAG_FLEX_IN_NEW_TRACK);
        lv_obj_swap(img, label);
    }

    return obj;
}

static lv_obj_t * create_slider(lv_obj_t * parent, const char * icon, const char * txt, int32_t min, int32_t max,
    lv_menu_builder_variant_t builder_variant)
{
}

```

(continues on next page)

(continua dalla pagina precedente)

```

    int32_t val)
{
    lv_obj_t * obj = create_text(parent, icon, txt, LV_MENU_ITEM_BUILDER_VARIANT_2);

    lv_obj_t * slider = lv_slider_create(obj);
    lv_obj_set_flex_grow(slider, 1);
    lv_slider_set_range(slider, min, max);
    lv_slider_set_value(slider, val, LV_ANIM_OFF);

    if(icon == NULL) {
        lv_obj_add_flag(slider, LV_OBJ_FLAG_FLEX_IN_NEW_TRACK);
    }

    return obj;
}

static lv_obj_t * create_switch(lv_obj_t * parent, const char * icon, const char * txt, bool chk)
{
    lv_obj_t * obj = create_text(parent, icon, txt, LV_MENU_ITEM_BUILDER_VARIANT_1);

    lv_obj_t * sw = lv_switch_create(obj);
    lv_obj_add_state(sw, chk ? LV_STATE_CHECKED : LV_STATE_DEFAULT);

    return obj;
}

#endif

```

6.23.7 API

lv_menu.h

lv_menu_private.h

6.24 Message Box (*lv_msgbox*)

6.24.1 Panoramica

I "Message box" [finestre di messaggio] fungono da pop-up. Sono costituite da un'area di contenuto con un helper per l'aggiunta di testo, un'intestazione facoltativa (che può contenere un titolo, un pulsante di chiusura e altri pulsanti) e un più di pagina facoltativo con pulsanti.

Il testo verrà suddiviso in più righe e l'altezza verrà impostata automaticamente. Se l'altezza viene impostata a livello di codice, il contenuto diventerà scorrevole.

La finestra di messaggio può essere modale (bloccando i clic sul resto dello schermo) o non modale.

6.24.2 Parti e Stili

La finestra di messaggio è costruita a partire da altri widget, quindi si può consultare la documentazione di questi Widget per i dettagli.

- Contenuto, intestazione e più di pagina: *Widget Base (lv_obj)*
- Pulsanti: *Pulsante (lv_button)*
- Titolo e testo del contenuto: *Label (lv_label)*

6.24.3 Utilizzo

Creare una message box

lv_msgbox_create(parent) crea una message box. Se parent è NULL, la finestra di messaggio sarà modale e utilizzerà il *Layer* del *Display di Default* come genitore.

Aggiunta di pulsanti

Per aggiungere un pulsante [OK], [Cancel] o altri pulsanti per consentire all'utente di scegliere tra diverse risposte, aggiungere ciascun pulsante utilizzando la funzione *lv_msgbox_add_footer_button*(msgbox, btn_text). La chiamata a questa funzione aggiunge un più di pagina (contenitore) se non ne è già presente uno e restituisce un puntatore al pulsante creato, che può essere utilizzato per aggiungere eventi LV_EVENT_CLICKED (o altri) per rilevare e agire sulla risposta dell'utente.

I pulsanti del più di pagina così aggiunti sono equidistanti e centrati.

È anche possibile aggiungere pulsanti all'intestazione con `lv_msgbox_add_header_button(msgbox, symbol)`. I pulsanti aggiunti in questo modo vengono aggiunti all'estremità destra dell'intestazione.

Ottenere le parti

I componenti costitutivi della finestra di messaggio possono essere ottenuti utilizzando le seguenti funzioni:

```
lv_obj_t * lv_msgbox_get_content(lv_obj_t * msgbox);
lv_obj_t * lv_msgbox_get_title(lv_obj_t * msgbox);
lv_obj_t * lv_msgbox_get_header(lv_obj_t * msgbox);
lv_obj_t * lv_msgbox_get_footer(lv_obj_t * msgbox);
```

Le funzioni che aggiungono qualcosa alla finestra di messaggio restituiscono un puntatore al Widget appena aggiunto:

```
lv_obj_t * lv_msgbox_add_text(lv_obj_t * msgbox, const char * text);
lv_obj_t * lv_msgbox_add_title(lv_obj_t * msgbox, const char * title);
lv_obj_t * lv_msgbox_add_close_button(lv_obj_t * msgbox);
lv_obj_t * lv_msgbox_add_header_button(lv_obj_t * msgbox, const void * symbol);
lv_obj_t * lv_msgbox_add_footer_button(lv_obj_t * msgbox, const char * text);
```

Chiudere la finestra di messaggio

`lv_msgbox_close(msgbox)` chiude (elimina) la finestra di messaggio.

`lv_msgbox_close_async(msgbox)` chiude (elimina) la finestra di messaggio in modo asincrono. Questo è utile se si desidera che la finestra di messaggio si chiuda alla successiva chiamata a `lv_timer_handler` anziché immediatamente.

6.24.4 Eventi

I Widget "Message Box" non inviano eventi speciali. Per i dettagli, consultare la documentazione di questi widget: *Pulsante* (`lv_button`) e *Label* (`lv_label`).

Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.24.5 Tasti

Nessun *Tasto* viene elaborato dai Widget "Message Box".

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.24.6 Esempio

Message box semplice

```
#include "../../lv_examples.h"
#if LV_USE_MSGBOX && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * btn = lv_event_get_target_obj(e);
    lv_obj_t * label = lv_obj_get_child(btn, 0);
    LV_UNUSED(label);
    LV_LOG_USER("Button %s clicked", lv_label_get_text(label));
}

void lv_example_msgbox_1(void)
{
    lv_obj_t * mbox1 = lv_msgbox_create(NULL);

    lv_msgbox_add_title(mbox1, "Hello");

    lv_msgbox_add_text(mbox1, "This is a message box with two buttons.");
    lv_msgbox_add_close_button(mbox1);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * btn;
btn = lv_msgbox_add_footer_button(mbox, "Apply");
lv_obj_add_event_cb(btn, event_cb, LV_EVENT_CLICKED, NULL);
btn = lv_msgbox_add_footer_button(mbox, "Cancel");
lv_obj_add_event_cb(btn, event_cb, LV_EVENT_CLICKED, NULL);
return;
}

#endif

```

Message box con scrolling e stile

```

#include "../../lv_examples.h"
#if LV_USE_MSGBOX && LV_BUILD_EXAMPLES

static void minimize_button_event_cb(lv_event_t * e)
{
    lv_obj_t * mbox = (lv_obj_t *) lv_event_get_user_data(e);
    lv_obj_add_flag(mbox, LV_OBJ_FLAG_HIDDEN);
}

void lv_example_msgbox_2(void)
{
    lv_obj_t * setting = lv_msgbox_create(lv_screen_active());
    lv_obj_set_style_clip_corner(setting, true, 0);

    /* setting fixed size */
    lv_obj_set_size(setting, 300, 200);

    /* setting's titlebar/header */
    lv_msgbox_add_title(setting, "Setting");
    lv_obj_t * minimize_button = lv_msgbox_add_header_button(setting, LV_SYMBOL_MINUS);
    lv_obj_add_event_cb(minimize_button, minimize_button_event_cb, LV_EVENT_CLICKED, setting);
    lv_msgbox_add_close_button(setting);

    /* setting's content */
    lv_obj_t * content = lv_msgbox_get_content(setting);
    lv_obj_set_flex_flow(content, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(content, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);
    lv_obj_set_style_pad_right(content, -1, LV_PART_SCROLLBAR);

    lv_obj_t * cont_brightness = lv_obj_create(content);
    lv_obj_set_size(cont_brightness, lv_pct(100), LV_SIZE_CONTENT);
    lv_obj_set_flex_flow(cont_brightness, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(cont_brightness, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER);

    lv_obj_t * lb_brightness = lv_label_create(cont_brightness);
    lv_label_set_text(lb_brightness, "Brightness : ");
    lv_obj_t * slider_brightness = lv_slider_create(cont_brightness);
    lv_obj_set_width(slider_brightness, lv_pct(100));
    lv_slider_set_value(slider_brightness, 50, LV_ANIM_OFF);

    lv_obj_t * cont_speed = lv_obj_create(content);
    lv_obj_set_size(cont_speed, lv_pct(100), LV_SIZE_CONTENT);
    lv_obj_set_flex_flow(cont_speed, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(cont_speed, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER);

    lv_obj_t * lb_speed = lv_label_create(cont_speed);
    lv_label_set_text(lb_speed, "Speed : ");
    lv_obj_t * slider_speed = lv_slider_create(cont_speed);
    lv_obj_set_width(slider_speed, lv_pct(100));
    lv_slider_set_value(slider_speed, 80, LV_ANIM_OFF);

    /* footer */
    lv_obj_t * apply_button = lv_msgbox_add_footer_button(setting, "Apply");
    lv_obj_set_flex_grow(apply_button, 1);

    lv_obj_t * cancel_button = lv_msgbox_add_footer_button(setting, "Cancel");
    lv_obj_set_flex_grow(cancel_button, 1);

    lv_obj_t * footer = lv_msgbox_get_footer(setting);
    lv_obj_set_style_bg_color(footer, lv_palette_main(LV_PALETTE_INDIGO), 0);
    lv_obj_set_style_bg_opa(footer, LV_OPA_100, 0);
}
#endif

```

6.24.7 API

lv_msgbox.h

lv_msgbox_private.h

6.25 Roller (lv_roller)

6.25.1 Panoramica

Il Roller consente all'utente finale di selezionare un elemento da una lista scorrendola. L'elemento al centro è quello selezionato e normalmente si distingue dagli altri per i diversi stili applicati.

6.25.2 Parti e Stili

- ***LV_PART_MAIN*** Il background del roller usa le proprietà di stile *tipiche del background* e del testo.
 - Lo stile `text_line_space` regola lo spazio tra gli elementi della lista. Si usa `lv_obj_set_style_text_line_space()` per impostare questo valore.
 - Quando il Roller scorre e non si ferma esattamente su un elemento, scorrerà automaticamente fino all'elemento valido più vicino in `anim_time` millisecondi, come specificato nello stile `anim_duration`. Si usa `lv_obj_set_style_anim_duration()` per impostare questo valore.
- ***LV_PART_SELECTED*** L'elemento selezionato (visualizzato al centro del Roller). Oltre alle tipiche proprietà di background, utilizza le proprietà di stile del testo per modificare l'aspetto del testo dell'elemento selezionato.

6.25.3 Utilizzo

Impostazione degli elementi della lista

Gli elementi della lista vengono passati al Roller come stringa con `lv_roller_set_options(roller, string_list, LV_ROLLER_MODE_NORMAL)`. Gli elementi devono essere separati da \n. Per esempio: "First\nSecond\nThird".

`LV_ROLLER_MODE_INFINITE` rende il Roller circolare.

È possibile selezionare un'opzione a livello di codice con `lv_roller_set_selected(roller, id, LV_ANIM_ON)`, dove `id` è l'indice a partire da zero dell'elemento della lista da selezionare.

Se non si conosce l'indice di un'opzione, è possibile selezionare un elemento anche con `lv_roller_set_selected_str(roller, str, LV_ANIM_ON)`, dove `str` la stringa corrispondente a uno degli elementi dell'elenco.

Prelevare l'opzione selezionata

Per ottenere l'*indice* dell'elemento attualmente selezionato, si usa `lv_roller_get_selected(roller)`.

`lv_roller_get_selected_str(roller, buf, buf_size)` copierà il nome dell'elemento selezionato in `buf`.

Righe visibili

Il numero di righe visibili può essere regolato con `lv_roller_set_visible_row_count(roller, num)`.

Questa funzione calcola l'altezza con lo stile corrente. Se il font, l'interlinea, lo spessore del bordo, ecc. del Roller cambiano, questa funzione deve essere richiamata nuovamente.

Data binding

Per familiarizzare con "observer", "subject" e data binding in generale, visitare la pagina [Observer](#).

Questo metodo di sottoscrizione a un Subject intero influisce direttamente sul valore intero di un Widget Roller. Si noti che si tratta di un binding bidirezionale (Subject <==> Widget), quindi l'interazione diretta di un utente finale col Widget Roller aggiorna il valore del Subject e viceversa.

Supporta solo soggetti interi.

- `lv_roller_bind_value(roller, &subject)`

6.25.4 Eventi

- ***LV_EVENT_VALUE_CHANGED*** Inviato quando viene selezionato un nuovo elemento dell'elenco.

Ulteriori Informazioni

Scoprire di più sugli [Eventi](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

6.25.5 Tasti

- LV_KEY_RIGHT/DOWN Seleziona l'opzione successiva
- LV_KEY_LEFT/UP Seleziona l'opzione precedente
- LY_KEY_ENTER Accetta l'opzione selezionata (invia l'evento `LV_EVENT_VALUE_CHANGED`)

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.25.6 Esempio

Roller semplice

```
#include "../../lv_examples.h"
#if LV_USE_ROLLER && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_roller_get_selected_str(obj, buf, sizeof(buf));
        LV_LOG_USER("Selected month: %s\n", buf);
    }
}

/**
 * An infinite roller with the name of the months
 */
void lv_example_roller_1(void)
{
    lv_obj_t * roller1 = lv_roller_create(lv_screen_active());
    lv_roller_set_options(roller1,
        "January\n"
        "February\n"
        "March\n"
        "April\n"
        "May\n"
        "June\n"
        "July\n"
        "August\n"
        "September\n"
        "October\n"
        "November\n"
        "December",
        LV_ROLLER_MODE_INFINITE);

    lv_roller_set_visible_row_count(roller1, 4);
    lv_obj_center(roller1);
    lv_obj_add_event_cb(roller1, event_handler, LV_EVENT_ALL, NULL);
}
#endif
```

Roller con stili

```
#include "../../lv_examples.h"
#if LV_USE_ROLLER && LV_FONT_MONTSEERRAT_22 && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_roller_get_selected_str(obj, buf, sizeof(buf));
        LV_LOG_USER("Selected value: %s", buf);
    }
}

/**
 * Roller with various alignments and larger text in the selected area
 */
void lv_example_roller_2(void)
{
    /*A style to make the selected option larger*/
    static lv_style_t style_sel;
    lv_style_init(&style_sel);
    lv_style_set_text_font(&style_sel, &lv_font_montserrat_22);
    lv_style_set_bg_color(&style_sel, lv_color_hex3(0xf88));
    lv_style_set_border_width(&style_sel, 2);
    lv_style_set_border_color(&style_sel, lv_color_hex3(0xf00));

    const char * opts = "1\n2\n3\n4\n5\n6\n7\n8\n9\n10";
    lv_obj_t * roller;

    /*A roller on the left with left aligned text, and custom width*/
    roller = lv_roller_create(lv_screen_active());
    lv_roller_set_options(roller, opts, LV_ROLLER_MODE_NORMAL);
    lv_roller_set_visible_row_count(roller, 2);
    lv_obj_set_width(roller, 100);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_add_style(roller, &style_sel, LV_PART_SELECTED);
lv_obj_set_style_text_align(roller, LV_TEXT_ALIGN_LEFT, 0);
lv_obj_set_style_bg_color(roller, lv_color_hex3(0x0f0), 0);
lv_obj_set_style_bg_grad_color(roller, lv_color_hex3(0xfa), 0);
lv_obj_set_style_bg_grad_dir(roller, LV_GRAD_DIR_VER, 0);
lv_obj_align(roller, LV_ALIGN_LEFT_MID, 10, 0);
lv_obj_add_event_cb(roller, event_handler, LV_EVENT_ALL, NULL);
lv_roller_set_selected(roller, 2, LV_ANIM_OFF);

/*A roller on the middle with center aligned text, and auto (default) width*/
roller = lv_roller_create(lv_screen_active());
lv_roller_set_options(roller, opts, LV_ROLLER_MODE_NORMAL);
lv_roller_set_visible_row_count(roller, 3);
lv_obj_add_style(roller, &style_sel, LV_PART_SELECTED);
lv_obj_align(roller, LV_ALIGN_CENTER, 0, 0);
lv_obj_add_event_cb(roller, event_handler, LV_EVENT_ALL, NULL);
lv_roller_set_selected(roller, 5, LV_ANIM_OFF);

/*A roller on the right with right aligned text, and custom width*/
roller = lv_roller_create(lv_screen_active());
lv_roller_set_options(roller, opts, LV_ROLLER_MODE_NORMAL);
lv_roller_set_visible_row_count(roller, 4);
lv_obj_set_width(roller, 80);
lv_obj_add_style(roller, &style_sel, LV_PART_SELECTED);
lv_obj_set_style_text_align(roller, LV_TEXT_ALIGN_RIGHT, 0);
lv_obj_align(roller, LV_ALIGN_RIGHT_MID, -10, 0);
lv_obj_add_event_cb(roller, event_handler, LV_EVENT_ALL, NULL);
lv_roller_set_selected(roller, 8, LV_ANIM_OFF);
}

#endif

```

Aggiungere una maschera di dissolvenza al roller

```

#include "../../lv_examples.h"

#if LV_USE_ROLLER && LV_DRAW_SW_COMPLEX && LV_BUILD_EXAMPLES && 1

static void generate_mask(lv_draw_buf_t * mask)
{
    /*Create a "8 bit alpha" canvas and clear it*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, mask);
    lv_canvas_fill_bg(canvas, lv_color_white(), LV_OPA_TRANSP);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    /*Draw a label to the canvas. The result "image" will be used as mask*/
    lv_draw_rect_dsc_t rect_dsc;
    lv_draw_rect_dsc_init(&rect_dsc);
    rect_dsc.bg.grad.dir = LV_GRAD_DIR_VER;
    rect_dsc.bg.grad.stops[0].color = lv_color_black();
    rect_dsc.bg.grad.stops[1].color = lv_color_white();
    rect_dsc.bg.grad.stops[0].opa = LV_OPA_COVER;
    rect_dsc.bg.grad.stops[1].opa = LV_OPA_COVER;
    lv_area_t a = {0, 0, mask->header.w - 1, mask->header.h / 2 - 10};
    lv_draw_rect(&layer, &rect_dsc, &a);

    a.y1 = mask->header.h / 2 + 10;
    a.y2 = mask->header.h - 1;
    rect_dsc.bg.grad.stops[0].color = lv_color_white();
    rect_dsc.bg.grad.stops[1].color = lv_color_black();
    lv_draw_rect(&layer, &rect_dsc, &a);

    lv_canvas_finish_layer(canvas, &layer);

    /*Comment it to make the mask visible*/
    lv_obj_delete(canvas);
}

/***
 * Add a fade mask to roller.
 */
void lv_example_roller_3(void)
{
    lv_obj_set_style_bg_color(lv_screen_active(), lv_palette_main(LV_PALETTE_BLUE_GREY), 0);

    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_bg_color(&style, lv_color_black());
    lv_style_set_text_color(&style, lv_color_white());
    lv_style_set_border_width(&style, 0);
    lv_style_set_radius(&style, 0);

    lv_obj_t * roller1 = lv_roller_create(lv_screen_active());
    lv_obj_add_style(roller1, &style, 0);
    lv_obj_set_style_bg_opa(roller1, LV_OPA_50, LV_PART_SELECTED);

    lv_roller_set_options(roller1,
        "January\n"
        "February\n"
        "March\n"
        "April\n"
        "May\n"
        "June\n"
        "July\n"
        "August\n"
        "September\n"
        "October\n"
        "November\n"
        "December",
        LV_ROLLER_MODE_NORMAL);

    lv_obj_center(roller1);
    lv_roller_set_visible_row_count(roller1, 4);

    /* Create the mask to make the top and bottom part of roller faded.

```

(continues on next page)

(continua dalla pagina precedente)

```
* The width and height are empirical values for simplicity/
LV_DRAW_BUF_DEFINE_STATIC(mask, 130, 150, LV_COLOR_FORMAT_L8);
LV_DRAW_BUF_INIT_STATIC(mask);

generate_mask(&mask);
lv_obj_set_style_bitmap_mask_src(roller1, &mask, 0);
}

#endif
```

6.25.7 API

lv_api_map_v8.h

lv_obj_property_names.h

lv_roller.h

lv_roller_private.h

6.26 Scale (lv_scale)

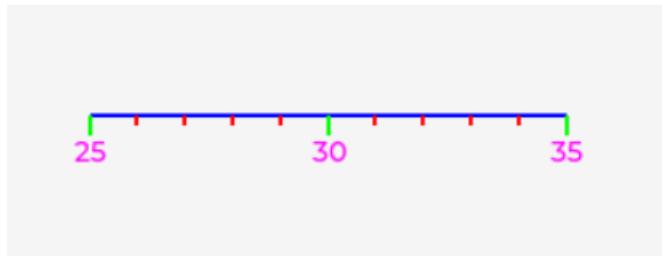
6.26.1 Panoramica

I Widget "Scale" mostrano scale lineari o circolari con intervalli, conteggi di tacche, posizionamento, etichettatura e sottosezioni (*Sezioni*) configurabili con stile personalizzato.

6.26.2 Parti e Stili

Il Widget "Scale" è composto dalle seguenti tre parti:

- *LV_PART_MAIN* Linea principale --- la linea blu nell'immagine di esempio.
- *LV_PART_ITEMS* Tacche secondarie --- le tacche secondarie rosse nell'immagine di esempio.
- *LV_PART_INDICATOR* Tacche principali e relative etichette (se abilitate) --- le tacche principali verdi e le etichette rosa nell'immagine di esempio.



6.26.3 Utilizzo

Modalità

Quando viene creato un Widget "Scale", inizia in MODALITÀ *LV_SCALE_MODE_HORIZONTAL_BOTTOM*. Questo rende la scala orizzontale con le tacche sotto la linea. Se c'è bisogno di una forma, un orientamento o una posizione di tacca diversa, si usa *lv_scale_set_mode(scale, mode)*, dove mode può essere uno qualsiasi di questi valori:

- *LV_SCALE_MODE_HORIZONTAL_TOP*
- *LV_SCALE_MODE_HORIZONTAL_BOTTOM*
- *LV_SCALE_MODE_VERTICAL_LEFT*
- *LV_SCALE_MODE_VERTICAL_RIGHT*
- *LV_SCALE_MODE_ROUND_INNER*
- *LV_SCALE_MODE_ROUND_OUTER*

Impostazione del range

Una "Scale" inizia la sua vita con un intervallo numerico predefinito di [0..100] e un intervallo angolare predefinito di 270. È possibile modificare questi intervalli con:

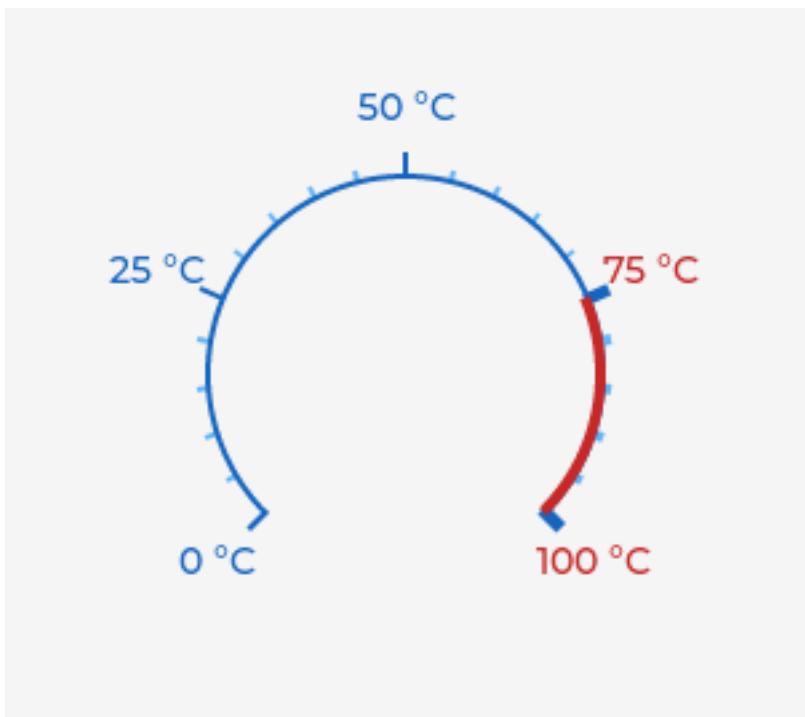
- `lv_scale_set_range(scale, min, max)`,
- `lv_scale_set_angle_range(scale, angle_range)`

dove `min` e `max` diventeranno i valori numerici minimo e massimo della Scale, e `angle_range` è l'angolo tra i limiti minimo e massimo della Scale.

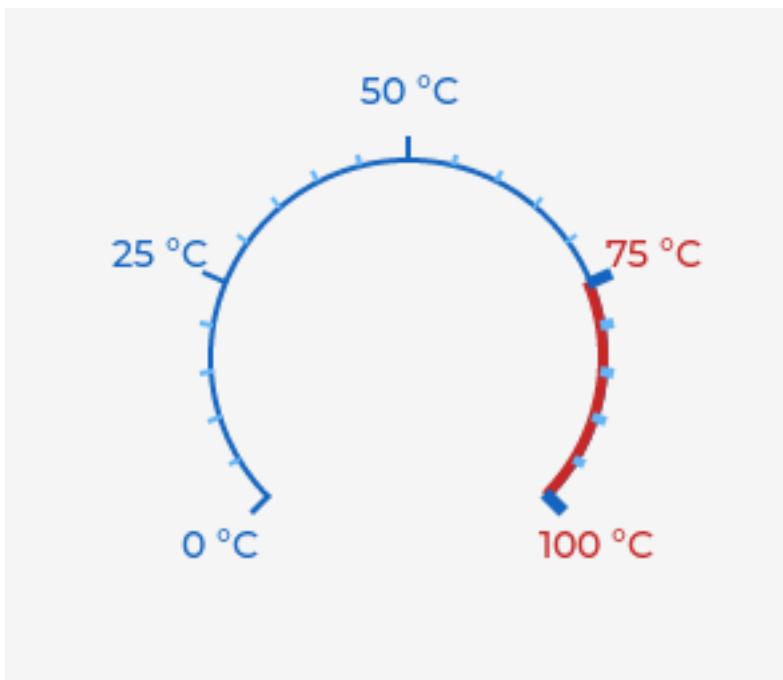
Ordine di disegno delle tacche

Normalmente, le tacche e le relative etichette vengono disegnate per prime, seguite dalla linea principale, dando l'impressione che le tacche e le relative etichette siano posizionate sotto la linea principale quando si sovrappongono. È possibile invertire questa sequenza facendo apparire le tacche e le etichette sopra la linea principale, utilizzando `lv_scale_set_draw_ticks_on_top(scale, true)`. (Questo effetto può essere invertito passando `false`.)

Esempio con tacche ed etichette disegnate *sotto* la linea principale (il default):



Esempio con tacche ed etichette disegnate *sopra* la linea principale:



Configurazione delle tacche

È possibile configurare le tacche maggiori e minori di una "Scale" chiamando 2 funzioni:

- `lv_scale_set_total_tick_count(scale, total_tick_count)` e
- `lv_scale_set_major_tick_every(scale, nth_tick)`.

Per far sì che le etichette vengano disegnate con le tacche maggiori, si chiama `lv_scale_set_label_show(scale, true)`. (Passare `false` per nasconderle di nuovo).

Per default, le etichette mostrate sono i valori numerici della scala nei punti delle tacche maggiori. È possibile specificare un contenuto di etichetta diverso chiamando `lv_scale_set_text_src(scale, custom_labels)` dove `custom_labels` è un array di puntatori a stringa. Esempio:

```
static char * custom_labels[3] = {"One", "Two", NULL};
```

Si noti che per terminare l'elenco viene utilizzato un puntatore `NULL`.

Il contenuto del buffer a cui si punta deve rimanere valido per tutta la durata della Scale.

Per una "Scale" in una delle modalità `..._ROUND_...`, è possibile far ruotare le etichette dei tick principali in modo che corrispondano alla rotazione degli stessi utilizzando `lv_obj_set_style_transform_rotation(scale, LV_SCALE_LABEL_ROTATE_MATCH_TICKS, LV_PART_INDICATOR)`.

In alternativa, le etichette possono essere ruotate di un valore fisso (per qualsiasi modalità della "Scale"). Questo esempio ruota le etichette di 20 gradi: `lv_obj_set_style_transform_rotation(scale, 200, LV_PART_INDICATOR)`.

Oppure è possibile eseguire entrambe le operazioni contemporaneamente: `lv_obj_set_style_transform_rotation(scale, LV_SCALE_LABEL_ROTATE_MATCH_TICKS + 200, LV_PART_INDICATOR)`.

Alcune etichette della "Scale" potrebbero essere disegnate capovolte (per corrispondere alla tacca) se include un certo intervallo di angoli. Se non si desidera che ciò avvenga, è possibile utilizzare un flag aggiuntivo per ruotare automaticamente le etichette e mantenerle in posizione verticale. Le etichette che sarebbero capovolte vengono quindi ruotate di 180° `lv_obj_set_style_transform_rotation(scale, LV_SCALE_LABEL_ROTATE_MATCH_TICKS | LV_SCALE_LABEL_ROTATE_KEEP_UPRIGHT, LV_PART_INDICATOR)`. Le etichette possono anche essere spostate di una distanza fissa in pixel X e Y utilizzando `lv_obj_set_style_translate_x(scale, 10, LV_PART_INDICATOR)`.

Nota

Il valore della tacca principale viene calcolato con l'API `lv_map` (quando non si impostano etichette personalizzate); questo calcolo tiene conto del numero totale di tacche e dell'intervallo della "Scale", quindi l'etichetta disegnata può presentare errori di arrotondamento quando il valore calcolato è un valore in virgola mobile.

La lunghezza delle tacche può essere configurata con la proprietà "length Style" su `LV_PART_INDICATOR` per le tacche principali e `LV_PART_ITEMS` per le tacche secondarie. Esempio con Stile locale: `lv_obj_set_style_length(scale, 5, LV_PART_INDICATOR)` per le tacche principali e `lv_obj_set_style_length(scale, 5, LV_PART_ITEMS)` per le tacche secondarie. Le tacche possono essere riempite in entrambe le direzioni (verso l'esterno o verso l'interno) per le Scale `..._ROUND_...` solo con: `lv_obj_set_style_radial_offset(scale, 5, LV_PART_INDICATOR)` per le tacche principali e `lv_obj_set_style_radial_offset(scale, 5, LV_PART_ITEMS)` per quelle secondarie. L'utilizzo combinato di lunghezza e offset radiale consente il controllo totale della posizione della tacca.

È anche possibile compensare le etichette dai segni principali (positivi o negativi) utilizzando `lv_obj_set_style_pad_radial(scale, 5, LV_PART_INDICATOR)`

Sezioni

Le Sezioni consentono a porzioni di una Scala di *trasmettere significato* utilizzando diverse proprietà di Stile per disegnarle (colori, spessori delle linee, font, ecc.).

Una Sezione rappresenta un sottointervallo della "Scale", i cui Stili (come i Cascading Style Sheets) hanno la precedenza durante il disegno delle PARTI (linee, archi, tacche ed etichette) della "Scale" che rientrano nell'intervallo di quella Sezione.

Se una PARTE di una Scala rientra nell'intervallo di 2 o più Sezioni (ovvero se tali Sezioni si sovrappongono), le proprietà dello Stile appartenenti alla Sezione aggiunta più di recente hanno la precedenza sulle stesse proprietà di stile delle altre Sezioni che "coinvolgono" quella PARTE.

Creazione di Sezioni

Una sezione viene creata utilizzando `lv_scale_add_section(scale)`, che restituisce un puntatore a un oggetto `lv_scale_section_t`. Questo crea una Sezione con intervallo [0..0] e senza Stili aggiunti, il che garantisce che la Sezione non venga ancora disegnata: necessita sia di un intervallo all'interno dell'intervallo della Scala che di almeno uno *Stile* aggiunto prima di poter essere utilizzata nel disegno della Scala.

Successivamente, si imposta l'intervallo usando `lv_scale_section_set_range(section, min, max)` dove `min` e `max` sono i valori limite della Sezione che normalmente dovrebbero essere compresi nell'intervallo di valori della "Scale". (Se rientrano solo parzialmente nell'intervallo della "Scale", questa utilizzerà solo la porzione della Sezione che si sovrappone al suo intervallo. Se l'intervallo di una Sezione non rientra affatto nell'intervallo della "Scale", non verrà utilizzato nel disegno. Questo può essere utile per "disattivare" temporaneamente una Sezione, ad esempio `lv_scale_section_set_range(section, 0, -1)`.)

Data binding

Per familiarizzare con "observer", "subject" e data binding in generale, visitare la pagina *Observer*.

Questo metodo di sottoscrizione a un Subject intero influisce direttamente sui valori minimi o massimi interi di una Sezione di un Widget "Scale". Si noti che si tratta di un binding unidirezionale (Subject ==> Widget) poiché i limiti della Sezione "Scale" non sono interattivi.

Supporta solo soggetti interi.

- `lv_scale_bind_section_min_value(scale, section1, &subject)`
- `lv_scale_bind_section_max_value(scale, section1, &subject)`

Stili delle Sezioni

Per impostare le proprietà di Stile di una Sezione, è possibile creare un oggetto `lv_style_t` per ogni "sezione" che si desidera visualizzare in modo diverso rispetto alla Scala padre. Aggiungere le proprietà di stile come documentato in *Fogli di Stile*.

Aggiungere ciascun oggetto `lv_style_t` a ogni Sezione a cui verrà applicato utilizzando `lv_scale_section_set_style(section, PART, style_pointer)`, dove:

- `style_pointer` dovrebbe puntare al contenuto di una variabile globale o statica (può essere allocata dinamicamente), poiché deve rimanere valida per tutta la durata della "Scale"; e
- `PART` indica a quale singola *PARTE* della "Scale" padre verrà applicato, ovvero `LV_PART_MAIN`, `LV_PART_ITEMS` o `LV_PART_INDICATOR`.

A differenza dell'aggiunta di stili normali ai Widget, non è possibile combinare le PARTI eseguendo l'operazione OR bit per bit sui valori delle PARTI per ottenere che lo stile venga applicato a più di una parte. Tuttavia, è possibile fare qualcosa di simile per ottenere lo stesso risultato:

```
static lv_style_t tick_style;
lv_style_init(&tick_style);
lv_style_set_line_color(&tick_style, lv_palette_darken(LV_PALETTE_RED, 3));
lv_scale_section_set_style(section, LV_PART_ITEMS, &tick_style);
lv_scale_section_set_style(section, LV_PART_INDICATOR, &tick_style);
```

per fare in modo che quell'oggetto Style venga applicato sia alle tacche maggiori che a quella minori.

Gli oggetti `lv_style_t` possono essere condivisi tra le Sezioni e tra le PARTI, ma a differenza dei normali Stili aggiunti a un Widget, una Sezione può avere solo 1 stile per PARTE. Quindi, procedendo in questo modo:

```
lv_scale_section_set_style(section, LV_PART_INDICATOR, &tick_style_1);
lv_scale_section_set_style(section, LV_PART_INDICATOR, &tick_style_2);
```

sostituisce `tick_style_1` con `tick_style_2` per la parte `LV_PART_INDICATOR` anziché aggiungerla.

Proprietà di Stile utili per le Sezioni

Le proprietà di Stile utilizzate durante il disegno della "Scale" (e quindi utili) sono:

- Per la linea principale *quando è una linea retta* (`LV_PART_MAIN`):

LV_STYLE_LINE_WIDTH
`lv_style_set_line_width()`

LV_STYLE_LINE_COLOR
`lv_style_set_line_color()`

LV_STYLE_LINE_OPA
`lv_style_set_line_opa()`

- Per la linea principale *quando è un arco* (`LV_PART_MAIN`):

LV_STYLE_ARC_WIDTH
`lv_style_set_arc_width()`

LV_STYLE_ARC_COLOR
`lv_style_set_arc_color()`

LV_STYLE_ARC_OPA
`lv_style_set_arc_opa()`

LV_STYLE_ARC_ROUNDED
`lv_style_set_arc_rounded()`

LV_STYLE_ARC_IMAGE_SRC
`lv_style_set_arc_image_src()`

- Per le linee delle tacche (`LV_PART_ITEMS` e `LV_PART_INDICATOR`):

LV_STYLE_LINE_WIDTH
`lv_style_set_line_width()`

LV_STYLE_LINE_COLOR`lv_style_set_line_color()`**LV_STYLE_LINE_OPA**`lv_style_set_line_opa()`

- Per le etichette sulle tacche principali ([LV_PART_INDICATOR](#))

LV_STYLE_TEXT_COLOR`lv_style_set_text_color()`**LV_STYLE_TEXT_OPA**`lv_style_set_text_opa()`**LV_STYLE_TEXT_LETTER_SPACE**`lv_style_set_text_letter_space()`**LV_STYLE_TEXT_FONT**`lv_style_set_text_font()`

6.26.4 Eventi

I Widget "Scale" non inviano eventi speciali.

Negli eventi [LV_EVENT_DRAW_TASK_ADDED](#), i membri `id1` e `id2` di un descrittore di disegno di linea principale o secondario saranno rispettivamente l'indice della tacca e il suo valore. Se la parte è [LV_PART_INDICATOR](#), si tratta di una tacca principale. Se la parte è [LV_PART_ITEMS](#), si tratta di una tacca secondaria.

i Ulteriori Informazioni

Scoprire di più sugli [Eventi](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

6.26.5 Tasti

Nessun *Tasto* viene elaborato dai Widget "Scale".

i Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

6.26.6 Esempio

Una semplice scala orizzontale

```
#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/***
 * A simple horizontal scale
 ***/
void lv_example_scale_1(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, lv_pct(80), 100);
    lv_scale_set_mode(scale, LV_SCALE_MODE_HORIZONTAL_BOTTOM);
    lv_obj_center(scale);

    lv_scale_set_label_show(scale, true);

    lv_scale_set_total_tick_count(scale, 31);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 10, 40);
}

#endif
```

Una scala verticale con sezioni e stile personalizzato

```
#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/** An vertical scale with section and custom styling */
void lv_exemple_scale_2(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, 60, 200);
    lv_scale_set_label_show(scale, true);
    lv_scale_set_mode(scale, LV_SCALE_MODE_VERTICAL_RIGHT);
    lv_obj_center(scale);

    lv_scale_set_total_tick_count(scale, 21);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_scale_set_range(scale, 0, 100);

    static const char * custom_labels[] = {"0 °C", "25 °C", "50 °C", "75 °C", "100 °C", NULL};
    lv_scale_set_text_src(scale, custom_labels);

    static lv_style_t indicator_style;
    lv_style_init(&indicator_style);

    /* Label style properties */
    lv_style_set_text_font(&indicator_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&indicator_style, lv_palette_darken(LV_PALETTE_BLUE, 3));

    /* Major tick properties */
    lv_style_set_line_color(&indicator_style, lv_palette_darken(LV_PALETTE_BLUE, 3));
    lv_style_set_width(&indicator_style, 10U); /*Tick length*/
    lv_style_set_line_width(&indicator_style, 2U); /*Tick width*/
    lv_obj_add_style(scale, &indicator_style, LV_PART_INDICATOR);

    static lv_style_t minor_ticks_style;
    lv_style_init(&minor_ticks_style);
    lv_style_set_line_color(&minor_ticks_style, lv_palette_lighten(LV_PALETTE_BLUE, 2));
    lv_style_set_width(&minor_ticks_style, 5U); /*Tick length*/
    lv_style_set_line_width(&minor_ticks_style, 2U); /*Tick width*/
    lv_obj_add_style(scale, &minor_ticks_style, LV_PART_ITEMS);

    static lv_style_t main_line_style;
    lv_style_init(&main_line_style);
    /* Main line properties */
    lv_style_set_line_color(&main_line_style, lv_palette_darken(LV_PALETTE_BLUE, 3));
    lv_style_set_line_width(&main_line_style, 2U); // Tick width
    lv_obj_add_style(scale, &main_line_style, LV_PART_MAIN);

    /* Add a section */
    static lv_style_t section_minor_tick_style;
    static lv_style_t section_label_style;
    static lv_style_t section_main_line_style;

    lv_style_init(&section_label_style);
    lv_style_init(&section_minor_tick_style);
    lv_style_init(&section_main_line_style);

    /* Label style properties */
    lv_style_set_text_font(&section_label_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&section_label_style, lv_palette_darken(LV_PALETTE_RED, 3));

    lv_style_set_line_color(&section_label_style, lv_palette_darken(LV_PALETTE_RED, 3));
    lv_style_set_line_width(&section_label_style, 5U); /*Tick width*/

    lv_style_set_line_color(&section_minor_tick_style, lv_palette_lighten(LV_PALETTE_RED, 2));
    lv_style_set_line_width(&section_minor_tick_style, 4U); /*Tick width*/

    /* Main line properties */
    lv_style_set_line_color(&section_main_line_style, lv_palette_darken(LV_PALETTE_RED, 3));
    lv_style_set_line_width(&section_main_line_style, 4U); /*Tick width*/

    /* Configure section styles */
    lv_scale_section_t * section = lv_scale_add_section(scale);
    lv_scale_set_section_range(scale, section, 75, 100);
    lv_scale_set_section_style_indicator(scale, section, &section_label_style);
    lv_scale_set_section_style_items(scale, section, &section_minor_tick_style);
    lv_scale_set_section_style_main(scale, section, &section_main_line_style);

    lv_obj_set_style_bg_color(scale, lv_palette_main(LV_PALETTE_BLUE_GREY), 0);
    lv_obj_set_style_bg_opa(scale, LV_OPA_50, 0);
    lv_obj_set_style_pad_left(scale, 8, 0);
    lv_obj_set_style_radius(scale, 8, 0);
    lv_obj_set_style_pad_ver(scale, 20, 0);
}

#endif
```

Una semplice scala circolare

```
#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

LV_IMAGE_DECLARE(img_hand);

static lv_obj_t * needle_line;
static lv_obj_t * needle_img;

static void set_needle_line_value(void * obj, int32_t v)
{
    lv_scale_set_line_needle_value((lv_obj_t *)obj, needle_line, 60, v);
}

static void set_needle_img_value(void * obj, int32_t v)
```

(continues on next page)

(continua dalla pagina precedente)

```
{
    lv_scale_set_image_needle_value((lv_obj_t *)obj, needle_img, v);
}

/**
 * A simple round scale
 */
void lv_example_scale_3(void)
{
    lv_obj_t * scale_line = lv_scale_create(lv_screen_active());

    lv_obj_set_size(scale_line, 150, 150);
    lv_scale_set_mode(scale_line, LV_SCALE_MODE_ROUND_INNER);
    lv_obj_set_style_bg_opa(scale_line, LV_OPA_COVER, 0);
    lv_obj_set_style_bg_color(scale_line, lv_palette_lighten(LV_PALETTE_RED, 5), 0);
    lv_obj_set_style_radius(scale_line, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_clip_corner(scale_line, true, 0);
    lv_obj_align(scale_line, LV_ALIGN_LEFT_MID, LV_PCT(2), 0);

    lv_scale_set_label_show(scale_line, true);

    lv_scale_set_total_tick_count(scale_line, 31);
    lv_scale_set_major_tick_every(scale_line, 5);

    lv_obj_set_style_length(scale_line, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale_line, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale_line, 10, 40);

    lv_scale_set_angle_range(scale_line, 270);
    lv_scale_set_rotation(scale_line, 135);

    needle_line = lv_line_create(scale_line);

    lv_obj_set_style_line_width(needle_line, 6, LV_PART_MAIN);
    lv_obj_set_style_line_rounded(needle_line, true, LV_PART_MAIN);

    lv_anim_t anim_scale_line;
    lv_anim_init(&anim_scale_line);
    lv_anim_set_var(&anim_scale_line, scale_line);
    lv_anim_set_exec_cb(&anim_scale_line, set_needle_line_value);
    lv_anim_set_duration(&anim_scale_line, 1000);
    lv_anim_set_repeat_count(&anim_scale_line, LV_ANIM_REPEAT_INFINITE);
    lv_anim_set_reverse_duration(&anim_scale_line, 1000);
    lv_anim_set_values(&anim_scale_line, 10, 40);
    lv_anim_start(&anim_scale_line);

    lv_obj_t * scale_img = lv_scale_create(lv_screen_active());

    lv_obj_set_size(scale_img, 150, 150);
    lv_scale_set_mode(scale_img, LV_SCALE_MODE_ROUND_INNER);
    lv_obj_set_style_bg_opa(scale_img, LV_OPA_COVER, 0);
    lv_obj_set_style_bg_color(scale_img, lv_palette_lighten(LV_PALETTE_RED, 5), 0);
    lv_obj_set_style_radius(scale_img, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_clip_corner(scale_img, true, 0);
    lv_obj_align(scale_img, LV_ALIGN_RIGHT_MID, LV_PCT(-2), 0);

    lv_scale_set_label_show(scale_img, true);

    lv_scale_set_total_tick_count(scale_img, 31);
    lv_scale_set_major_tick_every(scale_img, 5);

    lv_obj_set_style_length(scale_img, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale_img, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale_img, 10, 40);

    lv_scale_set_angle_range(scale_img, 270);
    lv_scale_set_rotation(scale_img, 135);

    /* image must point to the right. E.g. -0----->*
    needle_img = lv_image_create(scale_img);
    lv_image_set_src(needle_img, &img_hand);
    lv_obj_align(needle_img, LV_ALIGN_CENTER, 47, -2);
    lv_image_set_pivot(needle_img, 3, 4);

    lv_anim_t anim_scale_img;
    lv_anim_init(&anim_scale_img);
    lv_anim_set_var(&anim_scale_img, scale_img);
    lv_anim_set_exec_cb(&anim_scale_img, set_needle_img_value);
    lv_anim_set_duration(&anim_scale_img, 1000);
    lv_anim_set_repeat_count(&anim_scale_img, LV_ANIM_REPEAT_INFINITE);
    lv_anim_set_reverse_duration(&anim_scale_img, 1000);
    lv_anim_set_values(&anim_scale_img, 10, 40);
    lv_anim_start(&anim_scale_img);
}
#endif
```

Una scala circolare con sezione e stile personalizzato

```
#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/**
 * A round scale with section and custom styling
 */
void lv_example_scale_4(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, 150, 150);
    lv_scale_set_label_show(scale, true);
    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_OUTER);
    lv_obj_center(scale);

    lv_scale_set_total_tick_count(scale, 21);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_scale_set_range(scale, 0, 100);

static const char * custom_labels[] = {"0 °C", "25 °C", "50 °C", "75 °C", "100 °C", NULL};
lv_scale_set_text_src(scale, custom_labels);

static lv_style_t indicator_style;
lv_style_init(&indicator_style);

/* Label style properties */
lv_style_set_text_font(&indicator_style, LV_FONT_DEFAULT);
lv_style_set_text_color(&indicator_style, lv_palette_darken(LV_PALETTE_BLUE, 3));

/* Major tick properties */
lv_style_set_line_color(&indicator_style, lv_palette_darken(LV_PALETTE_BLUE, 3));
lv_style_set_width(&indicator_style, 10U); /*Tick length*/
lv_style_set_line_width(&indicator_style, 20); /*Tick width*/
lv_obj_add_style(scale, &indicator_style, LV_PART_INDICATOR);

static lv_style_t minor_ticks_style;
lv_style_init(&minor_ticks_style);
lv_style_set_line_color(&minor_ticks_style, lv_palette_lighten(LV_PALETTE_BLUE, 2));
lv_style_set_width(&minor_ticks_style, 5U); /*Tick length*/
lv_style_set_line_width(&minor_ticks_style, 20); /*Tick width*/
lv_obj_add_style(scale, &minor_ticks_style, LV_PART_ITEMS);

static lv_style_t main_line_style;
lv_style_init(&main_line_style);
/* Main line properties */
lv_style_set_arc_color(&main_line_style, lv_palette_darken(LV_PALETTE_BLUE, 3));
lv_style_set_arc_width(&main_line_style, 20); /*Tick width*/
lv_obj_add_style(scale, &main_line_style, LV_PART_MAIN);

/* Add a section */
static lv_style_t section_minor_tick_style;
static lv_style_t section_label_style;
static lv_style_t section_main_line_style;

lv_style_init(&section_label_style);
lv_style_init(&section_minor_tick_style);
lv_style_init(&section_main_line_style);

/* Label style properties */
lv_style_set_text_font(&section_label_style, LV_FONT_DEFAULT);
lv_style_set_text_color(&section_label_style, lv_palette_darken(LV_PALETTE_RED, 3));

lv_style_set_line_color(&section_label_style, lv_palette_darken(LV_PALETTE_RED, 3));
lv_style_set_line_width(&section_label_style, 5U); /*Tick width*/

lv_style_set_line_color(&section_minor_tick_style, lv_palette_lighten(LV_PALETTE_RED, 2));
lv_style_set_line_width(&section_minor_tick_style, 40); /*Tick width*/

/* Main line properties */
lv_style_set_arc_color(&section_main_line_style, lv_palette_darken(LV_PALETTE_RED, 3));
lv_style_set_arc_width(&section_main_line_style, 40); /*Tick width*/

/* Configure section styles */
lv_scale_section_t * section = lv_scale_add_section(scale);
lv_scale_set_section_range(section, 75, 100);
lv_scale_set_section_style_indicator(section, &section_label_style);
lv_scale_set_section_style_items(section, &section_minor_tick_style);
lv_scale_set_section_style_main(section, &section_main_line_style);
}

#endif

```

Una scala con sezione e stile personalizzato

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/**
 * An scale with section and custom styling
 */
void lv_example_scale_5(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, lv_display_get_horizontal_resolution(NULL) / 2, lv_display_get_vertical_resolution(NULL) / 2);
    lv_scale_set_label_show(scale, true);

    lv_scale_set_total_tick_count(scale, 10);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 25, 35);

    static const char * custom_labels[3] = {"One", "Two", NULL};
    lv_scale_set_text_src(scale, custom_labels);

    static lv_style_t indicator_style;
    lv_style_init(&indicator_style);
    /* Label style properties */
    lv_style_set_text_font(&indicator_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&indicator_style, lv_color_hex(0xffff00));
    /* Major tick properties */
    lv_style_set_line_color(&indicator_style, lv_color_hex(0x00ff00));
    lv_style_set_width(&indicator_style, 10U); // Tick length
    lv_style_set_line_width(&indicator_style, 20); // Tick width
    lv_obj_add_style(scale, &indicator_style, LV_PART_INDICATOR);

    static lv_style_t minor_ticks_style;
    lv_style_init(&minor_ticks_style);
    lv_style_set_line_color(&minor_ticks_style, lv_color_hex(0xffff00));
    lv_style_set_width(&minor_ticks_style, 5U); // Tick length
    lv_style_set_line_width(&minor_ticks_style, 20); // Tick width
    lv_obj_add_style(scale, &minor_ticks_style, LV_PART_ITEMS);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

static lv_style_t main_line_style;
lv_style_init(&main_line_style);
/* Main line properties */
lv_style_set_line_color(&main_line_style, lv_color_hex(0x0000ff));
lv_style_set_line_width(&main_line_style, 2U); // Tick width
lv_obj_add_style(scale, &main_line_style, LV_PART_MAIN);

lv_obj_center(scale);

/* Add a section */
static lv_style_t section_minor_tick_style;
static lv_style_t section_label_style;

lv_style_init(&section_label_style);
lv_style_init(&section_minor_tick_style);

/* Label style properties */
lv_style_set_text_font(&section_label_style, LV_FONT_DEFAULT);
lv_style_set_text_color(&section_label_style, lv_color_hex(0xffff00));
lv_style_set_text_letter_space(&section_label_style, 10);
lv_style_set_text_opa(&section_label_style, LV_OPA_50);

lv_style_set_line_color(&section_label_style, lv_color_hex(0xff0000));
// lv_style_set_width(&section_label_style, 200); // Tick length
lv_style_set_line_width(&section_label_style, 5U); // Tick width

lv_style_set_line_color(&section_minor_tick_style, lv_color_hex(0x0000ff));
// lv_style_set_width(&section_label_style, 200); // Tick length
lv_style_set_line_width(&section_minor_tick_style, 4U); // Tick width

/* Configure section styles */
lv_scale_section_t * section = lv_scale_add_section(scale);
lv_scale_set_section_range(scale, section, 25, 30);
lv_scale_set_section_style_indicator(scale, section, &section_label_style);
lv_scale_set_section_style_items(scale, section, &section_minor_tick_style);
}

#endif

```

Una scala circolare con più lancette, simile a un orologio

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

#if LV_USE_FLOAT
#define my_PRIPrecise "f"
#else
#define my_PRIPrecise LV_PRId32
#endif

static lv_obj_t * scale;
static lv_obj_t * minute_hand;
static lv_obj_t * hour_hand;
static lv_point_precise_t minute_hand_points[2];
static int32_t hour;
static int32_t minute;

static void timer_cb(lv_timer_t * timer)
{
    LV_UNUSED(timer);

    minute++;
    if(minute > 59) {
        minute = 0;
        hour++;
        if(hour > 11) {
            hour = 0;
        }
    }
}

/**
 * the scale will store the needle line points in the existing
 * point array if one was set with 'lv_line_set_points mutable'.
 * Otherwise, it will allocate the needle line points.
 */

/* the scale will store the minute hand line points in 'minute_hand_points' */
lv_scale_set_line_needle_value(scale, minute_hand, 60, minute);
/* log the points that were stored in the array */
LV_LOG_USER(
    "minute hand points - "
    "0: (%" my_PRIPrecise ", %" my_PRIPrecise "), "
    "1: (%" my_PRIPrecise ", %" my_PRIPrecise ")",
    minute_hand_points[0].x, minute_hand_points[0].y,
    minute_hand_points[1].x, minute_hand_points[1].y
);

/* the scale will allocate the hour hand line points */
lv_scale_set_line_needle_value(scale, hour_hand, 40, hour * 5 + (minute / 12));
}

/**
 * A round scale with multiple needles, resembling a clock
 */
void lv_example_scale_6(void)
{
    scale = lv_scale_create(lv_screen_active());

    lv_obj_set_size(scale, 150, 150);
    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_INNER);
    lv_obj_set_style_bg_opa(scale, LV_OPA_60, 0);
    lv_obj_set_style_bg_color(scale, lv_color_black(), 0);
    lv_obj_set_style_radius(scale, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_clip_corner(scale, true, 0);
    lv_obj_center(scale);

    lv_scale_set_label_show(scale, true);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_scale_set_total_tick_count(scale, 61);
lv_scale_set_major_tick_every(scale, 5);

static const char * hour_ticks[] = {"12", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", NULL};
lv_scale_set_text_src(scale, hour_ticks);

static lv_style_t indicator_style;
lv_style_init(&indicator_style);

/* Label style properties */
lv_style_set_text_font(&indicator_style, LV_FONT_DEFAULT);
lv_style_set_text_color(&indicator_style, lv_palette_main(LV_PALETTE_YELLOW));

/* Major tick properties */
lv_style_set_line_color(&indicator_style, lv_palette_main(LV_PALETTE_YELLOW));
lv_style_set_length(&indicator_style, 8); /* tick length */
lv_style_set_line_width(&indicator_style, 2); /* tick width */
lv_obj_add_style(scale, &indicator_style, LV_PART_INDICATOR);

/* Minor tick properties */
static lv_style_t minor_ticks_style;
lv_style_init(&minor_ticks_style);
lv_style_set_line_color(&minor_ticks_style, lv_palette_main(LV_PALETTE_YELLOW));
lv_style_set_length(&minor_ticks_style, 6); /* tick length */
lv_style_set_line_width(&minor_ticks_style, 2); /* tick width */
lv_obj_add_style(scale, &minor_ticks_style, LV_PART_ITEMS);

/* Main line properties */
static lv_style_t main_line_style;
lv_style_init(&main_line_style);
lv_style_set_arc_color(&main_line_style, lv_color_black());
lv_style_set_arc_width(&main_line_style, 5);
lv_obj_add_style(scale, &main_line_style, LV_PART_MAIN);

lv_scale_set_range(scale, 0, 60);

lv_scale_set_angle_range(scale, 360);
lv_scale_set_rotation(scale, 270);

minute_hand = lv_line_create(scale);
lv_line_set_points_mutable(minute_hand, minute_hand_points, 2);

lv_obj_set_style_line_width(minute_hand, 3, 0);
lv_obj_set_style_line_rounded(minute_hand, true, 0);
lv_obj_set_style_line_color(minute_hand, lv_color_white(), 0);

hour_hand = lv_line_create(scale);

lv_obj_set_style_line_width(hour_hand, 5, 0);
lv_obj_set_style_line_rounded(hour_hand, true, 0);
lv_obj_set_style_line_color(hour_hand, lv_palette_main(LV_PALETTE_RED), 0);

hour = 11;
minute = 5;
lv_timer_t * timer = lv_timer_create(timer_cb, 250, NULL);
lv_timer_ready(timer);
}

#endif

```

Personalizzazione del colore dell'etichetta principale della scala con l'evento LV_EVENT_DRAW_TASK_ADDED

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

#include "../../lvgl_private.h" /*To expose the fields of lv_draw_task_t*/

static void draw_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
    if(base_dsc->part == LV_PART_INDICATOR) {
        if(label_draw_dsc) {
            const lv_color_t color_idx[7] = {
                lv_palette_main(LV_PALETTE_RED),
                lv_palette_main(LV_PALETTE_ORANGE),
                lv_palette_main(LV_PALETTE_YELLOW),
                lv_palette_main(LV_PALETTE_GREEN),
                lv_palette_main(LV_PALETTE_CYAN),
                lv_palette_main(LV_PALETTE_BLUE),
                lv_palette_main(LV_PALETTE_PURPLE),
            };
            uint32_t major_tick = lv_scale_get_major_tick_every(obj);
            label_draw_dsc->color = color_idx[base_dsc->id1 / major_tick];

            /*Free the previously allocated text if needed*/
            if(label_draw_dsc->text_local) lv_free((void *)label_draw_dsc->text);

            /*Malloc the text and set text_local as 1 to make LVGL automatically free the text.
             * (Local texts are malloc'd internally by LVGL. Mimic this behavior here too)*/
            char tmp_buffer[20] = {0}; /* Big enough buffer */
            lv_snprintf(tmp_buffer, sizeof(tmp_buffer), "%.1f", (double)base_dsc->id2);
            label_draw_dsc->text = lv_strdup(tmp_buffer);
            label_draw_dsc->text_local = 1;

            lv_point_t size;
            lv_text_get_size(&size, label_draw_dsc->text, label_draw_dsc->font, 0, 0, 1000, LV_TEXT_FLAG_NONE);
            int32_t new_w = size.x;
            int32_t old_w = lv_area_get_width(&draw_task->area);

            /* Distribute the new size equally on both sides */
            draw_task->area.x1 -= (new_w - old_w) / 2;
            draw_task->area.x2 += ((new_w - old_w) + 1) / 2; /* +1 for rounding */
        }
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

        }
    }

/***
 * Customizing scale major tick label color with `LV_EVENT_DRAW_TASK_ADDED` event
*/
void lv_example_scale_7(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, lv_pct(80), 100);
    lv_scale_set_mode(scale, LV_SCALE_MODE_HORIZONTAL_BOTTOM);
    lv_obj_center(scale);

    lv_scale_set_label_show(scale, true);

    lv_scale_set_total_tick_count(scale, 31);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 10, 40);

    lv_obj_add_event_cb(scale, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(scale, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
}

#endif

```

Una scala circolare con etichette ruotate e traslate

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/***
 * A simple round scale with label/tick translation
*/
void lv_example_scale_8(void)
{
    lv_obj_t * scale_line = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale_line, 150, 150);
    lv_scale_set_mode(scale_line, LV_SCALE_MODE_ROUND_INNER);
    lv_obj_set_style_bg_opa(scale_line, LV_OPA_COVER, 0);
    lv_obj_set_style_bg_color(scale_line, lv_palette_lighten(LV_PALETTE_RED, 5), 0);
    lv_obj_set_style_radius(scale_line, LV_RADIUS_CIRCLE, 0);
    lv_obj_align(scale_line, LV_ALIGN_LEFT_MID, LV_PCT(2), 0);

    /*Set the texts' and major ticks' rotation (make the texts rotated) */
    lv_obj_set_style_transform_rotation(scale_line, LV_SCALE_LABEL_ROTATE_MATCH_TICKS | LV_SCALE_LABEL_ROTATE_KEEP_UPRIGHT,
                                      LV_PART_INDICATOR);
    lv_obj_set_style_translate_x(scale_line, 10, LV_PART_INDICATOR);
    lv_obj_set_style_length(scale_line, 15, LV_PART_INDICATOR);
    lv_obj_set_style_radial_offset(scale_line, 10, LV_PART_INDICATOR);

    /*Set the style of the minor ticks*/
    lv_obj_set_style_length(scale_line, 10, LV_PART_ITEMS);
    lv_obj_set_style_radial_offset(scale_line, 5, LV_PART_ITEMS);
    lv_obj_set_style_line_opa(scale_line, LV_OPA_50, LV_PART_ITEMS);

    lv_scale_set_label_show(scale_line, true);

    lv_scale_set_total_tick_count(scale_line, 31);
    lv_scale_set_major_tick_every(scale_line, 5);

    lv_scale_set_range(scale_line, 10, 40);

    lv_scale_set_angle_range(scale_line, 270);
    lv_scale_set_rotation(scale_line, 135);

    lv_obj_t * needle_line = lv_line_create(scale_line);

    lv_obj_set_style_line_width(needle_line, 3, LV_PART_MAIN);
    lv_obj_set_style_line_rounded(needle_line, true, LV_PART_MAIN);
    lv_scale_set_line_needle_value(scale_line, needle_line, 60, 33);
}

#endif

```

Una scala orizzontale con etichette ruotate e traslate

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/***
 * A simple horizontal scale with transforms
*/
void lv_example_scale_9(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, 200, 100);
    lv_scale_set_mode(scale, LV_SCALE_MODE_HORIZONTAL_BOTTOM);
    lv_obj_center(scale);

    lv_scale_set_label_show(scale, true);
    lv_obj_set_style_transform_rotation(scale, 450, LV_PART_INDICATOR);
    lv_obj_set_style_length(scale, 30, LV_PART_INDICATOR);
    lv_obj_set_style_translate_x(scale, 5, LV_PART_INDICATOR);

    lv_scale_set_total_tick_count(scale, 31);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);

```

(continues on next page)

(continua dalla pagina precedente)

```

    lv_scale_set_range(scale, 10, 40);
}

#endif

```

Uno stile di scala circolare che simula un cardiofrequenzimetro

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES && LV_FONT_MONTserrat_40 && LV_FONT_MONTserrat_18

static int32_t hr_value = 98;
static int8_t hr_step = 1;
static lv_obj_t * needle_line = NULL;
static lv_obj_t * hr_value_label = NULL;
static lv_obj_t * bpm_label = NULL;
static lv_obj_t * scale = NULL;

typedef struct {
    lv_style_t items;
    lv_style_t indicator;
    lv_style_t main;
} section_styles_t;

static section_styles_t zone1_styles;
static section_styles_t zone2_styles;
static section_styles_t zone3_styles;
static section_styles_t zone4_styles;
static section_styles_t zone5_styles;

static lv_color_t get_hr_zone_color(int32_t hr)
{
    if(hr < 117) return lv_palette_main(LV_PALETTE_GREY); /* Zone 1 */
    else if(hr < 135) return lv_palette_main(LV_PALETTE_BLUE); /* Zone 2 */
    else if(hr < 158) return lv_palette_main(LV_PALETTE_GREEN); /* Zone 3 */
    else if(hr < 176) return lv_palette_main(LV_PALETTE_ORANGE); /* Zone 4 */
    else return lv_palette_main(LV_PALETTE_RED); /* Zone 5 */
}

static void hr_anim_timer_cb(lv_timer_t * timer)
{
    LV_UNUSED(timer);

    hr_value += hr_step;

    if(hr_value >= 195) {
        hr_value = 195;
        hr_step = -1;
    }
    else if(hr_value <= 98) {
        hr_value = 98;
        hr_step = 1;
    }

    /* Update needle */
    lv_scale_set_line_needle_value(scale, needle_line, -8, hr_value);

    /* Update HR text */
    lv_label_set_text_fmt(hr_value_label, "%d", hr_value);

    /* Update text color based on zone */
    lv_color_t zone_color = get_hr_zone_color(hr_value);
    lv_obj_set_style_text_color(hr_value_label, zone_color, 0);
    lv_obj_set_style_text_color(bpm_label, zone_color, 0);
}

static void init_section_styles(section_styles_t * styles, lv_color_t color)
{
    lv_style_init(&styles->items);
    lv_style_set_line_color(&styles->items, color);
    lv_style_set_line_width(&styles->items, 0);

    lv_style_init(&styles->indicator);
    lv_style_set_line_color(&styles->indicator, color);
    lv_style_set_line_width(&styles->indicator, 0);

    lv_style_init(&styles->main);
    lv_style_set_arc_color(&styles->main, color);
    lv_style_set_arc_width(&styles->main, 20);
}

static void add_section(lv_obj_t * target_scale,
                      int32_t from,
                      int32_t to,
                      const section_styles_t * styles)
{
    lv_scale_section_t * sec = lv_scale_add_section(target_scale);
    lv_scale_set_section_range(target_scale, sec, from, to);
    lv_scale_set_section_style_items(target_scale, sec, &styles->items);
    lv_scale_set_section_style_indicator(target_scale, sec, &styles->indicator);
    lv_scale_set_section_style_main(target_scale, sec, &styles->main);
}

void lv_example_scale_10(void)
{
    scale = lv_scale_create(lv_screen_active());
    lv_obj_center(scale);
    lv_obj_set_size(scale, 200, 200);

    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_INNER);
    lv_scale_set_range(scale, 98, 195);
    lv_scale_set_total_tick_count(scale, 15);
    lv_scale_set_major_tick_every(scale, 3);
    lv_scale_set_angle_range(scale, 280);
    lv_scale_set_rotation(scale, 130);
    lv_scale_set_label_show(scale, false);

    lv_obj_set_style_length(scale, 6, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_style_arc_width(scale, 0, LV_PART_MAIN);

/* Zone 1: (Grey) */
init_section_styles(&zone1_styles, lv_palette_main(LV_PALETTE_GREY));
add_section(scale, 98, 117, &zone1_styles);

/* Zone 2: (Blue) */
init_section_styles(&zone2_styles, lv_palette_main(LV_PALETTE_BLUE));
add_section(scale, 117, 135, &zone2_styles);

/* Zone 3: (Green) */
init_section_styles(&zone3_styles, lv_palette_main(LV_PALETTE_GREEN));
add_section(scale, 135, 158, &zone3_styles);

/* Zone 4: (Orange) */
init_section_styles(&zone4_styles, lv_palette_main(LV_PALETTE_ORANGE));
add_section(scale, 158, 176, &zone4_styles);

/* Zone 5: (Red) */
init_section_styles(&zone5_styles, lv_palette_main(LV_PALETTE_RED));
add_section(scale, 176, 195, &zone5_styles);

needle_line = lv_line_create(scale);

/* Optional styling */
lv_obj_set_style_line_color(needle_line, lv_color_black(), LV_PART_MAIN);
lv_obj_set_style_line_width(needle_line, 12, LV_PART_MAIN);
lv_obj_set_style_length(needle_line, 20, LV_PART_MAIN);
lv_obj_set_style_line_rounded(needle_line, false, LV_PART_MAIN);
lv_obj_set_style_pad_right(needle_line, 50, LV_PART_MAIN);

int32_t current_hr = 145;

lv_scale_set_line_needle_value(scale, needle_line, 0, current_hr);

lv_obj_t *circle = lv_obj_create(lv_screen_active());
lv_obj_set_size(circle, 130, 130);
lv_obj_center(circle);

lv_obj_set_style_radius(circle, LV_RADIUS_CIRCLE, 0);

lv_obj_set_style_bg_color(circle, lv_obj_get_style_bg_color(lv_screen_active(), LV_PART_MAIN), 0);
lv_obj_set_style_bg_opa(circle, LV_OPA_COVER, 0);
lv_obj_set_style_border_width(circle, 0, LV_PART_MAIN);

lv_obj_t *hr_container = lv_obj_create(circle);
lv_obj_center(hr_container);
lv_obj_set_size(hr_container, lv_pct(100), LV_SIZE_CONTENT);
lv_obj_set_style_bg_opa(hr_container, LV_OPA_TRANS, 0);
lv_obj_set_style_border_width(hr_container, 0, 0);
lv_obj_set_layout(hr_container, LV_LAYOUT_FLEX);
lv_obj_set_flex_flow(hr_container, LV_FLEX_FLOW_COLUMN);
lv_obj_set_style_pad_all(hr_container, 0, LV_PART_MAIN);
lv_obj_set_style_pad_row(hr_container, 0, 0);
lv_obj_set_flex_align(hr_container, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

hr_value_label = lv_label_create(hr_container);
lv_label_set_text_fmt(hr_value_label, "%d", current_hr);
lv_obj_set_style_text_font(hr_value_label, &lv_font_montserrat_40, 0);
lv_obj_set_style_text_align(hr_value_label, LV_TEXT_ALIGN_CENTER, 0);

bpm_label = lv_label_create(hr_container);
lv_label_set_text(bpm_label, "bpm");
lv_obj_set_style_text_font(bpm_label, &lv_font_montserrat_18, 0);
lv_obj_set_style_text_align(bpm_label, LV_TEXT_ALIGN_CENTER, 0);

lv_color_t zone_color = get_hr_zone_color(current_hr);
lv_obj_set_style_text_color(hr_value_label, zone_color, 0);
lv_obj_set_style_text_color(bpm_label, zone_color, 0);

lv_timer_create(hr_anim_timer_cb, 80, NULL);
}

#endif

```

Uno stile di scala circolare che simula un widget tramonto/alba

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES && LV_FONT_MONTSERAT_12 && LV_FONT_MONTSERAT_14 && LV_FONT_MONTSERAT_16 && LV_FONT_MONTSERAT_20

static void label_color_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    if(!draw_task) return;

    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    if(!base_dsc || base_dsc->part != LV_PART_INDICATOR) return;

    lv_draw_label_dsc_t * label_dsc = lv_draw_task_get_label_dsc(draw_task);
    if(!label_dsc || !label_dsc->text) return;

    const char * txt = label_dsc->text;

    if(lv_strcmp(txt, "06") == 0 || lv_strcmp(txt, "12") == 0 || 
       lv_strcmp(txt, "18") == 0 || lv_strcmp(txt, "24") == 0) {
        label_dsc->color = lv_color_white();
    }
    else {
        label_dsc->color = lv_palette_darken(LV_PALETTE_GREY, 1);
    }
}

void lv_example_scale_ll(void)
{
    lv_obj_t * bg = lv_obj_create(lv_screen_active());
    lv_obj_set_size(bg, 210, 210);
    lv_obj_center(bg);
    lv_obj_set_style_radius(bg, LV_RADIUS_CIRCLE, 0);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_style_bg_color(bg, lv_palette_darken(LV_PALETTE_GREY, 4), 0);
lv_obj_set_style_bg_opa(bg, LV_OPA_COVER, 0);
lv_obj_remove_flag(bg, LV_OBJ_FLAG_SCROLLABLE);
lv_obj_set_style_pad_all(bg, 0, LV_PART_MAIN);

lv_obj_t * scale = lv_scale_create(bg);
lv_obj_center(scale);
lv_obj_set_size(scale, 150, 150);
lv_obj_set_style_arc_width(scale, 5, LV_PART_MAIN);

lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_OUTER);
lv_scale_set_range(scale, 0, 24);
lv_scale_set_total_tick_count(scale, 25);
lv_scale_set_major_tick_every(scale, 1);
lv_scale_set_angle_range(scale, 360);
lv_scale_set_rotation(scale, 105);
lv_scale_set_label_show(scale, true);
lv_obj_set_style_text_font(scale, &lv_font_montserrat_12, LV_PART_INDICATOR);
lv_obj_set_style_pad_radial(scale, -6, LV_PART_INDICATOR);

/* Rotate the labels of the ticks */
lv_obj_set_style_transform_rotation(scale, LV_SCALE_LABEL_ROTATE_MATCH_TICKS | LV_SCALE_LABEL_ROTATE_KEEP_UPRIGHT,
                                  LV_PART_INDICATOR);

/* Style for major ticks */
static lv_style_t style_ticks;
lv_style_init(&style_ticks);
lv_style_set_line_color(&style_ticks, lv_palette_darken(LV_PALETTE_GREY, 1));
lv_style_set_line_width(&style_ticks, 2);
lv_style_set_width(&style_ticks, 10);
lv_obj_add_style(scale, &style_ticks, LV_PART_INDICATOR);

/* Style for NIGHT – blue */
static lv_style_t style_night;
lv_style_init(&style_night);
lv_style_set_arc_color(&style_night, lv_palette_main(LV_PALETTE_BLUE));

/* Style for DAY – dark yellow */
static lv_style_t style_day;
lv_style_init(&style_day);
lv_style_set_arc_color(&style_day, lv_palette_darken(LV_PALETTE_YELLOW, 3));

/* NIGHT section */
lv_scale_section_t * section_night1 = lv_scale_add_section(scale);
lv_scale_set_section_range(scale, section_night1, 17, 5);
lv_scale_set_section_style_main(scale, section_night1, &style_night);

/* DAY section */
lv_scale_section_t * section_day = lv_scale_add_section(scale);
lv_scale_set_section_range(scale, section_day, 5, 17);
lv_scale_set_section_style_main(scale, section_day, &style_day);

static const char * hour_labels[] = {
    "01", "02", "03", "04", "05",
    "06", "07", "08", "09", "10",
    "11", "12", "13", "14", "15",
    "16", "17", "18", "19", "20",
    "21", "22", "23", "24",
    NULL
};
lv_scale_set_text_src(scale, hour_labels);

lv_obj_add_flag(scale, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
lv_obj_add_event_cb(scale, label_color_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);

lv_obj_t * today = lv_label_create(bg);
lv_label_set_text(today, "TODAY");
lv_obj_set_style_text_font(today, &lv_font_montserrat_16, 0);
lv_obj_set_style_text_color(today, lv_color_white(), 0);
lv_obj_align(today, LV_ALIGN_TOP_MID, 0, 60);

lv_obj_t * sunrise_lbl = lv_label_create(bg);
lv_label_set_text(sunrise_lbl, "SUNRISE");
lv_obj_set_style_text_font(sunrise_lbl, &lv_font_montserrat_14, 0);
lv_obj_set_style_text_color(sunrise_lbl, lv_palette_main(LV_PALETTE_GREY), 0);
lv_obj_align(sunrise_lbl, LV_ALIGN_LEFT_MID, 37, -10);

lv_obj_t * sunrise_time = lv_label_create(bg);
lv_label_set_text(sunrise_time, "6:43");
lv_obj_set_style_text_font(sunrise_time, &lv_font_montserrat_20, 0);
lv_obj_set_style_text_color(sunrise_time, lv_color_white(), 0);
lv_obj_align_to(sunrise_time, sunrise_lbl, LV_ALIGN_OUT_BOTTOM_MID, 0, 2);

lv_obj_t * sunset_lbl = lv_label_create(bg);
lv_label_set_text(sunset_lbl, "SUNSET");
lv_obj_set_style_text_font(sunset_lbl, &lv_font_montserrat_14, 0);
lv_obj_set_style_text_color(sunset_lbl, lv_palette_main(LV_PALETTE_GREY), 0);
lv_obj_align(sunset_lbl, LV_ALIGN_RIGHT_MID, -37, -10);

lv_obj_t * sunset_time = lv_label_create(bg);
lv_label_set_text(sunset_time, "17:37");
lv_obj_set_style_text_font(sunset_time, &lv_font_montserrat_20, 0);
lv_obj_set_style_text_color(sunset_time, lv_color_white(), 0);
lv_obj_align_to(sunset_time, sunset_lbl, LV_ALIGN_OUT_BOTTOM_MID, 0, 2);

}

#endif

```

Uno stile di scala circolare che simula una bussola

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

static lv_obj_t * scale;
static lv_obj_t * label;

static const char * heading_to_cardinal(int32_t heading)

```

(continues on next page)

(continua dalla pagina precedente)

```

{
    /* Normalize heading to range [0, 360] */
    while(heading < 0) heading += 360;
    while(heading >= 360) heading -= 360;

    if(heading < 23) return "N";
    else if(heading < 68) return "NE";
    else if(heading < 113) return "E";
    else if(heading < 158) return "SE";
    else if(heading < 203) return "S";
    else if(heading < 248) return "SW";
    else if(heading < 293) return "W";
    else if(heading < 338) return "NW";

    return "N";
}

static void set_heading_value(void * obj, int32_t v)
{
    LV_UNUSED(obj);
    lv_scale_set_rotation(scale, 270 + v);
    lv_label_set_text_fmt(label, "%d\n%s", (int)v, heading_to_cardinal(v));
}

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    lv_draw_label_dsc_t * label_dsc = lv_draw_task_get_label_dsc(draw_task);
    lv_draw_line_dsc_t * line_dsc = lv_draw_task_get_line_dsc(draw_task);
    if(base_dsc->part == LV_PART_INDICATOR) {
        if(label_dsc) {
            if(base_dsc->id1 == 0) {
                label_dsc->color = lv_palette_main(LV_PALETTE_RED);
            }
        }
        if(line_dsc) {
            if(base_dsc->id1 == 60) {
                line_dsc->color = lv_palette_main(LV_PALETTE_RED);
            }
        }
    }
}

/***
 * A round scale style simulating a compass
 */
void lv_example_scale_12(void)
{
    scale = lv_scale_create(lv_screen_active());

    lv_obj_set_size(scale, 200, 200);
    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_INNER);
    lv_obj_set_align(scale, LV_ALIGN_CENTER);

    lv_scale_set_total_tick_count(scale, 61);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_obj_set_style_line_width(scale, 3, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 0, 360);

    static const char * custom_labels[] = {"N", "30", "60", "E", "120", "150", "S", "210", "240", "W", "300", "330", NULL};
    lv_scale_set_text_src(scale, custom_labels);

    lv_scale_set_angle_range(scale, 360);
    lv_scale_set_rotation(scale, 270);

    lv_obj_add_event_cb(scale, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(scale, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);

    label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, 100);
    lv_obj_set_align(label, LV_ALIGN_CENTER);
    lv_label_set_text(label, "0\nN");
    lv_obj_set_style_text_align(label, LV_TEXT_ALIGN_CENTER, 0);

    set_heading_value(NULL, 0);

    lv_obj_t * symbol = lv_label_create(scale);
    lv_obj_set_align(symbol, LV_ALIGN_TOP_MID);
    lv_obj_set_y(symbol, 5);
    lv_label_set_text(symbol, LV_SYMBOL_UP);
    lv_obj_set_style_text_align(symbol, LV_TEXT_ALIGN_CENTER, 0);
    lv_obj_set_style_text_color(symbol, lv_palette_main(LV_PALETTE_RED), 0);

    lv_anim_t anim_scale;
    lv_anim_init(&anim_scale);
    lv_anim_set_var(&anim_scale, scale);
    lv_anim_set_exec_cb(&anim_scale, set_heading_value);
    lv_anim_set_duration(&anim_scale, 5000);
    lv_anim_set_repeat_delay(&anim_scale, 500);
    lv_anim_set_repeat_count(&anim_scale, LV_ANIM_REPEAT_INFINITE);
    lv_anim_set_reverse_duration(&anim_scale, 5000);
    lv_anim_set_reverse_delay(&anim_scale, 500);
    lv_anim_set_values(&anim_scale, 0, 360);
    lv_anim_start(&anim_scale);
}

#endif

```

Segni di spunta e etichette degli assi con scorrimento su un grafico

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

/***
 * Use lv_scale to add ticks to a scrollable chart
 */
void lv_example_chart_2(void)
{
    /*Create a container*/
    lv_obj_t * main_cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(main_cont, 200, 150);
    lv_obj_center(main_cont);

    /*Create a transparent wrapper for the chart and the scale.
     *Set a large width, to make it scrollable on the main container*/
    lv_obj_t * wrapper = lv_obj_create(main_cont);
    lv_obj_remove_style_all(wrapper);
    lv_obj_set_size(wrapper, lv_pct(300), lv_pct(100));
    lv_obj_set_flex_flow(wrapper, LV_FLEX_FLOW_COLUMN);

    /*Create a chart on the wrapper
     *Set it's width to 100% to fill the large wrapper*/
    lv_obj_t * chart = lv_chart_create(wrapper);
    lv_obj_set_width(chart, lv_pct(100));
    lv_obj_set_flex_grow(chart, 1);
    lv_chart_set_type(chart, LV_CHART_TYPE_BAR);
    lv_chart_set_axis_range(chart, LV_CHART_AXIS_PRIMARY_Y, 0, 100);
    lv_chart_set_axis_range(chart, LV_CHART_AXIS_SECONDARY_Y, 0, 400);
    lv_chart_set_point_count(chart, 12);
    lv_obj_set_style_radius(chart, 0, 0);

    /*Create a scale also with 100% width/
    lv_obj_t * scale_bottom = lv_scale_create(wrapper);
    lv_scale_set_mode(scale_bottom, LV_SCALE_MODE_HORIZONTAL_BOTTOM);
    lv_obj_set_size(scale_bottom, lv_pct(100), 25);
    lv_scale_set_total_tick_count(scale_bottom, 12);
    lv_scale_set_major_tick_every(scale_bottom, 1);
    lv_obj_set_style_pad_hor(scale_bottom, lv_chart_get_first_point_center_offset(chart), 0);

    static const char * month[] = {"Jan", "Feb", "March", "Apr", "May", "Jun", "July", "Aug", "Sept", "Oct", "Nov", "Dec", NULL};
    lv_scale_set_text_src(scale_bottom, month);

    /*Add two data series*/
    lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_lighten(LV_PALETTE_GREEN, 2), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_darken(LV_PALETTE_GREEN, 2), LV_CHART_AXIS_PRIMARY_Y);

    /*Set the next points on 'ser1'*/
    uint32_t i;
    for(i = 0; i < 12; i++) {
        lv_chart_set_next_value(chart, ser1, (int32_t)lv_rand(10, 60));
        lv_chart_set_next_value(chart, ser2, (int32_t)lv_rand(50, 90));
    }
    lv_chart_refresh(chart); /*Required after direct set*/
}
#endif
```

6.26.7 API

Scale (lv_scale) .. Autogenerato

lv_scale.h

lv_scale_private.h

6.27 Slider (lv_slider)

6.27.1 Panoramica

Il Widget Slider ha l'aspetto di una *Bar (lv_bar)* con una manopola. La manopola può essere trascinata per impostare il valore dello Slider. Come la Barra, uno Slider può essere verticale o orizzontale.

6.27.2 Parti e Stili

- *LV_PART_MAIN* Il background dello Slider. Utilizza le *tipiche proprietà di stile del background*. padding rende l'indicatore più piccolo nella rispettiva direzione.
- *LV_PART_INDICATOR* L'indicatore che mostra lo stato corrente dello Slider; utilizza anch'esso le *tipiche proprietà di stile del background*.
- *LV_PART_KNOB* Un rettangolo (o cerchio) disegnato in corrispondenza del valore corrente; utilizza anch'esso le *tipiche proprietà di stile del background* per descrivere le manopole. Per default, la manopola è rotonda (lo stile del raggio può modificarlo) con una lunghezza del lato pari alla dimensione minore dello Slider. La manopola può essere ingrandita con i valori di padding. I valori di padding possono anche essere asimmetrici.

6.27.3 Utilizzo

Valore, intervallo e orientamento

Una volta creato, uno Slider ha:

- valore == 0
- intervallo di default di [0..100],
- orientamento orizzontale, con
- larghezza di default di circa 2 pollici (in base al valore configurato di `LV_DPI_DEF`),
- altezza di default di circa 1/10 di pollice (in base al valore configurato di `LV_DPI_DEF`).

Per impostare valori diversi, utilizzare:

- `lv_slider_set_value(slider, new_value, LV_ANIM_ON / OFF)` (il tempo di animazione è impostato dalla proprietà `anim_time` degli stili);
- `lv_slider_set_range(slider, min, max);` e
- per orientamento, larghezza e altezza, è sufficiente impostare le proprietà `width` e `height`;
- `lv_slider_set_orientation(slider, orientation)` per sovrascrivere l'orientamento determinato da `width` e `height`. I valori validi per `orientation` sono:
 - `LV_SLIDER_ORIENTATION_AUTO`
 - `LV_SLIDER_ORIENTATION_HORIZONTAL`
 - `LV_SLIDER_ORIENTATION_VERTICAL`

La direzione di disegno di default è da sinistra a destra nell'orientamento orizzontale e dal basso verso l'alto nell'orientamento verticale. Se il valore minimo è impostato su un valore maggiore del valore massimo (ad esempio [100..0]), la direzione di disegno viene invertita.

Modalità

Lo Slider può essere in una delle seguenti modalità:

- `LV_SLIDER_MODE_NORMAL` Uno Slider normale come descritto sopra (il default)
- `LV_SLIDER_SYMMETRICAL` Disegna l'indicatore dal valore zero al valore corrente. Richiede un intervallo minimo negativo e un intervallo massimo positivo.
- `LV_SLIDER_RANGE` Consente anche di impostare il valore iniziale tramite `lv_bar_set_start_value(slider, new_value, LV_ANIM_ON / OFF)`. Il valore iniziale deve essere sempre inferiore al valore finale.

La modalità può essere modificata con `lv_slider_set_mode(slider, LV_SLIDER_MODE_...)`

Modalità solo manopola

Normalmente, lo Slider può essere regolato trascinando la manopola o cliccando sulla barra dello Slider. In quest'ultimo caso, la manopola si sposta nel punto cliccato e il valore dello Slider cambia di conseguenza. In alcuni casi è preferibile impostare lo Slider in modo che reagisca solo al trascinamento della manopola. Questa funzione si attiva aggiungendo il flag `LV_OBJ_FLAG_ADV_HITTEST`: `lv_obj_add_flag(slider, LV_OBJ_FLAG_ADV_HITTEST)`.

Qualsiasi area di clic estesa (impostata da `lv_obj_set_ext_click_area(slider, value)`) aumenta l'area di clic della manopola.

Data binding

Per familiarizzare con "observer", "subject" e data binding in generale, visitare la pagina *Observer*.

Questo metodo di sottoscrizione a un Subject intero influisce direttamente sul valore intero di un Widget Slider. Si noti che si tratta di un binding bidirezionale (Subject <==> Widget), quindi l'interazione diretta di un utente finale con il Widget Slider aggiorna il valore del Subject e viceversa.

Supporta Subject interi e float.

- `lv_slider_bind_value(slider, &subject)`

6.27.4 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato mentre lo Slider viene trascinato o modificato con i tasti. L'evento viene inviato continuamente mentre lo Slider viene trascinato
- `LV_EVENT_RELEASED` Inviato una volta quando lo Slider viene rilasciato.

Ulteriori Informazioni

Eventi della Bar.

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.27.5 Tasti

- `LV_KEY_UP/RIGHT` Incrementa il valore dello Slider di 1.
- `LV_KEY_DOWN/LEFT` Decrementa il valore dello Slider di 1.

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.27.6 Esempio

Slider semplice

```
#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

static void slider_event_cb(lv_event_t * e);
static lv_obj_t * slider_label;

/**
 * A default slider with a label displaying the current value
 */
void lv_example_slider_1(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_obj_center(slider);
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);

    lv_obj_set_style_anim_duration(slider, 2000, 0);
    /*Create a label below the slider*/
    slider_label = lv_label_create(lv_screen_active());
    lv_label_set_text(slider_label, "0%");
    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
}

static void slider_event_cb(lv_event_t * e)
{
    lv_obj_t * slider = lv_event_get_target_obj(e);
    char buf[8];
    lv_snprintf(buf, sizeof(buf), "%d%", (int)lv_slider_get_value(slider));
    lv_label_set_text(slider_label, buf);
    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
}

#endif
```

Slider con stile personalizzato

```
#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

/**
 * Show how to style a slider.
 */
void lv_example_slider_2(void)
{
    /*Create a transition*/
```

(continues on next page)

(continua dalla pagina precedente)

```

static const lv_style_prop_t props[] = {LV_STYLE_BG_COLOR, 0};
static lv_style_transition_dsc_t transition_dsc;
lv_style_transition_dsc_init(&transition_dsc, props, lv_anim_path_linear, 300, 0, NULL);

static lv_style_t style_main;
static lv_style_t style_indicator;
static lv_style_t style_knob;
static lv_style_t style_pressed_color;
lv_style_init(&style_main);
lv_style_set_bg_opa(&style_main, LV_OPA_COVER);
lv_style_set_bg_color(&style_main, lv_color_hex3(0xbbb));
lv_style_set_radius(&style_main, LV_RADIUS_CIRCLE);
lv_style_set_pad_ver(&style_main, -2); /*Makes the indicator larger*/

lv_style_init(&style_indicator);
lv_style_set_bg_opa(&style_indicator, LV_OPA_COVER);
lv_style_set_bg_color(&style_indicator, lv_palette_main(LV_PALETTE_CYAN));
lv_style_set_radius(&style_indicator, LV_RADIUS_CIRCLE);
lv_style_set_transition(&style_indicator, &transition_dsc);

lv_style_init(&style_knob);
lv_style_set_bg_opa(&style_knob, LV_OPA_COVER);
lv_style_set_bg_color(&style_knob, lv_palette_main(LV_PALETTE_CYAN));
lv_style_set_border_color(&style_knob, lv_palette_darken(LV_PALETTE_CYAN, 3));
lv_style_set_border_width(&style_knob, 2);
lv_style_set_radius(&style_knob, LV_RADIUS_CIRCLE);
lv_style_set_pad_all(&style_knob, 6); /*Makes the knob larger*/
lv_style_set_transition(&style_knob, &transition_dsc);

lv_style_init(&style_pressed_color);
lv_style_set_bg_color(&style_pressed_color, lv_palette_darken(LV_PALETTE_CYAN, 2));

/*Create a slider and add the style*/
lv_obj_t * slider = lv_slider_create(lv_screen_active());
lv_obj_remove_style_all(slider); /*Remove the styles coming from the theme*/

lv_obj_add_style(slider, &style_main, LV_PART_MAIN);
lv_obj_add_style(slider, &style_indicator, LV_PART_INDICATOR);
lv_obj_add_style(slider, &style_pressed_color, LV_PART_INDICATOR | LV_STATE_PRESSED);
lv_obj_add_style(slider, &style_knob, LV_PART_KNOB);
lv_obj_add_style(slider, &style_pressed_color, LV_PART_KNOB | LV_STATE_PRESSED);

lv_obj_center(slider);
}

#endif

```

Slider estensibile

```

#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

#define MAX_VALUE 100
#define MIN_VALUE 0

static void slider_event_cb(lv_event_t * e);

/**
 * Show the current value when the slider is pressed by extending the drawer
 */
void lv_example_slider_3(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider;
    slider = lv_slider_create(lv_screen_active());
    lv_obj_center(slider);

    lv_slider_set_mode(slider, LV_SLIDER_MODE_RANGE);
    lv_slider_set_range(slider, MIN_VALUE, MAX_VALUE);
    lv_slider_set_value(slider, 70, LV_ANIM_OFF);
    lv_slider_set_start_value(slider, 20, LV_ANIM_OFF);

    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_ALL, NULL);
    lv_obj_refresh_ext_draw_size(slider);
}

static void slider_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);

    /*Provide some extra space for the value*/
    if(code == LV_EVENT_REFR_EXT_DRAW_SIZE) {
        lv_event_set_ext_draw_size(e, 50);
    }
    else if(code == LV_EVENT_DRAW_MAIN_END) {
        if(!lv_obj_has_state(obj, LV_STATE_PRESSED)) return;

        lv_area_t slider_area;
        lv_obj_get_coords(obj, &slider_area);
        lv_area_t indic_area = slider_area;
        lv_area_set_width(&indic_area, lv_area_get_width(&slider_area) * lv_slider_get_value(obj) / MAX_VALUE);
        indic_area.x1 += lv_area_get_width(&slider_area) * lv_slider_get_left_value(obj) / MAX_VALUE;
        char buf[16];
        lv_snprintf(buf, sizeof(buf), "%d - %d", (int)lv_slider_get_left_value(obj), (int)lv_slider_get_value(obj));

        lv_point_t label_size;
        lv_text_get_size(&label_size, buf, LV_FONT_DEFAULT, 0, 0, LV_COORD_MAX, LV_TEXT_FLAG_NONE);
        lv_area_t label_area;
        label_area.x1 = 0;
        label_area.x2 = label_size.x - 1;
        label_area.y1 = 0;
        label_area.y2 = label_size.y - 1;

        lv_area_align(&indic_area, &label_area, LV_ALIGN_OUT_TOP_MID, 0, -10);
        lv_draw_label_dsc_t label_draw_dsc;

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_draw_label_dsc_init(&label_draw_dsc);
label_draw_dsc.color = lv_color_hex3(0x888);
label_draw_dsc.text_local = true;
lv_layer_t * layer = lv_event_get_layer(e);
lv_draw_label(layer, &label_draw_dsc, &label_area);
}
#endif

```

Slider con direzione opposta

```

#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

static void slider_event_cb(lv_event_t * e);
static lv_obj_t * slider_label;

/***
 * Slider with opposite direction
 */
void lv_example_slider_4(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_obj_center(slider);
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    /*Reverse the direction of the slider*/
    lv_slider_set_range(slider, 100, 0);
    /*Create a label below the slider*/
    slider_label = lv_label_create(lv_screen_active());
    lv_label_set_text(slider_label, "0%");
    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);

    static void slider_event_cb(lv_event_t * e)
    {
        lv_obj_t * slider = lv_event_get_target_obj(e);
        char buf[8];
        lv_snprintf(buf, sizeof(buf), "%d%", (int)lv_slider_get_value(slider));
        lv_label_set_text(slider_label, buf);
        lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
    }
}

#endif

```

6.27.7 API

lv_api_map_v9_1.h

lv_obj_property_names.h

lv_slider.h

lv_slider_private.h

6.28 Spangroup (lv_spangroup)

6.28.1 Panoramica

Il Widget Spangroup viene utilizzato per visualizzare testo formattato. A differenza del Widget Label, gli Spangroup possono visualizzare testo con stili diversi, con colori e dimensioni diversi. Vedere esempio di seguito.

Uno Spangroup contiene 0 o più Descrittori Span. Ogni Span contiene il proprio testo e le proprie proprietà di stile. Si aggiunge 1 Span (come elemento figlio) allo Spangroup per ogni "span" di testo con stile univoco necessario. Ogni Span aggiunto viene posto alla fine dell'elenco. La sequenza dell'elenco determina l'ordine in cui gli Span vengono visualizzati. Gli Span possono essere aggiunti e rimossi dallo Spangroup per tutta la sua durata. Il numero di Span che possono essere aggiunti è limitato solo dalla RAM disponibile.

6.28.2 Parti e Stili

- *LV_PART_MAIN* Lo Spangroup ha una sola parte.

6.28.3 Utilizzo

Impostare testo e stile

Ogni Span necessario a uno Spangroup si aggiunge in questo modo:

```
lv_span_t * span = lv_spangroup_new_span(spangroup);
```

Dopo aver creato uno Span, si usano le seguenti funzioni per impostarne le proprietà di testo e stile:

- `lv_span_set_text(span, "text")`
- `lv_style_set_<property_name>(&span->style, value)`

Esempio di quest'ultimo: `lv_style_set_text_color(&span->style, lv_palette_main(LV_PALETTE_RED))`.

Se `mode != LV_SPAN_MODE_FIXED` del Widget Spangroup, si chiama `lv_spangroup_refr_mode(spangroup)` dopo aver modificato uno qualsiasi dei suoi Span per assicurarsi che venga ridisegnato correttamente.

Recupero di uno Span figlio

Gli Spangroup memorizzano i propri elementi figlio in modo diverso dai normali Widget, quindi le normali funzioni per ottenere elementi figlio non funzioneranno.

`lv_spangroup_get_child(spangroup, id)` restituirà un puntatore all'elemento figlio Span all'indice `id`. Inoltre, `id` può essere negativo per indicizzare dalla fine dello Spangroup, dove -1 è l'elemento figlio più giovane, -2 è il secondo più giovane, ecc.

Ad esempio `lv_span_t * span = lv_spangroup_get_child(spangroup, 0)` restituirà il primo elemento figlio dello Spangroup. `lv_span_t * span = lv_spangroup_get_child(spangroup, -1)` restituirà l'ultimo elemento figlio (o il più recente).

Conteggio dei figli

Su usa `lv_spangroup_get_span_count(spangroup)` per ottenere il numero di Span contenuti.

Ad esempio `uint32_t size = lv_spangroup_get_span_count(spangroup)`

Rimozione di uno Span

È possibile rimuovere uno Span in qualsiasi momento durante la vita dello Spangroup utilizzando la funzione `lv_spangroup_delete_span(spangroup, span)`.

Allineamento del testo

Come il Widget Label, uno Spangroup può essere impostato su una delle seguenti modalità di allineamento del testo:

- `LV_TEXT_ALIGN_LEFT` Allinea il testo a sinistra.
- `LV_TEXT_ALIGN_CENTER` Centra il testo.
- `LV_TEXT_ALIGN_RIGHT` Allinea il testo al bordo destro.
- `LV_TEXT_ALIGN_AUTO` Allinea automaticamente.

Usare la funzione `lv_spangroup_set_align(spangroup, LV_TEXT_ALIGN_...)` per impostare l'allineamento del testo.

Modalità

DEPRECATO, impostare la larghezza a `LV_SIZE_CONTENT` o su un valore fisso per controllare l'espansione/wrapping.

Uno Spangroup può essere impostato su una delle seguenti modalità:

- `LV_SPAN_MODE_FIXED` Ne fissa le dimensioni.
- `LV_SPAN_MODE_EXPAND` Espande le dimensioni fino a raggiungere quelle del testo, ma rimane su una sola riga.

- `LV_SPAN_MODE_BREAK` Mantiene la larghezza; interrompe le righe troppo lunghe ed espande automaticamente l'altezza.

Usare `lv_spangroup_set_mode(spangroup, LV_SPAN_MODE_BREAK)` per impostarne la modalità.

Overflow

Uno Spangroup può essere impostato per gestire l'overflow del testo in uno dei seguenti modi:

- `LV_SPAN_OVERFLOW_CLIP` tronca il testo al limite dell'area.
- `LV_SPAN_OVERFLOW_ELLIPSIS` visualizza un'ellissi (...) quando il testo supera l'area.

Si usa `lv_spangroup_set_overflow(spangroup, LV_SPAN_OVERFLOW_CLIP)` per impostare la modalità di overflow dello Spangroup.

Rientro prima riga

Usare `lv_spangroup_set_indent(spangroup, 20)` per impostare il rientro della prima riga. Tutte le modalità supportano le unità pixel. Inoltre, le modalità `LV_SPAN_MODE_FIXED` e `LV_SPAN_MODE_BREAK` supportano anche le unità percentuali (ad esempio `lv_pct(10)`).

Righe

Usare `lv_spangroup_set_max_lines(spangroup, 10)` per impostare il numero massimo di righe da visualizzare in modalità `LV_SPAN_MODE_BREAK`. Un valore negativo indica nessun limite.

Data binding

Per familiarizzare con "observer", "subject" e data binding in generale, visitare la pagina *Observer*.

Molto simile al binding Label-text, anche il testo di uno Span può essere associato a un oggetto.

L'unica differenza è che nella funzione di binding devono essere specificati sia lo Spangroup che lo Span:

```
lv_spangroup_bind_span_text(spangroup, span1, &subject, format_string)
```

Si noti che prima di chiamare `lv_spangroup_delete_span`, è necessario chiamare manualmente `lv_observer_remove` poiché LVGL non può rimuovere automaticamente il binding.

6.28.4 Eventi

I Widget Span non inviano eventi speciali.

Ulteriori Informazioni

Scoprire di più sugli [Eventi](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

6.28.5 Tasti

Nessun *Tasto* viene elaborato dai Widget Span.

Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

6.28.6 Esempio

Span con stili personalizzati

```
#include "../../lv_examples.h"
#if LV_USE_SPAN && LV_BUILD_EXAMPLES

static void click_event_cb(lv_event_t * e)
{
    lv_indev_t * indev = lv_event_get_indev(e);
    lv_point_t point;
    lv_indev_get_point(indev, &point);
#if LV_USE_LOG
    lv_obj_t * spans = lv_event_get_target_obj(e);
    lv_span_t * span = lv_spangroup_get_span_by_point(spans, &point);
    LV_LOG_USER("%s", span ? lv_span_get_text(span) : "NULL");
#endif
}

/***
 * Create spans and get clicked one
 */
void lv_example_span_1(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_border_width(&style, 1);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_ORANGE));
    lv_style_set_pad_all(&style, 2);

    lv_obj_t * spans = lv_spangroup_create(lv_screen_active());
    /* Setting a fixed width and height to LV_SIZE_CONTENT will make the text wrap */
    lv_obj_set_width(spans, 300);
    lv_obj_set_height(spans, LV_SIZE_CONTENT);
    lv_obj_center(spans);
    lv_obj_add_style(spans, &style, 0);
    lv_obj_add_flag(spans, LV_OBJ_FLAG_CLICKABLE);

    lv_spangroup_set_align(spans, LV_TEXT_ALIGN_LEFT);
    lv_spangroup_set_overflow(spans, LV_SPAN_OVERFLOW_CLIP);
    lv_spangroup_set_indent(spans, 20);

    lv_span_t * span = lv_spangroup_add_span(spans);
    lv_span_set_text(span, "China is a beautiful country.");
    lv_style_set_text_color(lv_span_get_style(span), lv_palette_main(LV_PALETTE_RED));
    lv_style_set_text_decor(lv_span_get_style(span), LV_TEXT_DECOR_UNDERLINE);
    lv_style_set_text_opa(lv_span_get_style(span), LV_OPA_50);

    span = lv_spangroup_add_span(spans);
    lv_span_set_text_static(span, "good good study, day day up.");
#if LV_FONT_MONTSEERRAT_24
    lv_style_set_text_font(lv_span_get_style(span), &lv_font_montserrat_24);
#endif
    lv_style_set_text_color(lv_span_get_style(span), lv_palette_main(LV_PALETTE_GREEN));

    span = lv_spangroup_add_span(spans);
    lv_span_set_text_static(span, "LVGL is an open-source graphics library.");
    lv_style_set_text_color(lv_span_get_style(span), lv_palette_main(LV_PALETTE_BLUE));

    span = lv_spangroup_add_span(spans);
    lv_span_set_text_static(span, "the boy no name.");
    lv_style_set_text_color(lv_span_get_style(span), lv_palette_main(LV_PALETTE_GREEN));
#if LV_FONT_MONTSEERRAT_20
    lv_style_set_text_font(lv_span_get_style(span), &lv_font_montserrat_20);
#endif
    lv_style_set_text_decor(lv_span_get_style(span), LV_TEXT_DECOR_UNDERLINE);

    span = lv_spangroup_add_span(spans);
    lv_span_set_text(span, "I have a dream that hope to come true.");
    lv_style_set_text_decor(lv_span_get_style(span), LV_TEXT_DECOR_STRIKETHROUGH);

    lv_spangroup_refresh(spans);

    lv_obj_add_event_cb(spans, click_event_cb, LV_EVENT_CLICKED, NULL);
}

#endif
}
```

6.28.7 API

lv_api_map_v9_1.h

lv_span.h

lv_span_private.h

6.29 Spinbox (lv_spinbox)

6.29.1 Panoramica

Spinbox contiene un numero intero visualizzato come numero decimale con una possibile posizione fissa del punto decimale e un numero di cifre configurabile. Il valore può essere aumentato o diminuito tramite *Tasti* o funzioni API. Spinbox è una *Text Area* (*lv_textarea*) con comportamenti estesi per consentire la visualizzazione e la modifica di un valore numerico con vincoli configurabili.

6.29.2 Parti e Stili

Le parti di Spinbox sono identiche a quelle della [Text Area](#).

Valore, intervallo e passo

- `lv_spinbox_set_value(spinbox, 1234)` imposta un nuovo valore per lo Spinbox.
- `lv_spinbox_increment(spinbox)` e `lv_spinbox_decrement(spinbox)` incrementano/decrementano il valore dello Spinbox in base alla cifra attualmente selezionata.
- `lv_spinbox_set_range(spinbox, -1000, 2500)` ne imposta l'intervallo. Se il valore viene modificato da `lv_spinbox_set_value(spinbox)`, dai *Tasti*, da `lv_spinbox_increment(spinbox)` o da `lv_spinbox_decrement(spinbox)` questo intervallo verrà rispettato.
- `lv_spinbox_set_step(spinbox, 100)` imposta quale cifra modificare durante l'incremento/decremento. È possibile impostare solo multipli di dieci.
- `lv_spinbox_set_cursor_pos(spinbox, 1)` imposta il cursore su una cifra specifica da modificare in caso di incremento/decremento. La posizione '0' imposta il cursore sulla cifra meno significativa.

Se si utilizza un encoder come dispositivo di input, la cifra selezionata viene spostata a destra per default ogni volta che si clicca sul pulsante dell'encoder. Per modificare questo comportamento e spostarla a sinistra, è possibile utilizzare `lv_spinbox_set_digit_step_direction(spinbox, LV_DIR_LEFT)`.

Formato

`lv_spinbox_set_digit_format(spinbox, digit_count, separator_position)` imposta il formato del numero. `digit_count` è il numero totale di cifre da visualizzare. `separator_position` è il numero di cifre iniziali prima della virgola. Passare 0 per `separator_position` per non visualizzare alcun separatore decimale.

Rollover

`lv_spinbox_set_rollover(spinbox, true / false)` abilita/disabilita la modalità rollover. Se viene raggiunto il valore minimo o massimo con il rollover abilitato e l'utente tenta di continuare a modificare il valore nella stessa direzione, il valore verrà modificato fino all'altro limite. Se il rollover è disabilitato, il valore si fermerà al valore minimo o massimo.

6.29.3 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato quando il valore è cambiato.

Ulteriori Informazioni

[Eventi della Textarea](#).

Scoprire di più sugli [Eventi](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

Data binding

Per familiarizzare con "observer", "subject" e data binding in generale, visitare la pagina [Observer](#).

Questo metodo di sottoscrizione a un Subject intero influisce direttamente sul valore intero di un Widget Subject. Si noti che si tratta di un'associazione bidirezionale (Subject <==> Widget), quindi l'interazione diretta di un utente finale con il Widget Spinbox aggiorna il valore del Subject e viceversa.

Supporta i Subject interi.

- `lv_spinbox_bind_value(spinbox, &subject)`

6.29.4 Tasti

- LV_KEY_LEFT/RIGHT Con il *tastierino numerico* sposta il cursore a sinistra/destra. Con L'*Encoder* decrementa/incrementa la cifra selezionata.
- LV_KEY_UP/DOWN Con *tastierino numerico* e *Encoder* incrementa/decrementa il valore.
- LV_KEY_ENTER con l'*Encoder*, sposta il focus sulla cifra successiva. Se il focus è sull'ultima cifra, il focus si sposta sulla prima.

Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

6.29.5 Esempio

Spinbox semplice

```
#include "../../lv_examples.h"
#if LV_USE_SPINBOX && LV_BUILD_EXAMPLES

static lv_obj_t * spinbox;

static void lv_spinbox_increment_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_SHORT_CLICKED || code == LV_EVENT_LONG_PRESSED_REPEAT) {
        lv_spinbox_increment(spinbox);
    }
}

static void lv_spinbox_decrement_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_SHORT_CLICKED || code == LV_EVENT_LONG_PRESSED_REPEAT) {
        lv_spinbox_decrement(spinbox);
    }
}

void lv_example_spinbox_1(void)
{
    spinbox = lv_spinbox_create(lv_screen_active());
    lv_spinbox_set_range(spinbox, -1000, 25000);
    lv_spinbox_set_digit_count(spinbox, 5);
    lv_spinbox_set_dec_point_pos(spinbox, 2);
    lv_spinbox_set_step_prev(spinbox);
    lv_obj_set_width(spinbox, 100);
    lv_obj_center(spinbox);

    int32_t h = lv_obj_get_height(spinbox);

    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, h, h);
    lv_obj_align_to(btn, spinbox, LV_ALIGN_OUT_RIGHT_MID, 5, 0);
    lv_obj_set_style_bg_image_src(btn, LV_SYMBOL_PLUS, 0);
    lv_obj_add_event_cb(btn, lv_spinbox_increment_event_cb, LV_EVENT_ALL, NULL);

    btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, h, h);
    lv_obj_align_to(btn, spinbox, LV_ALIGN_OUT_LEFT_MID, -5, 0);
    lv_obj_set_style_bg_image_src(btn, LV_SYMBOL_MINUS, 0);
    lv_obj_add_event_cb(btn, lv_spinbox_decrement_event_cb, LV_EVENT_ALL, NULL);
}

#endif
```

6.29.6 API

[lv_spinbox.h](#)

[lv_spinbox_private.h](#)

6.30 Spinner (lv_spinner)

6.30.1 Panoramica

Il Widget Spinner è un arco rotante lungo un anello, solitamente utilizzato per indicare che è in corso un'attività.

6.30.2 Parti e Stili

Le parti dello Spinner sono identiche a quelle di [Arc](#).

6.30.3 Utilizzo

Creare uno Spinner

Per creare uno Spinner, si usa `lv_spinner_create(parent)`.

Usare `lv_spinner_set_anim_params(spinner, spin_duration, angle)` per personalizzare la durata di una rotazione e la lunghezza dell'arco.

6.30.4 Eventi

I Widget Spinner non inviano eventi speciali.

Ulteriori Informazioni

Eventi di Arc

Scoprire di più sugli [Eventi](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

6.30.5 Tasti

Nessun *Tasto* viene elaborato dai Widget Spinner.

Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

6.30.6 Esempio

Spinner semplice

```
#include "../../lv_examples.h"
#if LV_USE_SPINNER && LV_BUILD_EXAMPLES

void lv_example_spinner_1(void)
{
    /*Create a spinner*/
    lv_obj_t * spinner = lv_spinner_create(lv_screen_active());
    lv_obj_set_size(spinner, 100, 100);
    lv_obj_center(spinner);
    lv_spinner_set_anim_params(spinner, 10000, 200);
}

#endif
```

6.30.7 API

[lv_spinner.h](#)

[lv_spinner_private.h](#)

6.31 Switch (lv_switch)

6.31.1 Panoramica

I Widget Switch hanno l'aspetto di piccoli cursori e vengono utilizzati per visualizzare, e facoltativamente modificare, un valore che può essere "on" o "off".

Per default, uno Switch è orientato orizzontalmente. Il suo orientamento sarà verticale se si imposta `width < height`.

6.31.2 Parti e Stili

- *LV_PART_MAIN* Background dello Switch; utilizza le *tipiche proprietà di stile del background*. padding rende l'indicatore più piccolo nella rispettiva direzione.
- *LV_PART_INDICATOR* L'indicatore che mostra lo stato corrente dello Switch; utilizza anch'esso le *tipiche proprietà di stile del background*.
- *LV_PART_KNOB* Un rettangolo (o cerchio) disegnato a sinistra o a destra dell'indicatore; utilizza anch'esso le *tipiche proprietà di stile del background* per modificare l'aspetto della manopola. Per default, la manopola è rotonda (lo stile del raggio può modificarlo) con un diametro leggermente inferiore al lato più piccolo dello slider. La manopola può essere ingrandita con i valori di padding. I valori di padding possono anche essere asimmetrici.

6.31.3 Utilizzo

Cambio di stato

Lo Switch utilizza lo stato standard *LV_STATE_CHECKED* state.

Per ottenere lo stato corrente dello Switch (con `true` impostato su ON), utilizzare `lv_obj_has_state(widget, LV_STATE_CHECKED)`.

Chiamare `lv_obj_add_state(widget, LV_STATE_CHECKED)` per attivarlo, oppure `lv_obj_remove_state(widget, LV_STATE_CHECKED)` per disattivarlo da programma.

Modificare l'orientamento

Quando viene creato uno Switch, il suo orientamento predefinito è *LV_SWITCH_ORIENTATION_AUTO*, che ne determina l'orientamento in base a width e height. È possibile modificare questo comportamento utilizzando `lv_switch_set_orientation(widget, orientation)`. I valori possibili per `orientation` sono:

- *LV_SWITCH_ORIENTATION_AUTO*
- *LV_SWITCH_ORIENTATION_HORIZONTAL*
- *LV_SWITCH_ORIENTATION_VERTICAL*

6.31.4 Eventi

- *LV_EVENT_VALUE_CHANGED* Inviato quando lo Switch cambia stato.

Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.31.5 Tasti

- *LV_KEY_UP/RIGHT* Accende lo Switch
- *LV_KEY_DOWN/LEFT* Spegne lo Switch
- *LV_KEY_ENTER* Attiva/disattiva lo Switch

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.31.6 Esempio

Switch semplice

```
#include "../../lv_examples.h"
#if LV_USE_SWITCH && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        LV_UNUSED(obj);
        LV_LOG_USER("State: %s\n", lv_obj_has_state(obj, LV_STATE_CHECKED) ? "On" : "Off");
    }
}

void lv_example_switch_1(void)
{
    lv_obj_set.flex_flow(lv_screen_active(), LV_FLEX_FLOW_COLUMN);
    lv_obj_set.flex_align(lv_screen_active(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    lv_obj_t * sw;

    sw = lv_switch_create(lv_screen_active());
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);
    lv_obj_add_flag(sw, LV_OBJ_FLAG_EVENT_BUBBLE);

    sw = lv_switch_create(lv_screen_active());
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);

    sw = lv_switch_create(lv_screen_active());
    lv_obj_add_state(sw, LV_STATE_DISABLED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);

    sw = lv_switch_create(lv_screen_active());
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_state(sw, LV_STATE_DISABLED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);
}

#endif
```

Orientamento dello Switch

```
#include "../../lv_examples.h"
#if LV_USE_SWITCH && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        LV_UNUSED(obj);
        LV_LOG_USER("State: %s\n", lv_obj_has_state(obj, LV_STATE_CHECKED) ? "On" : "Off");
    }
}

void lv_example_switch_2(void)
{
    lv_obj_set.flex_flow(lv_screen_active(), LV_FLEX_FLOW_ROW);
    lv_obj_set.flex_align(lv_screen_active(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    lv_obj_t * sw;

    sw = lv_switch_create(lv_screen_active());
    lv_obj_set_size(sw, 30, 60);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);

    sw = lv_switch_create(lv_screen_active());
    lv_obj_set_size(sw, 30, 60);
    lv_switch_set_orientation(sw, LV_SWITCH_ORIENTATION_VERTICAL);
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);
}

#endif
```

6.31.7 API

lv_switch.h

lv_switch_private.h

6.32 Tabella (lv_table)

6.32.1 Panoramica

Le "Table" [tabelle] sono costituite da righe, colonne e celle contenenti testo.

Il Widget Table è molto leggero perché memorizza solo le stringhe di testo. Non vengono creati veri e propri widget per le celle: vengono semplicemente disegnati al volo.

La Tabella viene aggiunta al gruppo di default (se impostato). La Table è un Widget modificabile, che consente la selezione di una cella con encoder e navigazione da tastiera.

6.32.2 Parti e Stili

- *LV_PART_MAIN* Il background della Table; utilizza le *tipiche proprietà di stile del background*.
- *LV_PART_ITEMS* Anche le celle della tabella utilizzano le *tipiche proprietà di stile del background* e le proprietà di stile del testo.

6.32.3 Utilizzo

Impostare il valore della cella

Le celle possono memorizzare solo testo, quindi i numeri devono essere convertiti in testo prima di essere visualizzati in una Tabella.

`lv_table_set_cell_value(table, row, col, "Content")`. Il testo viene salvato dalla Tabella, quindi il buffer contenente la stringa può essere una variabile locale.

È possibile utilizzare interruzioni di riga nel testo, ad esempio "Value\n60.3".

Nuove righe e colonne vengono aggiunte automaticamente in base alle esigenze.

Righe e Colonne

Per impostare esplicitamente il numero di righe e colonne, si usano `lv_table_set_row_count(table, row_cnt)` e `lv_table_set_column_count(table, col_cnt)`.

Larghezza e Altezza

La larghezza delle colonne può essere impostata con `lv_table_set_column_width(table, col_id, width)`. La larghezza complessiva del Widget Table sarà impostata sulla somma delle larghezze di tutte le colonne.

L'altezza viene calcolata automaticamente in base agli stili delle celle (carattere, spaziatura interna, ecc.) e al numero di righe.

Unione delle celle

Le celle possono essere unite orizzontalmente con `lv_table_set_cell_ctrl(table, row, col, LV_TABLE_CELL_CTRL_MERGE_RIGHT)`. Per unire più celle adiacenti, si chiama questa funzione per ciascuna cella.

Scrolling

Se la larghezza o l'altezza di una Tabella è impostata su `LV_SIZE_CONTENT`, tale dimensione verrà utilizzata per visualizzare l'intera tabella nella rispettiva direzione. Ad esempio, `lv_obj_set_size(table, LV_SIZE_CONTENT, LV_SIZE_CONTENT)` imposta automaticamente la dimensione della Tabella per visualizzare tutte le colonne e le righe.

Se la larghezza o l'altezza sono impostate su un numero inferiore alla loro dimensione "intrinseca", la tabella diventa scorrevole.

Impostare i dati utente della cella

I dati personalizzati possono essere associati a una cella della tabella. La durata dei dati deve essere gestita dall'utente. Se i dati vengono allocati dinamicamente, l'utente deve liberarli per evitare "leak" [perdite] di memoria. Ciò può essere fatto associandoli all'evento `LV_EVENT_DELETE` event.

```
static void table_delete_event_cb(lv_event_t * e) {
    lv_obj_t * my_table = lv_event_get_target_obj(e);
    const uint32_t row_count = lv_table_get_row_count(table);
    const uint32_t col_count = lv_table_get_column_count(table);

    /* Assuming every cell has custom data associated with it*/
    for(uint32_t i = 0; i < row_count; ++i) {
        for(uint32_t j = 0; j < col_count; ++j) {

```

(continues on next page)

(continua dalla pagina precedente)

```

    void * cell_user_data = lv_table_get_cell_user_data(table, i, j);
    lv_free(cell_user_data);
}
}

lv_obj_add_event_cb(table, table_delete_event_cb, LV_EVENT_DELETE, NULL);

```

6.32.4 Eventi

- *LV_EVENT_VALUE_CHANGED* Inviato quando una nuova cella viene selezionata con i tasti.

Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.32.5 Tasti

I seguenti *tasti* vengono elaborati dai Widget Table:

- LV_KEY_RIGHT/LEFT/UP/DOWN/ Selezione una cella.

Si noti che, come di consueto, lo stato di *LV_KEY_ENTER* viene tradotto in *LV_EVENT_PRESSED/PRESSING/RELEASED* ecc.

lv_table_get_selected_cell(table, &row, &col) può essere utilizzato per ottenere la cella attualmente selezionata. Riga e colonna verranno impostate su *LV_TABLE_CELL_NONE* se non è selezionata alcuna cella.

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.32.6 Esempio

Tabella semplice

```

#include "../../lv_examples.h"
#if LV_USE_TABLE && LV_BUILD_EXAMPLES

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    /*If the cells are drawn...*/
    if(base_dsc->part == LV_PART_ITEMS) {
        uint32_t row = base_dsc->id1;
        uint32_t col = base_dsc->id2;

        /*Make the texts in the first cell center aligned*/
        if(row == 0) {
            lv_draw_label_dsc_t * label_dsc = lv_draw_task_get_label_dsc(draw_task);
            if(label_dsc) {
                label_dsc->align = LV_TEXT_ALIGN_CENTER;
            }
            lv_draw_fill_dsc_t * fill_draw_dsc = lv_draw_task_get_fill_dsc(draw_task);
            if(fill_draw_dsc) {
                fill_draw_dsc->color = lv_color_mix(lv_palette_main(LV_PALETTE_BLUE), fill_draw_dsc->color, LV_OPA_20);
                fill_draw_dsc->opa = LV_OPA_COVER;
            }
        }
        /*In the first column align the texts to the right*/
        else if(col == 0) {
            lv_draw_label_dsc_t * label_dsc = lv_draw_task_get_label_dsc(draw_task);
            if(label_dsc) {
                label_dsc->align = LV_TEXT_ALIGN_RIGHT;
            }
        }

        /*Make every 2nd row grayish*/
        if((row != 0 && row % 2 == 0) || (row == 0)) {
            lv_draw_fill_dsc_t * fill_draw_dsc = lv_draw_task_get_fill_dsc(draw_task);
            if(fill_draw_dsc) {
                fill_draw_dsc->color = lv_color_mix(lv_palette_main(LV_PALETTE_GREY), fill_draw_dsc->color, LV_OPA_10);
                fill_draw_dsc->opa = LV_OPA_COVER;
            }
        }
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

void lv_example_table_1(void)
{
    lv_obj_t * table = lv_table_create(lv_screen_active());

    /*Fill the first column*/
    lv_table_set_cell_value(table, 0, 0, "Name");
    lv_table_set_cell_value(table, 1, 0, "Apple");
    lv_table_set_cell_value(table, 2, 0, "Banana");
    lv_table_set_cell_value(table, 3, 0, "Lemon");
    lv_table_set_cell_value(table, 4, 0, "Grape");
    lv_table_set_cell_value(table, 5, 0, "Melon");
    lv_table_set_cell_value(table, 6, 0, "Peach");
    lv_table_set_cell_value(table, 7, 0, "Nuts");

    /*Fill the second column*/
    lv_table_set_cell_value(table, 0, 1, "Price");
    lv_table_set_cell_value(table, 1, 1, "$7");
    lv_table_set_cell_value(table, 2, 1, "$4");
    lv_table_set_cell_value(table, 3, 1, "$6");
    lv_table_set_cell_value(table, 4, 1, "$2");
    lv_table_set_cell_value(table, 5, 1, "$5");
    lv_table_set_cell_value(table, 6, 1, "$1");
    lv_table_set_cell_value(table, 7, 1, "$9");

    /*Set a smaller height to the table. It'll make it scrollable*/
    lv_obj_set_height(table, 200);
    lv_obj_center(table);

    /*Add an event callback to to apply some custom drawing*/
    lv_obj_add_event_cb(table, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(table, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
}

#endif

```

Elenco leggero da tabella

```

#include "../../lv_examples.h"
#if LV_USE_TABLE && LV_BUILD_EXAMPLES

#define ITEM_CNT 200

static void draw_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);

    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    /*If the cells are drawn...*/
    if(base_dsc->part == LV_PART_ITEMS && lv_draw_task_get_type(draw_task) == LV_DRAW_TASK_TYPE_FILL) {
        /*Draw the background*/
        bool chk = lv_table_has_cell_ctrl(obj, base_dsc->id1, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
        lv_draw_rect_dsc_t rect_dsc;
        lv_draw_rect_dsc_init(&rect_dsc);
        rect_dsc.bg_color = chk ? lv_theme_get_color_primary(obj) : lv_palette_lighten(LV_PALETTE_GREY, 2);
        rect_dsc.radius = LV_RADIUS_CIRCLE;

        lv_area_t sw_area;
        sw_area.x1 = 0;
        sw_area.x2 = 40;
        sw_area.y1 = 0;
        sw_area.y2 = 24;
        lv_area_t draw_task_area;
        lv_draw_task_get_area(draw_task, &draw_task_area);
        lv_area_align(&draw_task_area, &sw_area, LV_ALIGN_RIGHT_MID, -15, 0);
        lv_draw_rect(base_dsc->layer, &rect_dsc, &sw_area);

        /*Draw the knob*/
        rect_dsc.bg_color = lv_color_white();
        lv_area_t knob_area;
        knob_area.x1 = 0;
        knob_area.x2 = 18;
        knob_area.y1 = 0;
        knob_area.y2 = 18;
        if(chk) {
            lv_area_align(&sw_area, &knob_area, LV_ALIGN_RIGHT_MID, -3, 0);
        } else {
            lv_area_align(&sw_area, &knob_area, LV_ALIGN_LEFT_MID, 3, 0);
        }
        lv_draw_rect(base_dsc->layer, &rect_dsc, &knob_area);
    }
}

static void change_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    uint32_t col;
    uint32_t row;
    lv_table_get_selected_cell(obj, &row, &col);
    bool chk = lv_table_has_cell_ctrl(obj, row, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
    if(chk) lv_table_clear_cell_ctrl(obj, row, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
    else lv_table_set_cell_ctrl(obj, row, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
}

/***
 * A very light-weighted list created from table
 */
void lv_example_table_2(void)
{
    /*Measure memory usage*/
    lv_mem_monitor_t mon1;
    lv_mem_monitor(&mon1);

    uint32_t t = lv_tick_get();

    lv_obj_t * table = lv_table_create(lv_screen_active());
}

```

(continues on next page)

(continua dalla pagina precedente)

```

/*Set a smaller height to the table. It'll make it scrollable*/
lv_obj_set_size(table, LV_SIZE_CONTENT, 200);

lv_table_set_column_width(table, 0, 150);
lv_table_set_row_count(table, ITEM_CNT); /*Not required but avoids a lot of memory reallocation lv_table_set_set_value*/
lv_table_set_column_count(table, 1);

/*Don't make the cell pressed, we will draw something different in the event*/
lv_obj_remove_style(table, NULL, LV_PART_ITEMS | LV_STATE_PRESSED);

uint32_t i;
for(i = 0; i < ITEM_CNT; i++) {
    lv_table_set_cell_value_fmt(table, i, 0, "Item %" LV_PRIu32, i + 1);
}

lv_obj_align(table, LV_ALIGN_CENTER, 0, -20);

/*Add an event callback to to apply some custom drawing*/
lv_obj_add_event_cb(table, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
lv_obj_add_event_cb(table, change_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
lv_obj_add_flag(table, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);

lv_mem_monitor_t mon2;
lv_mem_monitor(&mon2);

size_t mem_used = mon1.free_size - mon2.free_size;

uint32_t elaps = lv_tick_elaps(t);

lv_obj_t * label = lv_label_create(lv_screen_active());
lv_label_set_text_fmt(label, "%" LV_PRIu32 " items were created in %" LV_PRIu32 " ms\n"
                      "using %zu bytes of memory",
                      (uint32_t)ITEM_CNT, elaps, mem_used);

lv_obj_align(label, LV_ALIGN_BOTTOM_MID, 0, -10);
}

#endif

```

MicroPython

Nessun esempio ancora.

6.32.7 API

lv_api_map_v8.h
lv_table.h
lv_table_private.h

6.33 Tab View (lv_tabview)

6.33.1 Panoramica

Il Widget "Tab View" [Visualizzazione schede] può essere utilizzato per organizzare i contenuti in schede. La "Tab View" è composta da altri Widget:

- Contenitore principale: *Widget Base (lv_obj)*
- Pulsanti Tab: un *Widget Base (lv_obj)* con *Pulsante (lv_button)*
- Contenitore per le schede: *Widget Base (lv_obj)*
- Contenuto delle schede: *Widget Base (lv_obj)*

I pulsanti scheda possono essere posizionati in alto, in basso, a sinistra e a destra della "Tab View".

È possibile selezionare una nuova scheda cliccando su un pulsante scheda o scorrendo orizzontalmente sul contenuto.

6.33.2 Parti e Stili

Non ci sono parti specifiche nella "Tab View", ma i Widget *Widget Base (lv_obj)* e *Pulsante (lv_button)* vengono utilizzati per creare la "Tab View".

6.33.3 Utilizzo

Creazione di una Tab View

`lv_tabview_create(parent)` crea una nuova Tab View vuota.

Aggiunta di schede

È possibile aggiungere nuove schede con `lv_tabview_add_tab(tabview, "Tab name")`. Questo restituirà un puntatore a un *Widget Base* (`lv_obj`) in cui è possibile creare il contenuto della scheda.

Rinominare le schede

Un tab può essere rinominato con `lv_tabview_set_tab_text(tabview, tab_id, "New Name")`.

Passare a una nuova scheda

Per selezionare una nuova scheda si può:

- Cliccare sul pulsante della scheda
- Scorrere orizzontalmente
- Usare la funzione `lv_tabview_set_active(tabview, tab_id, LV_ANIM_ON / OFF)`

Impostare la posizione della barra delle schede

Utilizzando la funzione `lv_tabview_set_tab_bar_position(tabview, LV_DIR_LEFT / RIGHT / TOP / BOTTOM)` la barra delle schede può essere spostata su qualsiasi lato.

Impostare le dimensioni della barra delle schede

La dimensione della barra dei "tab" può essere regolata tramite `lv_tabview_set_tab_bar_size(tabview, size)`. Quando i tab sono in alto o in basso, questo si riferisce all'altezza della barra dei tab, mentre quando sono ai lati, si riferisce alla larghezza.

Accesso alle parti

- `lv_tabview_get_content(tabview)` restituisce un puntatore al contenitore per il contenuto delle schede (un *Widget Base* (`lv_obj`))
- `lv_tabview_get_tab_bar(tabview)` restituisce un puntatore al contenitore per i pulsanti delle schede (un *Widget Base* (`lv_obj`))

6.33.4 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato quando si seleziona una nuova scheda facendo scorrere o cliccando sul pulsante della scheda. `lv_tabview_get_tab_active(tabview)` restituisce l'indice a base zero della scheda corrente.

Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.33.5 Tasti

I tasti hanno effetto solo sui pulsanti delle schede. Aggiungere questi pulsanti a un gruppo tramite codice, se necessario.

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.33.6 Esempio

Tabview Semplice

```
#include "../../lv_examples.h"
#if LV_USE_TABVIEW && LV_BUILD_EXAMPLES

void lv_example_tabview_1(void)
{
    /*Create a Tab view object*/
    lv_obj_t * tabview;
    tabview = lv_tabview_create(lv_screen_active());

    /*Add 3 tabs (the tabs are page (lv_page) and can be scrolled*/
    lv_obj_t * tab1 = lv_tabview_add_tab(tabview, "Tab 1");
    lv_obj_t * tab2 = lv_tabview_add_tab(tabview, "Tab 2");
    lv_obj_t * tab3 = lv_tabview_add_tab(tabview, "Tab 3");

    /*Add content to the tabs*/
    lv_obj_t * label = lv_label_create(tab1);
    lv_label_set_text(label, "This the first tab\n\n"
                        "If the content\n"
                        "of a tab\n"
                        "becomes too\n"
                        "longer\n"
                        "than the\n"
                        "container\n"
                        "then it\n"
                        "automatically\n"
                        "becomes\n"
                        "scrollable.\n"
                        "\n"
                        "\n"
                        "\n"
                        "Can you see it?");

    label = lv_label_create(tab2);
    lv_label_set_text(label, "Second tab");

    label = lv_label_create(tab3);
    lv_label_set_text(label, "Third tab");

    lv_obj_scroll_to_view_recursive(label, LV_ANIM_ON);
}

#endif
```

Tab a sinistra, con stile e nessuno scorrimento

```
#include "../../lv_examples.h"
#if LV_USE_TABVIEW && LV_BUILD_EXAMPLES

/*A vertical tab view with disabled scrolling and some styling*/
void lv_example_tabview_2(void)
{
    /*Create a Tab view object*/
    lv_obj_t * tabview;
    uint32_t tab_count = 0;
    uint32_t i = 0;

    tabview = lv_tabview_create(lv_screen_active());
    lv_tabview_set_tab_bar_position(tabview, LV_DIR_LEFT);
    lv_tabview_set_tab_bar_size(tabview, 80);

    lv_obj_set_style_bg_color(tabview, lv_palette_lighten(LV_PALETTE_RED, 2), 0);

    lv_obj_t * tab_buttons = lv_tabview_get_tab_bar(tabview);
    lv_obj_set_style_bg_color(tab_buttons, lv_palette_darken(LV_PALETTE_GREY, 3), 0);
    lv_obj_set_style_text_color(tab_buttons, lv_palette_lighten(LV_PALETTE_GREY, 5), 0);

    /*Add 5 tabs (the tabs are page (lv_page) and can be scrolled*/
    lv_obj_t * tab1 = lv_tabview_add_tab(tabview, "Tab 1");
    lv_obj_t * tab2 = lv_tabview_add_tab(tabview, "Tab 2");
    lv_obj_t * tab3 = lv_tabview_add_tab(tabview, "Tab 3");
    lv_obj_t * tab4 = lv_tabview_add_tab(tabview, "Tab 4");
    lv_obj_t * tab5 = lv_tabview_add_tab(tabview, "Tab 5");

    tab_count = lv_tabview_get_tab_count(tabview);
    for(i = 0; i < tab_count; i++) {
        lv_obj_t * button = lv_obj_get_child(tab_buttons, i);
        lv_obj_set_style_border_side(button, LV_BORDER_SIDE_RIGHT, LV_PART_MAIN | LV_STATE_CHECKED);
    }
    lv_obj_set_style_bg_color(tab2, lv_palette_lighten(LV_PALETTE_AMBER, 3), 0);
    lv_obj_set_style_bg_opa(tab2, LV_OPA_COVER, 0);

    /*Add content to the tabs*/
    lv_obj_t * label = lv_label_create(tab1);
    lv_label_set_text(label, "First tab");

    label = lv_label_create(tab2);
    lv_label_set_text(label, "Second tab");

    label = lv_label_create(tab3);
    lv_label_set_text(label, "Third tab");

    label = lv_label_create(tab4);
    lv_label_set_text(label, "Fourth tab");
```

(continues on next page)

(continua dalla pagina precedente)

```

label = lv_label_create(tab5);
lv_label_set_text(label, "Fifth tab");

lv_obj_remove_flag(lv_tabview_get_content(tabview), LV_OBJ_FLAG_SCROLLABLE);
}
#endif

```

6.33.7 API

lv_api_map_v8.h
lv_api_map_v9_4.h
lv_tabview.h
lv_tabview_private.h

6.34 Text Area (lv_textarea)

6.34.1 Panoramica

La Text Area è un *Widget Base* (*lv_obj*) con una *Label* (*lv_label*) e un cursore. È possibile aggiungervi testo o caratteri. Le righe lunghe vengono mandate a capo e, quando il testo diventa sufficientemente lungo, è possibile scorrere la "Text Area".

Sono supportate la modalità a riga singola e la modalità password.

6.34.2 Parti e Stili

- *LV_PART_MAIN* Il background della Text Area; utilizza le *tipiche proprietà di stile del background* e le proprietà di stile relative al testo, tra cui *text_align*, per allineare il testo a sinistra, a destra o al centro.
- *LV_PART_SCROLLBAR* La scrollbar che viene visualizzata quando il testo è più lungo della sua altezza.
- *LV_PART_SELECTED* Determina lo stile del *testo selezionato*. È possibile utilizzare solo le proprietà di stile *text_color* e *bg_color*. *bg_color* deve essere impostato direttamente sull'etichetta della Text Area.
- *LV_PART_CURSOR* Indica la posizione in cui vengono inseriti i caratteri. L'area del cursore è sempre il riquadro di delimitazione del carattere corrente. È possibile creare un cursore a blocco aggiungendo un colore di sfondo e un'opacità di sfondo allo stile di *LV_PART_CURSOR*. Per creare un cursore a "barra", lasciare il cursore trasparente e impostare un bordo sinistro. La proprietà di stile *anim_time* imposta il tempo di lampeggiò del cursore.
- *LV_PART_TEXTAREA_PLACEHOLDER* Unica per la Text Area; consente di applicare uno stile al *testo segnaposto*.

6.34.3 Utilizzo

Aggiunta di testo

Si può inserire testo o caratteri nella posizione corrente del cursore con:

- *lv_textarea_add_char*(*textarea*, 'c')
- *lv_textarea_add_text*(*textarea*, "insert this text")

Per aggiungere caratteri estesi come 'á', 'ß' o caratteri CJK, si usa *lv_textarea_add_text*(*textarea*, "á").

lv_textarea_set_text(*textarea*, "New text") sostituisce tutto il testo esistente con "New text".

Testo segnaposto

Il testo segnaposto è il testo che viene visualizzato quando la Text Area è vuota. Questo può essere un modo pratico per fornire all'utente finale un suggerimento su cosa digitare.

Specificare il testo segnaposto utilizzando *lv_textarea_set_placeholder_text*(*textarea*, "Placeholder text").

Eliminare un carattere

Per eliminare il carattere a sinistra della posizione corrente del cursore, utilizzare `lv_textarea_delete_char(textarea)`.

Per eliminare a destra, utilizzare `lv_textarea_delete_char_forward(textarea)`

Spostamento del cursore

La posizione del cursore può essere modificata a livello di codice utilizzando `lv_textarea_set_cursor_pos(textarea, cursor_pos)` dove `cursor_pos` è l'indice a partire da zero del carattere davanti al quale deve essere posizionato il cursore. `LV_TEXTAREA_CURSOR_LAST` può essere passato per indicare "dopo l'ultimo carattere".

È possibile spostare il cursore di una posizione (o riga) alla volta con

- `lv_textarea_cursor_right(textarea)`
- `lv_textarea_cursor_left(textarea)`
- `lv_textarea_cursor_up(textarea)`
- `lv_textarea_cursor_down(textarea)`

Se si applica `lv_textarea_set_cursor_click_pos(textarea, true)`, il cursore salterà nella posizione in cui è stato cliccato sull'area di testo.

Nascondere il cursore

Il cursore è normalmente sempre visibile. Può essere una buona idea impostarne lo stile in modo che sia visibile solo nello stato `LV_STATE_FOCUSED`. Vedere gli [Stili](#) per maggiori informazioni su come farlo.

Modalità a una riga

La Text Area può essere configurata per mantenere tutto il testo su una singola riga con `lv_textarea_set_one_line(textarea, true)`. In questa modalità:

- l'altezza viene impostata automaticamente per visualizzare una sola riga,
- i caratteri di interruzione di riga vengono ignorati e
- il ritorno a capo automatico è disabilitato.

Modalità password

La Text Area supporta la modalità password, che può essere abilitata con `lv_textarea_set_password_mode(textarea, true)`.

Per default, se il carattere • (Bullet, U+2022) è presente nel font, i caratteri immessi vengono convertiti in tale carattere dopo un ritardo configurabile dopo ogni nuovo carattere immesso. Se • non è presente nel font, verrà utilizzato *. È possibile sovrascrivere il carattere di "mascheramento" di default con `lv_textarea_set_password_bullet(textarea, str)` dove `str` è una stringa C terminata da NUL. Esempio:

```
lv_textarea_set_password_bullet(textarea, "x");
```

In modalità password, `lv_textarea_get_text(textarea)` restituisce il testo effettivamente inserito, non i caratteri del punto elenco.

Il tempo di visibilità può essere regolato con `LV_TEXTAREA_DEF_PWD_SHOW_TIME` in `lv_conf.h`.

Caratteri accettati

È possibile impostare un elenco di caratteri accettati con `lv_textarea_set_accepted_chars(textarea, list)` dove `list` è un puntatore a una stringa terminata da NULL, oppure NULL per accettare tutti i caratteri. I caratteri inseriti non presenti in questo elenco verranno ignorati.

```
lv_textarea_set_accepted_chars(textarea, "0123456789.-");
```

Lunghezza massima del testo

Il numero massimo di caratteri può essere limitato utilizzando `lv_textarea_set_max_length(textarea, max_char_num)`.

Testo molto lungo

Se il testo nella Text Area fosse molto lungo (ad esempio > 20.000 caratteri), lo scorrimento e il disegno potrebbero risultare lenti. Tuttavia, impostando `LV_LABEL_LONG_TXT_HINT` in `lv_conf.h` su un valore diverso da zero, le prestazioni con il testo lungo migliorano significativamente. Questo risultato viene ottenuto salvando alcune informazioni aggiuntive sulla posizione verticale corrente del testo visualizzato. Con questa modalità configurata, lo scorrimento e il disegno sono veloci quanto con il testo breve "normale". Il costo è di 12 byte extra per ogni etichetta in RAM.

Questo valore è impostato su 1 per default. Se non si utilizza testo lungo, è possibile risparmiare 12 byte per etichetta impostandolo su 0.

Selezione del testo

Se `LV_LABEL_TEXT_SELECTION` è impostato su un valore diverso da zero in `lv_conf.h`, alcune funzionalità aggiuntive (e 8 byte per etichetta) vengono aggiunte ai Widget Label e ai Widget Text Area, e la funzionalità di selezione del testo viene automatizzata nei Widget Text Area. (Se non si utilizza testo selezionato nell'applicazione, è possibile risparmiare 8 byte per etichetta nella RAM impostando tale macro su 0.)

È possibile selezionare qualsiasi parte del testo se abilitata con `lv_textarea_set_text_selection(textarea, true)`. Funziona in modo molto simile a quando si seleziona del testo sul PC cliccando e trascinando con il mouse o un altro dispositivo di input puntatore. Se si passa `false` a questa funzione per disabilitare la selezione del testo, qualsiasi testo selezionato al momento della chiamata verrà deselectionato.

In futuro si prevede di aggiungere la selezione del testo tramite Shift+Click e la selezione del testo tramite tastiera con Shift+Frecchia.

Se è necessario gestire il testo selezionato a livello di codice, oltre alla funzione `lv_textarea_set_text_selection(textarea, enable)`, ecco gli strumenti a disposizione. (`ta_label` è un puntatore all'etichetta della Text Area recuperata con `ta_label = lv_textarea_get_label(textarea);`)

- `lv_textarea_get_text_selection(textarea)` indica se la selezione del testo è abilitata.
- `lv_textarea_text_is_selected(textarea)` indica se è attualmente selezionato del testo.
- `lv_textarea_clear_selection(textarea)` cancella la selezione del testo corrente.
- `lv_label_set_text_selection_start(ta_label, index)` dove `index` è l'indice a partire da zero del primo carattere del testo selezionato. Passare `LV_DRAW_LABEL_NO_TXT_SEL` per specificare che non è stato selezionato alcun testo.
- `lv_label_set_text_selection_end(ta_label, index)` dove `index` è l'indice a partire da zero del carattere subito dopo il testo selezionato. Passare `LV_DRAW_LABEL_NO_TXT_SEL` per specificare che non è stato selezionato alcun testo.
- `lv_label_get_text_selection_start(ta_label)` indice a partire da zero del primo carattere del testo selezionato. `LV_DRAW_LABEL_NO_TXT_SEL` indica che non è stato selezionato alcun testo.
- `lv_label_get_text_selection_end(ta_label)` indice a base zero del carattere subito dopo il testo selezionato. `LV_DRAW_LABEL_NO_TXT_SEL` indica che non è stato selezionato alcun testo.

Normalmente non ce n'è bisogno, poiché la Text Area automatizza il processo di selezione del testo, ma per modificare la selezione a livello di codice, gli strumenti sopra indicati sono quelli da usare.

6.34.4 Eventi

- ***LV_EVENT_INSERT*** Inviato subito prima dell'inserimento di un carattere o di un testo. Il parametro evento è il testo che sta per essere inserito. `lv_textarea_set_insert_replace(textarea, "New text")` può essere chiamato dall'interno di quell'evento per sostituire il testo da inserire. Il contenuto del buffer passato deve sopravvivere abbastanza a lungo da consentire il ritorno della chiamata a `lv_timer_handler()` che gestisce l'evento (dopodiché l'etichetta dell'area di testo avrà copiato il testo). Quindi non dovrebbe trattarsi di un buffer locale (creato nello stack) il cui contenuto verrà distrutto prima che ciò accada. Passare "" significa "non inserire nulla".
- ***LV_EVENT_VALUE_CHANGED*** Inviato quando il contenuto della Text Area è cambiato.
- ***LV_EVENT_READY*** Inviato quando ***LV_KEY_ENTER*** viene premuto (o inviato) a una Text Area di una sola riga.

Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.34.5 Tasti

- **LV_KEY_UP/DOWN/LEFT/RIGHT** Sposta il cursore
- **Any character** Aggiunge il carattere alla posizione corrente del cursore

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.34.6 Esempio

Area di testo semplice

```
#include "../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_BUILD_EXAMPLES

static void textarea_event_handler(lv_event_t * e)
{
    lv_obj_t * ta = lv_event_get_target_obj(e);
    LV_UNUSED(ta);
    LV_LOG_USER("Enter was pressed. The current text is: %s", lv_textarea_get_text(ta));
}

static void btmn_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_obj_t * ta = (lv_obj_t *)lv_event_get_user_data(e);
    const char * txt = lv_buttonmatrix_get_button_text(obj, lv_buttonmatrix_get_selected_button(obj));

    if(lv_strcmp(txt, LV_SYMBOL_BACKSPACE) == 0) lv_textarea_delete_char(ta);
    else if(lv_strcmp(txt, LV_SYMBOL_NEW_LINE) == 0) lv_obj_send_event(ta, LV_EVENT_READY, NULL);
    else lv_textarea_add_text(ta, txt);
}

void lv_example_textarea_1(void)
{
    lv_obj_t * ta = lv_textarea_create(lv_screen_active());
    lv_textarea_set_one_line(ta, true);
    lv_obj_align(ta, LV_ALIGN_TOP_MID, 0, 10);
    lv_obj_add_event_cb(ta, textarea_event_handler, LV_EVENT_READY, ta);
    lv_obj_add_state(ta, LV_STATE_FOCUSED); /*To be sure the cursor is visible*/

    static const char * btmn_map[] = {"1", "2", "3", "\n",
                                    "4", "5", "6", "\n",
                                    "7", "8", "9", "\n",
                                    LV_SYMBOL_BACKSPACE, "0", LV_SYMBOL_NEW_LINE, ""};

    lv_obj_t * btmn = lv_buttonmatrix_create(lv_screen_active());
    lv_obj_set_size(btmn, 200, 150);
    lv_obj_align(btmn, LV_ALIGN_BOTTOM_MID, 0, -10);
    lv_obj_add_event_cb(btmn, btmn_event_handler, LV_EVENT_VALUE_CHANGED, ta);
    lv_obj_remove_flag(btmn, LV_OBJ_FLAG_CLICK_FOCUSABLE); /*To keep the text area focused on button clicks*/
    lv_buttonmatrix_set_map(btmn, btmn_map);
}

#endif
```

Area di testo con campo password

```
#include "../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e);

static lv_obj_t * kb;

void lv_example_textarea_2(void)
{
    /*Create the password box*/
    lv_obj_t * pwd_ta = lv_textarea_create(lv_screen_active());
    lv_textarea_set_text(pwd_ta, "");
    lv_textarea_set_password_mode(pwd_ta, true);
    lv_textarea_set_one_line(pwd_ta, true);
    lv_obj_set_width(pwd_ta, lv_pct(40));
    lv_obj_set_pos(pwd_ta, 5, 20);
    lv_obj_add_event_cb(pwd_ta, ta_event_cb, LV_EVENT_ALL, NULL);

    /*Create a label and position it above the text box*/
    lv_obj_t * pwd_label = lv_label_create(lv_screen_active());
    lv_label_set_text(pwd_label, "Password:");
    lv_obj_align_to(pwd_label, pwd_ta, LV_ALIGN_OUT_TOP_LEFT, 0, 0);

    /*Create the one-line mode text area*/
    lv_obj_t * text_ta = lv_textarea_create(lv_screen_active());
    lv_textarea_set_one_line(text_ta, true);
    lv_textarea_set_password_mode(text_ta, false);
    lv_obj_set_width(text_ta, lv_pct(40));
    lv_obj_add_event_cb(text_ta, ta_event_cb, LV_EVENT_ALL, NULL);
    lv_obj_align(text_ta, LV_ALIGN_TOP_RIGHT, -5, 20);

    /*Create a label and position it above the text box*/
    lv_obj_t * oneline_label = lv_label_create(lv_screen_active());
    lv_label_set_text(oneline_label, "Text!");
    lv_obj_align_to(oneline_label, text_ta, LV_ALIGN_OUT_TOP_LEFT, 0, 0);

    /*Create a keyboard*/
    kb = lv_keyboard_create(lv_screen_active());
    lv_obj_set_size(kb, LV_HOR_RES, LV_VER_RES / 2);

    lv_keyboard_set_textarea(kb, pwd_ta); /*Focus it on one of the text areas to start*/

    /*The keyboard will show Arabic characters if they are enabled */
    #if LV_USE_ARABIC_PERSIAN_CHARS && LV_FONT_DEJAVU_16_PERSIAN_HEBREW
    lv_obj_set_style_text_font(kb, &lv_font_dejavu_16_persian_hebrew, 0);
    lv_obj_set_style_text_font(text_ta, &lv_font_dejavu_16_persian_hebrew, 0);
    lv_obj_set_style_text_font(pwd_ta, &lv_font_dejavu_16_persian_hebrew, 0);
    #endif
}

static void ta_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * ta = lv_event_get_target_obj(e);
    if(code == LV_EVENT_CLICKED || code == LV_EVENT_FOCUSED) {
        /*Focus on the clicked text area*/
        if(kb != NULL) lv_keyboard_set_textarea(kb, ta);
    }

    else if(code == LV_EVENT_READY) {
        LV_LOG_USER("Ready, current text: %s", lv_textarea_get_text(ta));
    }
}
#endif

```

Formattazione automatica del testo

```
#include "../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e);

static lv_obj_t * kb;

/***
 * Automatically format text like a clock. E.g. "12:34"
 * Add the ':' automatically.
 */
void lv_example_textarea_3(void)
{
    /*Create the text area*/
    lv_obj_t * ta = lv_textarea_create(lv_screen_active());
    lv_obj_add_event_cb(ta, ta_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_textarea_set_accepted_chars(ta, "0123456789:");
    lv_textarea_set_max_length(ta, 5);
    lv_textarea_set_one_line(ta, true);
    lv_textarea_set_text(ta, "");

    /*Create a keyboard*/
    kb = lv_keyboard_create(lv_screen_active());
    lv_obj_set_size(kb, LV_HOR_RES, LV_VER_RES / 2);
    lv_keyboard_set_mode(kb, LV_KEYBOARD_MODE_NUMBER);
    lv_keyboard_set_textarea(kb, ta);
}

static void ta_event_cb(lv_event_t * e)
{
    lv_obj_t * ta = lv_event_get_target_obj(e);
    const char * txt = lv_textarea_get_text(ta);
    if(txt[0] >= '0' && txt[0] <= '9' &&
       txt[1] >= '0' && txt[1] <= '9' &&
       txt[2] != ':') {
        lv_textarea_set_cursor_pos(ta, 2);
        lv_textarea_add_char(ta, ':');
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```
}
#endif
```

Stile del cursore dell'area di testo

```
#include "../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void create_styled_textarea_cursor(const char * txt, int32_t y_ofs, lv_style_t * cursor_style)
{
    lv_obj_t * ta = lv_textarea_create(lv_screen_active());
    lv_textarea_set_text(ta, txt);
    lv_obj_set_width(ta, 280);
    lv_obj_align(ta, LV_ALIGN_TOP_MID, 0, y_ofs);
    lv_textarea_set_one_line(ta, true);
    lv_obj_add_state(ta, LV_STATE_FOCUSED);
    lv_obj_add_style(ta, cursor_style, LV_PART_CURSOR | LV_STATE_FOCUSED);
    lv_textarea_set_cursor_pos(ta, 0);
}

void lv_example_textarea_4(void)
{
    static lv_style_t style_simple, style_block, style_underline;

    /* Thin left bar cursor (simple) */
    lv_style_init(&style_simple);
    lv_style_set_border_color(&style_simple, lv_palette_main(LV_PALETTE_RED));

    /* Underline cursor */
    lv_style_init(&style_underline);
    lv_style_set_bg_opa(&style_underline, LV_OPA_TRANSP);
    lv_style_set_border_color(&style_underline, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_border_side(&style_underline, LV_BORDER_SIDE_BOTTOM);
    lv_style_set_pad_hori(&style_underline, 1); /* set width of cursor using pad */
    lv_style_set_border_width(&style_underline, 3); /* set thickness of underline cursor */

    /* Full block cursor with many styles */
    lv_style_init(&style_block);
    lv_style_set_bg_opa(&style_block, LV_OPA_COVER);
    lv_style_set_bg_color(&style_block, lv_palette_main(LV_PALETTE_ORANGE));
    lv_style_set_bg_grad_color(&style_block, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_bg_grad_dir(&style_block, LV_GRAD_DIR_VER);
    lv_style_set_border_color(&style_block, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_border_side(&style_block, LV_BORDER_SIDE_FULL);
    lv_style_set_border_width(&style_block, 4);
    lv_style_set_text_color(&style_block, lv_color_white());
    lv_style_set_pad_all(&style_block, 1); /* set width of cursor using pad */

    /* Create 3 independent textareas, each with a unique styled cursor */
    create_styled_textarea_cursor("This is a simple red cursor", 10, &style_simple);
    create_styled_textarea_cursor("This is an underline blue cursor", 110, &style_underline);
    create_styled_textarea_cursor("This is a complex block cursor", 60, &style_block);
}

#endif
```

6.34.7 API

lv_obj_property_names.h
lv_textarea.h
lv_textarea_private.h

6.35 Tile View (lv_tileview)

6.35.1 Panoramica

La Tile View è un Widget contenitore i cui elementi (chiamati *tile*) possono essere disposti in una griglia. Un utente può navigare tra i tile scorrendo. Qualsiasi direzione di scorrimento può essere disabilitata sui tile singolarmente per impedire lo spostamento da un tile all'altro.

Se la Tile View è dimensionata per lo schermo, l'interfaccia utente assomiglia a quella ai potrebbe aver visto sugli smartwatch.

6.35.2 Parti e Stili

La Tile View è composta da un contenitore *Widget Base* (*lv_obj*) e da tile *Widget Base* (*lv_obj*).

Le parti e gli stili funzionano come per *Widget Base* (*lv_obj*).

6.35.3 Utilizzo

Aggiungere un tile

`lv_tileview_add_tile(tileview, col_id, row_id, dir)` crea un nuovo tile nella colonna `col_id`-esima e nella riga `row_id`-esima. `dir` può essere `LV_DIR_LEFT/RIGHT/TOP/BOTTOM/HOR/VER/ALL` o valori combinati OR per consentire lo spostamento ai tile [riquadri] adiacenti nella direzione specificata tramite scorrimento.

Il valore restituito è un puntatore al tile (un *Widget Base* (`lv_obj`)) su cui è possibile creare il contenuto del "tab".

Cambio di 'Tile'

La Tile View può scorrere fino a un tile specificato con `lv_tileview_set_tile(tileview, tile, LV_ANIM_ON / OFF)` o con `lv_tileview_set_tile_by_index(tileview, col_id, row_id, LV_ANIM_ON / OFF)`.

6.35.4 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato dopo la visualizzazione di un nuovo tile tramite scorrimento. `lv_tileview_get_tile_active(tileview)` può essere utilizzato all'interno dell'evento per ottenere un puntatore al tile appena visualizzato.

Ulteriori Informazioni

Scoprire di più sugli *Eventi* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

6.35.5 Tasti

Nessun *Tasto* viene elaborato dai Widget Tileview.

Ulteriori Informazioni

Scoprire di più sui *Tasti*.

6.35.6 Esempio

Tileview con contenuto

```
#include "../../lv_examples.h"
#if LV_USE_TILEVIEW && LV_BUILD_EXAMPLES

/**
 * Create a 2x2 tile view and allow scrolling only in an "L" shape.
 * Demonstrate scroll chaining with a long list that
 * scrolls the tile view when it can't be scrolled further.
 */
void lv_example_tileview_1(void)
{
    lv_obj_t * tv = lv_tileview_create(lv_screen_active());

    /*Tile1: just a label*/
    lv_obj_t * tile1 = lv_tileview_add_tile(tv, 0, 0, LV_DIR_BOTTOM);
    lv_obj_t * label = lv_label_create(tile1);
    lv_label_set_text(label, "Scroll down");
    lv_obj_center(label);

    /*Tile2: a button*/
    lv_obj_t * tile2 = lv_tileview_add_tile(tv, 0, 1, (lv_dir_t)(LV_DIR_TOP | LV_DIR_RIGHT));
    lv_obj_t * btn = lv_button_create(tile2);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Scroll up or right");

    lv_obj_set_size(btn, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_center(btn);

    /*Tile3: a list*/
    lv_obj_t * tile3 = lv_tileview_add_tile(tv, 1, 1, LV_DIR_LEFT);
    lv_obj_t * list = lv_list_create(tile3);
    lv_obj_set_size(list, LV_PCT(100), LV_PCT(100));

    lv_list_add_button(list, NULL, "One");
    lv_list_add_button(list, NULL, "Two");
    lv_list_add_button(list, NULL, "Three");
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_list_add_button(list, NULL, "Four");
lv_list_add_button(list, NULL, "Five");
lv_list_add_button(list, NULL, "Six");
lv_list_add_button(list, NULL, "Seven");
lv_list_add_button(list, NULL, "Eight");
lv_list_add_button(list, NULL, "Nine");
lv_list_add_button(list, NULL, "Ten");

}

#endif

```

6.35.7 API

lv_api_map_v8.h

lv_tileview.h

lv_tileview_private.h

6.36 Window (lv_win)

6.36.1 Panoramica

Il Widget Window è costituito da un'intestazione (simile a una barra del titolo) con titolo, pulsanti e un'area per i contenuti.

6.36.2 Parti e Stili

La Window è costruita a partire da altri Widget, quindi di può consultare la relativa documentazione per i dettagli:

- Background: *Widget Base (lv_obj)*
- Intestazione sul background: *Widget Base (lv_obj)*
- Titolo sull'intestazione: *Label (lv_label)*
- Pulsanti sull'intestazione: *Pulsante (lv_button)*
- Content Area sul background: *Widget Base (lv_obj)*

6.36.3 Utilizzo

Creazione di una Window

lv_win_create(parent) crea una Window inizialmente composta dalla seguente struttura di Widget:

- Background (un *Widget Base (lv_obj)*, il contenitore della finestra principale), è configurato per essere un contenitore Flex-Flow che distribuisce verticalmente i Widget in esso contenuti (*LV_FLEX_FLOW_COLUMN*).
- L'Header (Intestazione) (come una barra del titolo) è inizialmente vuota ed è un contenitore Flex-Flow configurato per disporre i Widget in essa contenuti orizzontalmente (*LV_FLEX_FLOW_ROW*), da sinistra a destra. L'intestazione occupa l'intera larghezza del Background (il suo elemento padre) e la parte superiore per circa 1,25 cm (secondo *LV_DPI_DEF*).
- La "Content Area" (un *Widget Base (lv_obj)*) occupa l'intera larghezza del Background (il suo elemento padre) e la parte rimanente dell'area del Background disponibile sotto l'Header. Non è *di per sé* un contenitore Flex-Flow, ma è possibile renderlo tale. Vedere *Flex* per i dettagli.

Titolo e pulsanti

È possibile aggiungere widget Button e Label all'Header utilizzando queste due funzioni. Verranno posizionati nell'intestazione in ordine da sinistra a destra man mano che vengono aggiunti. Queste funzioni possono essere chiamate in qualsiasi ordine e un numero qualsiasi di volte.

- *lv_win_add_title(win, "The title")* aggiunge un'etichetta all'intestazione, con modalità estesa *LV_LABEL_LONG_DOT* in modo che, se il contenuto del testo è più largo dell'area occupata, il testo verrà troncato con

un'ellissi ("...") alla fine. È impostato su FLEX GROW 1, quindi se è l'unica Label nell'intestazione, occuperà tutto lo spazio disponibile dopo l'aggiunta di eventuali Pulsanti. Se vengono aggiunte più etichette, ciascuna condividerà equamente lo spazio, a meno che il suo valore di FLEX GROW non venga modificato. Consultare [Flex](#) per i dettagli su come funziona. Per questo motivo, tutti i pulsanti aggiunti dopo l'ultima etichetta verranno "impilati" all'estrema destra dell'Header.

- `lv_win_add_button(win, icon, button_width)` aggiunge un Pulsante con la larghezza specificata che occupa l'intera altezza dell'Header (il suo elemento padre). Se `icon` non è NULL, viene creata un'immagine, centrata sul pulsante, utilizzando `icon` come sorgente. Sono supportate tutte le sorgenti di immagini valide, ma una sorgente comune da utilizzare è una delle macro `LV_SYMBOL_...`, come `LV_SYMBOL_CLOSE` per fornire un simbolo "x" (chiudi). Si ottiene un puntatore al Pulsante creato, in modo da poter aggiungere una callback di evento e/o qualsiasi altra cosa necessaria.

6.36.4 Ottener le parti

`lv_win_get_header(win)` restituisce un puntatore all'intestazione, `lv_win_get_content(win)` restituisce un puntatore al contenitore a cui è possibile aggiungere il contenuto della finestra.

6.36.5 Eventi

I Widget Window non inviano eventi speciali, tuttavia è possibile aggiungere eventi a qualsiasi Pulsante aggiunto.

Ulteriori Informazioni

Scoprire di più sugli [Eventi](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

6.36.6 Tasti

Nessun *Tasto* viene elaborato dai Widget Window.

Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

6.36.7 Esempio

Window semplice

```
#include "../../lv_examples.h"
#if LV_USE_WIN && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    LV_UNUSED(obj);
    LV_LOG_USER("Button %d clicked", (int)lv_obj_get_index(obj));
}

void lv_example_win_1(void)
{
    lv_obj_t * win = lv_win_create(lv_screen_active());
    lv_obj_t * btn;
    btn = lv_win_add_button(win, LV_SYMBOL_LEFT, 40);
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_win_add_title(win, "A title");

    btn = lv_win_add_button(win, LV_SYMBOL_RIGHT, 40);
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    btn = lv_win_add_button(win, LV_SYMBOL_CLOSE, 60);
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_obj_t * cont = lv_win_get_content(win); /*Content can be added here*/
    lv_obj_t * label = lv_label_create(cont);
    lv_label_set_text(label, "This is\n"
        "a pretty\n"
        "long text\n"
        "to see how\n"
        "the window\n"
        "becomes\n");
}

```

(continues on next page)

(continua dalla pagina precedente)

```
    "scrollable.\n"
    "\n"
    "\n"
    "Some more\n"
    "text to be\n"
    "sure it\n"
    "overflows. :))";
}

#endif
```

6.36.8 API

lv_windows_context.h

lv_windows_display.h

lv_windows_input.h

lv_win.h

lv_win_private.h

6.37 Nuovo Widget

CAPITOLO 7

Moduli Principali

7.1 Display (lv_display)

7.1.1 Panoramica

Cos'è un Display?

In LVGL, un *lv_display_t* (da non confondere con *Screen*) è un tipo di dati che rappresenta un singolo pannello di visualizzazione, ovvero l'hardware che visualizza i pixel renderizzati in LVGL sul dispositivo.

Quanti Display può Utilizzare LVGL?

LVGL può utilizzare un numero qualsiasi di display. È limitato solo dalla RAM disponibile e dal tempo MCU.

Perché si dovrebbero supportare più display? Ecco alcuni esempi:

- Avere un display TFT "normale" con interfaccia utente locale e creare schermi "virtuali" su VNC su richiesta. (È necessario aggiungere il driver VNC).
- Avere un display TFT di grandi dimensioni e un piccolo display monocromatico.
- Avere alcuni display più piccoli e semplici in uno strumento o dispositivo tecnologico di grandi dimensioni.
- Avere due display TFT di grandi dimensioni: uno per il cliente e uno per il commesso.

Se si imposta LVGL per utilizzare più di un display, tenere presente che alcune funzioni utilizzano il *Display di Default* durante la loro esecuzione, ad esempio la creazione di *Screen*.

Attributi

Una volta creato, un oggetto Display ricorda le caratteristiche dell'hardware di visualizzazione che rappresenta, così come altri elementi rilevanti per la sua durata:

- Risoluzione (larghezza e altezza in pixel)
- Profondità Colore (bit per pixel)
- Formato Colore (come vengono visualizzati i colori in pixel)
- DPI (il valore di default è *LV_DPI_DEF* in *lv_conf.h*, ma può essere modificato con *lv_display_set_dpi*(*disp*, *new_dpi*)).
- 4 *Layer dello Screen* creati automaticamente con ogni display

- Tutti gli *Screen* creati in associazione a questo display (e non ancora eliminati: ne viene visualizzato solo uno alla volta)
- I *Buffer di Disegno* assegnati
- La funzione *Callback Flush* che sposta i pixel dal *Buffer di Disegno* all'hardware del Display
- Quali aree del display sono state aggiornate (rese "sporche") in modo che la logica di rendering possa calcolare cosa visualizzare durante un aggiornamento del display
- Puntatore custom facoltativo come *Dati Utente*

Dati Utente

Con `lv_display_set_user_data(display1, p)` è possibile memorizzare un puntatore personalizzato nell'oggetto `lv_display_t`. Questo puntatore può essere utilizzato in seguito, ad esempio in *Eventi del Display*. Vedere l'esempio di codice per sapere come farlo in *Eventi del Display*.

API

lv_display.h

7.1.2 Configurazione del/i Display

Durante l'inizializzazione del sistema, è necessario eseguire le seguenti operazioni per ogni pannello di visualizzazione fisico che si desidera venga utilizzato da LVGL:

- *creargli un oggetto lv_display_t*,
- assegnare i relativi *Buffer di Disegno*, e
- assegnargli una *Callback Flush*.

Creazione di un Display

Per creare un display per LVGL:

```
lv_display_t * display1 = lv_display_create(hor_res, ver_res)
```

È possibile creare *più display* con un driver diverso per ciascuno (vedere di seguito).

Quando viene creato un oggetto `lv_display_t`, vengono creati 4 schermi impostati per aiutare a gestire la sovrapposizione dei Widget visualizzati. Vedere *Screen Trasparenti* e *Layer dello Screen* per ulteriori informazioni.

Display di Default

Quando viene creato il primo oggetto *Display* (`lv_display`), questo diventa il Display di Default. Se vengono creati altri Oggetti Display (per gestire altri Pannelli Display), quello di Default rimane il primo creato.

Per impostare un altro *Display* (`lv_display`) come Display di Default, si chiama `lv_display_set_default()`.

Vedere *Quanti Display può Utilizzare LVGL?* per ulteriori informazioni sull'utilizzo di display multipli.

Per molte funzioni `lv_display_...()`, il passaggio di NULL per l'argomento `disp` farà sì che la funzione utilizzi il Display di Default. Per esserne certi, consultare la documentazione API della funzione che si sta chiamando.

Buffer di Disegno

Durante l'inizializzazione del sistema, è necessario impostare i buffer di disegno che LVGL utilizzerà per ogni display. Per farlo, chiamare:

```
lv_display_set_buffers(display1, buf1, buf2, buf_size_in_bytes, render_mode)
```

- `buf1`, un buffer in cui LVGL può eseguire il rendering dei pixel
- `buf2`, un secondo buffer opzionale (vedere di seguito)

- `buf_size_in_bytes`, dimensione del/i buffer in byte
- `render_mode` è uno dei seguenti:
 - `LV_DISPLAY_RENDER_MODE_PARTIAL` Utilizzare il/i buffer per eseguire il rendering sul display utilizzando buffer più piccoli delle dimensioni del display. Si consiglia l'utilizzo di buffer di dimensioni pari ad almeno 1/10 delle dimensioni del display. In *Callback Flush*, le immagini renderizzate devono essere copiate nell'area specificata del display. In questa modalità, se si preme un pulsante, verrà ridisegnata solo l'area del pulsante.
 - `LV_DISPLAY_RENDER_MODE_DIRECT` La dimensione del buffer deve corrispondere alla dimensione del display. LVGL eseguirà il rendering nella posizione corretta del buffer. Con questo metodo, i buffer conterranno sempre l'intera immagine del display. Se vengono utilizzati due buffer, le aree renderizzate vengono automaticamente copiate nell'altro buffer dopo il flush. Per questo motivo, in *Callback Flush* in genere è necessario modificare solo l'indirizzo di un frame buffer. Se si preme un pulsante, verrà ridisegnata solo l'area del pulsante.
 - `LV_DISPLAY_RENDER_MODE_FULL` La dimensione del buffer deve corrispondere alla dimensione del display. LVGL eseguirà sempre il rendering dell'intero schermo, anche se è stato modificato solo 1 pixel. Se vengono forniti due buffer di disegno delle dimensioni del display, la gestione del display da parte di LVGL funziona come un doppio buffer "tradizionale". Ciò significa che la *Callback Flush* deve solo aggiornare l'indirizzo del frame buffer al parametro `px_map`.

Semplice Esempio

```
/* Declare buffer for 1/10 screen size; BYTES_PER_PIXEL will be 2 for RGB565. */
#define BYTES_PER_PIXEL (LV_COLOR_FORMAT_GET_SIZE(LV_COLOR_FORMAT_RGB565))
static uint8_t buf1[MY_DISP_HOR_RES * MY_DISP_VER_RES / 10 * BYTES_PER_PIXEL];
/* Set display buffer for display `display1`.. */
lv_display_set_buffers(display1, buf1, NULL, sizeof(buf1), LV_DISPLAY_RENDER_MODE_PARTIAL);
```

Un Buffer

Se viene utilizzato un solo buffer, LVGL disegna il contenuto dello schermo in quel buffer e lo invia al display tramite *Callback Flush*. LVGL attende quindi che venga chiamata `lv_display_flush_ready()` (ovvero che il contenuto del buffer sia stato inviato al display) prima di disegnarvi qualcosa di nuovo.

Due Buffer

Se vengono utilizzati due buffer, LVGL può disegnare in un buffer mentre il contenuto dell'altro buffer viene inviato al display in background. È necessario utilizzare DMA o altro hardware per trasferire i dati al display in modo che la MCU possa continuare a disegnare. In questo modo, il *rendering* e l'*aggiornamento* del display diventano operazioni parallele.

Tre Buffer

Il triplo buffering migliora il parallelismo tra rendering e trasferimento dati rispetto al doppio buffering. Quando un buffer ha completato il rendering e un altro è in fase di trasferimento DMA, il terzo buffer abilita il rendering immediato del frame successivo, eliminando i tempi di inattività di CPU/GPU causati dall'attesa del completamento del DMA. Il terzo buffer viene configurato utilizzando la funzione `lv_display_set_3rd_draw_buffer()`.

Callback Flush

I buffer di disegno sono semplici array che LVGL utilizza per il rendering del contenuto del display. Una volta completato il rendering, il contenuto del buffer di disegno viene inviato al display tramite una funzione di *Callback Flush*.

Un esempio è il seguente:

```
void my_flush_cb(lv_display_t * display, const lv_area_t * area, uint8_t * px_map)
{
    /* The most simple case (also the slowest) to send all rendered pixels to the
     * screen one-by-one. 'put_px' is just an example. It needs to be implemented by you. */
    uint16_t * buf16 = (uint16_t *)px_map; /* Let's say it's a 16 bit (RGB565) display */
    int32_t x, y;
```

(continues on next page)

(continua dalla pagina precedente)

```

for(y = area->y1; y <= area->y2; y++) {
    for(x = area->x1; x <= area->x2; x++) {
        put_px(x, y, *buf16);
        buf16++;
    }
}

/* IMPORTANT!!!
 * Inform LVGL that flushing is complete so buffer can be modified again. */
lv_display_flush_ready(display);
}

```

Durante l'inizializzazione del sistema, indica a LVGL che si vuole che la funzione copi i pixel dai buffer di pixel renderizzati a un display specifico, procedendo come segue:

```
lv_display_set_flush_cb(display, my_flush_cb)
```

Si noti che il display di destinazione viene passato alla funzione, quindi si può utilizzare la stessa funzione per più display o utilizzare funzioni diverse per più display. Dipende da cosa si sceglie.

i Nota

`lv_display_flush_ready(display1)` deve essere chiamata al termine del flushing per informare LVGL che il buffer è nuovamente disponibile per il rendering di nuovo contenuto.

LVGL potrebbe visualizzare lo schermo in più blocchi e quindi chiamare la funzione di Callback Flush più volte. Per verificare se la chiamata corrente riguarda l'ultimo blocco in fase di rendering, si usa `lv_display_flush_is_last(display1)`.

Callback Flush-Wait

Utilizzando `lv_display_flush_ready()`, LVGL normalmente esegue un ciclo in attesa del flush.

Tuttavia, con l'aiuto di `lv_display_set_flush_wait_cb()`, è possibile impostare una callback di attesa personalizzata per il flush. Questa callback può utilizzare un semaforo, un mutex o qualsiasi altra cosa per ottimizzare l'attesa del completamento del flush. La callback non deve necessariamente chiamare `lv_display_flush_ready()`, poiché il chiamante se ne occupa (cancellando il flag `flushing` del display) al termine della callback.

Tuttavia, se una Callback Flush-Wait non è impostata, LVGL presume che `lv_display_flush_ready()` venga chiamata al termine del flush.

API

`lv_display.h`

`lv_display_private.h`

7.1.3 Layer dello Screen

Quando viene creato un oggetto `lv_display_t`, vengono creati e associati a esso 4 *Screen* permanenti che facilitano il "layering" [sovraposizione].

1. Layer Inferiore (sotto lo Schermo Attivo, trasparente, non scorrevole, ma cliccabile)
2. *Lo Screen Attivo*
3. Layer Superiore (sopra lo Schermo Attivo, trasparente e non scorrevole né cliccabile)
4. Layer di Sistema (sopra il Layer Superiore, trasparente e non scorrevole né cliccabile)

1, 3 e 4 sono indipendenti da *Lo Screen Attivo* e verranno visualizzati (se contengono qualcosa di visibile) indipendentemente da quale schermo sia *Lo Screen Attivo*.

i Nota

Affinché il layer inferiore sia visibile, lo sfondo dello Schermo Attivo deve essere almeno parzialmente, se non completamente, trasparente.

È possibile ottenere puntatori a ciascuna di queste schermate sul *Display di Default* utilizzando (rispettivamente):

- `lv_screen_active()`,
- `lv_layer_top()`,
- `lv_layer_sys()` e
- `lv_layer_bottom()`.

È possibile ottenere puntatori a ciascuna di queste schermate su un display specificato utilizzando (rispettivamente):

- `lv_display_get_screen_active(disp)`,
- `lv_display_get_layer_top(disp)`,
- `lv_display_get_layer_sys(disp)` e
- `lv_display_get_layer_bottom(disp)`.

Per impostare uno Schermo creato come *Lo Screen Attivo*, si chiama `lv_screen_load()` o `lv_screen_load_anim()`.

Chiamando `lv_display_get_screen_loading(disp)` verrà restituito lo screen in fase di caricamento. Uno Screen è considerato "in fase di caricamento" fino al termine dell'animazione di caricamento. Se non viene caricato alcuno Screen, questa funzione restituirà NULL.

Layer Superiori e di Sistema

LVGL utilizza il Layer Superiore e quello di Sistema per garantire che alcuni tra *Tutti i Widget* siano *sempre* sopra gli altri layer.

È possibile aggiungere liberamente "finestre pop-up" al *Layer Superiore*. Il Layer Superiore è stato concepito per creare Widget visibili su tutti gli Screen visualizzati su un Display. Il *Layer di Sistema*, invece, è destinato a funzioni a livello di sistema (ad esempio, il cursore del mouse verrà posizionato lì con `lv_indev_set_cursor()`).

Questi layer funzionano come qualsiasi altro widget, ovvero hanno stili e al loro interno è possibile creare qualsiasi tipo di Widget.

i Nota

Mentre il Layer Superiore e quello di Sistema vengono creati dal rispettivo *Display* (`lv_display`) come non scorrevoli e non cliccabili, questi comportamenti possono essere sovrascritti come per qualsiasi altro Widget utilizzando rispettivamente `lv_obj_set_scrollbar_mode(scr1, LV_SCROLLBAR_MODE_xxx)` e `lv_obj_add_flag(scr1, LV_OBJ_FLAG_CLICKABLE)`.

Se il flag `LV_OBJ_FLAG_CLICKABLE` è impostato sul Layer Superiore, questo assorbirà tutti i clic dell'utente e fungerà da Widget modale.

```
lv_obj_add_flag(lv_layer_top(), LV_OBJ_FLAG_CLICKABLE);
```

Layer Inferiore

Similmente ai Layer Superiore e di Sistema, anche il Layer Inferiore ha le stesse dimensioni del Display, ma si trova sotto *Lo Screen Attivo*. È visibile solo se l'opacità dello sfondo della schermata attiva è < 255.

Ulteriori Informazioni

Screen Trasparenti.

API

lv_display.h

7.1.4 Formato Colore

Il formato colore di default del display è impostato in base a `LV_COLOR_DEPTH` (vedere `lv_conf.h`)

- `LV_COLOR_DEPTH` 32: XRGB8888 (4 byte/pixel)
- `LV_COLOR_DEPTH` 24: RGB888 (3 byte/pixel)
- `LV_COLOR_DEPTH` 16: RGB565 (2 byte/pixel)
- `LV_COLOR_DEPTH` 8: L8 (1 byte/pixel)
- `LV_COLOR_DEPTH` 1: I1 (1 bit/pixel) Supporto solo per buffer con mappatura orizzontale. [Display Monocromatici](#) per maggiori dettagli:

Il `color_format` può essere modificato con `lv_display_set_color_depth(display, LV_COLOR_FORMAT_...)`. Oltre al valore di default `LV_COLOR_FORMAT_ARGB8888`, è possibile utilizzare anche altri formati.

È molto importante che il buffer di disegno sia sufficientemente grande per il formato colore selezionato.

Scambio di Byte Endian

Nel caso del formato colore RGB565, potrebbe essere necessario scambiare i 2 byte perché la periferica SPI, I2C o la porta parallela a 8 bit li invia nell'ordine sbagliato.

La soluzione ideale è configurare l'hardware per gestire i dati a 16 bit con un ordine di byte diverso; tuttavia, se ciò non fosse possibile, si può chiamare `lv_draw_sw_rgb565_swap(buf, buf_size_in_px)` nella [Callback Flush](#) per scambiare i byte.

Volendo, è anche possibile scrivere una funzione personalizzata o utilizzare le istruzioni assembly per uno scambio di byte il più veloce possibile.

Si noti che non si tratta di scambiare i canali Rosso e Blu, ma di convertire

RRRRR GGG | GGG BBBBB

in

GGG BBBBB | RRRRR GGG.

Display Monocromatici

LVGL supporta il rendering diretto in un formato a 1 bit per i display monocromatici. Per abilitarlo, impostare `LV_COLOR_DEPTH` 1 o utilizzare `lv_display_set_color_format(display, LV_COLOR_FORMAT_I1)`.

Il formato `LV_COLOR_FORMAT_I1` presuppone che i byte siano mappati su righe (ovvero, i bit di un byte siano scritti uno accanto all'altro). L'ordine dei bit è MSB prima, ovvero:

bits	MSB	LSB
	7 6 5 4 3 2 1 0	

sono rappresentati sul display come:

pixels	0 1 2 3 4 5 6 7
	Left Right

Assicurarsi che il controller LCD sia configurato di conseguenza.

Internamente, LVGL arrotonda le aree ridisegnate ai limiti dei byte. Pertanto, le aree aggiornate:

- inizieranno con una coordinata Nx8, e
- terminano con una coordinata Nx8 - 1.

Quando si impostano i buffer per il rendering (`lv_display_set_buffers()`), aumentare la dimensione del buffer di 8 byte. Questo è necessario perché LVGL riserva 2 x 4 byte nel buffer, poiché si presume che questi vengano utilizzati come palette.

Per saltare la palette, includere la seguente riga nella funzione *Callback Flush*: `px_map += 8`.

Come di consueto, i display monocromatici supportano anche le modalità di rendering parziale, completo e diretto. In modalità completo e diretto, la dimensione del buffer dovrebbe essere sufficientemente grande per l'intero schermo, ovvero (`horizontal_resolution x vertical_resolution / 8`) + 8 byte. Poiché LVGL non può gestire larghezze frazionarie, assicurarsi di arrotondare la risoluzione orizzontale a 8 bit (ad esempio da 90 a 96).

La funzione `lv_draw_sw_il_convert_to_vtiled()` viene utilizzata per convertire un buffer di disegno in formato colore I1 da un layout di buffer per riga (htiled) a uno per colonna (vtiled). Questa conversione è necessaria per alcuni controller video che richiedono una mappatura del buffer di disegno diversa. La funzione presuppone che la larghezza e l'altezza del buffer siano arrotondate a un multiplo di 8. L'ordine dei bit del buffer vtiled risultante può essere specificato utilizzando il parametro `bit_order_lsb`.

Per maggiori dettagli, fare riferimento all'implementazione di `lv_draw_sw_il_convert_to_vtiled()` in `src/draw/sw/lv_draw_sw.c`.

Per garantire che le aree ridisegnate inizino e finiscano sui limiti dei byte, è possibile aggiungere una callback di tipo "rounder" al driver video. Questa callback arrotonda la larghezza e l'altezza al multiplo di 8 più vicino.

Ecco un esempio di come implementare e impostare una callback di arrotondamento:

```
static void my_rounder_cb(lv_event_t *e)
{
    lv_area_t *area = lv_event_get_param(e);

    /* Round the height to the nearest multiple of 8 */
    area->y1 = (area->y1 & ~0x7);
    area->y2 = (area->y2 | 0x7);
}

lv_display_add_event_cb(display, my_rounder_cb, LV_EVENT_INVALIDATE_AREA, display);
```

In questo esempio, la funzione `my_rounder_cb` arrotonda le coordinate dell'area ridisegnata al multiplo di 8 più vicino. Le coordinate x1 e y1 vengono arrotondate per difetto, mentre x2 e y2 vengono arrotondate per eccesso. Questo garantisce che la larghezza e l'altezza dell'area ridisegnata siano sempre multipli di 8.

Screen Trasparenti

Solitamente, l'opacità dello schermo è `LV_OPA_COVER` per fornire uno sfondo uniforme per i suoi elementi secondari. In caso contrario (opacità < 100%), il `bottom_layer` del display sarà visibile. Se anche l'opacità del layer inferiore non è `LV_OPA_COVER`, LVGL non avrà uno sfondo uniforme da disegnare.

Questa configurazione (Schermo trasparente) potrebbe essere utile per creare, ad esempio, menù OSD (On-Screen Display) in cui un video viene riprodotto su un layer hardware diverso del pannello del display e un menù è sovrapposto a un layer superiore.

Per visualizzare correttamente un'interfaccia utente su uno schermo trasparente, il formato colore del display deve essere impostato su uno con un canale alfa (ad esempio `LV_COLOR_FORMAT_ARGB8888`).

In sintesi, per abilitare schermi e display trasparenti per interfacce utente simili a menù OSD:

- Impostare `bg_opa` dello schermo su trasparente: `lv_obj_set_style_bg_opa(lv_screen_active(), LV_OPA_TRANSP, LV_PART_MAIN)`
- Impostare `bg_opa` del layer inferiore su trasparente: `lv_obj_set_style_bg_opa(lv_layer_bottom(), LV_OPA_TRANSP, LV_PART_MAIN)`
- Impostare un formato colore con canale alfa. Es. `lv_display_set_color_format(disp, LV_COLOR_FORMAT_ARGB8888)`

API

lv_obj_style_gen.h
lv_display.h
lv_draw_sw_utils.h
lv_conf_internal.h

7.1.5 Refreshing

Comportamento del Refresh di Default

Normalmente le aree sporche (ovvero non valide) vengono controllate e ridisegnate ogni *LV_DEF_REFR_PERIOD* millisecondi (impostati in *lv_conf.h*). Ciò avviene a seguito di un *Timer* (*lv_timer*) di aggiornamento creato al momento della creazione del display ed eseguito a quell'intervallo.

Disaccoppiamento del Timer di Aggiornamento del Display

Tuttavia, in alcuni casi potrebbe essere necessario un maggiore controllo su quando avviene l'aggiornamento del display, ad esempio:

- per sincronizzare il rendering con VSYNC o il segnale TE [Tearing Effect];
- per impostare il refresh del display immediatamente dopo un singolo aggiornamento dello schermo di tutti i widget da aggiornare (ovvero, aggiornarli solo una volta immediatamente prima del refresh del display per ridurre il sovraccarico della CPU).

È possibile farlo nel modo seguente:

```
/* Delete original display refresh timer */
lv_display_delete_refr_timer(display);

/* Call this to refresh dirty (changed) areas of the display. */
lv_display_refr_timer(NULL);
```

Se si dispone di più display, si chiama *lv_display_set_default(display1)* per selezionare il display da aggiornare prima di *lv_display_refr_timer(NULL)*.

Nota

lv_timer_handler() e *lv_display_refr_timer()* non devono essere eseguiti contemporaneamente.

Se il monitoraggio delle prestazioni è abilitato, il valore di *LV_DEF_REFR_PERIOD* deve essere impostato in modo che corrisponda al periodo di aggiornamento del display per garantire che i risultati statistici siano corretti.

Forzare un Refresh

Normalmente le aree invalidate (contrassegnate per il ridisegno) vengono renderizzate in *lv_timer_handler()* ogni *LV_DEF_REFR_PERIOD* millisecondi. Tuttavia, utilizzando *lv_refr_now(display)* è possibile indicare a LVGL di ridisegnare immediatamente le aree non valide. L'aggiornamento avverrà in *lv_refr_now()*, il che potrebbe richiedere più tempo.

Il parametro di *lv_refr_now()* è un puntatore al display da aggiornare. Se viene passato NULL, tutti i display con timer di aggiornamento attivi verranno aggiornati.

API

lv_refr.h
lv_display.h
lv_api_map_v9_1.h
lv_conf_internal.h

lv_timer.h

7.1.6 Eventi del Display

`lv_display_add_event_cb(disp, event_cb, LV_EVENT_..., user_data)` aggiunge un gestore eventi a un display.

Se è stato aggiunto `user_data` al Display, lo si può recuperare in un evento come questo:

```
lv_display_t * display;
my_type_t * my_user_data;
display = (lv_display_t *)lv_event_get_current_target(e);
my_user_data = lv_display_get_user_data(display);
```

I seguenti eventi vengono inviati per gli oggetti Display (`lv_display_t`):

- **`LV_EVENT_INVALIDATE_AREA`** Un'area viene invalidata (contrassegnata per il ridisegno). `lv_event_get_param(e)` restituisce un puntatore a un oggetto `lv_area_t` con le coordinate dell'area da invalidare. L'area può essere modificata liberamente se necessario per adattarla a un requisito specifico del display. Di solito necessario con i display monocromatici per invalidare $N \times 8$ righe o colonne in un unico passaggio.
- **`LV_EVENT_RESOLUTION_CHANGED`**: Inviato quando la risoluzione cambia a causa di `lv_display_set_resolution()` o di `lv_display_set_rotation()`.
- **`LV_EVENT_COLOR_FORMAT_CHANGED`**: Inviato come risultato di qualsiasi chiamata a `lv_display_set_color_format()`.
- **`LV_EVENT_REFR_REQUEST`**: Inviato quando si è verificato un evento che richiede un ridisegno.
- **`LV_EVENT_REFR_START`**: Inviato prima dell'inizio di un ciclo di aggiornamento. Inviato anche se non c'è nulla da ridisegnare.
- **`LV_EVENT_REFR_READY`**: Inviato al completamento dell'aggiornamento (dopo il rendering e la chiamata a *Callback Flush*). Inviato anche se non si è verificato alcun ridisegno.
- **`LV_EVENT_RENDER_START`**: Inviato appena prima dell'inizio del rendering.
- **`LV_EVENT_RENDER_READY`**: Inviato dopo il completamento del rendering (prima di chiamare *Callback Flush*)
- **`LV_EVENT_FLUSH_START`**: Inviato prima della chiamata a *Callback Flush*.
- **`LV_EVENT_FLUSH_FINISH`**: Inviato dopo il ritorno della chiamata a *Callback Flush*.
- **`LV_EVENT_FLUSH_WAIT_START`**: Inviato all'inizio della chiamata interna a `wait_for_flushing()` -- si verifica indipendentemente dal fatto che si verifichi effettivamente un'attesa. La chiamata termina immediatamente se `disp->flushing == 0`.
- **`LV_EVENT_FLUSH_WAIT_FINISH`**: Inviato quando la chiamata a `wait_for_flushing()` sta per terminare, indipendentemente dal fatto che si sia verificata un'attesa effettiva.

API

lv_display.h

7.1.7 Modifica della Risoluzione

Per impostare la risoluzione del display dopo la creazione, si usa `lv_display_set_resolution(display, hor_res, ver_res)`

Non è obbligatorio utilizzare l'intero display per LVGL, tuttavia in alcuni casi la risoluzione fisica è importante. Ad esempio, il touchpad continua a visualizzare l'intera risoluzione e i valori devono essere convertiti nell'area di visualizzazione LVGL attiva. Quindi la risoluzione fisica e l'offset dell'area attiva possono essere impostati con `lv_display_set_physical_resolution(disp, hor_res, ver_res)` e `lv_display_set_offset(disp, x, y)`

API

lv_display.h

7.1.8 Misurazione dell'Inattività

Il tempo di inattività di un utente viene misurato e memorizzato con ciascun oggetto `lv_display_t`. Ogni utilizzo di un *Input Device* conta come attività. Per ottenere il tempo trascorso dall'ultima attività, si usa `lv_display_get_inactivity_time(display1)`. Se viene passato NULL, verrà restituito il tempo di inattività più basso tra tutti i display (in questo caso NULL non indica *Display di Default*).

È possibile avviare manualmente un'attività utilizzando `lv_display_trigger_activity(display1)`. Se `display1` è NULL, verrà utilizzato *Display di Default* (**non tutti i display**).

Ulteriori Informazioni

- `lv_port_disp_template.c` per un template per il proprio driver.
- *Drawing* per saperne di più sul funzionamento del rendering in LVGL.

API

lv_display.h

7.1.9 Rotazione

LVGL supporta la rotazione del display con incrementi di 90 gradi.

L'orientamento del display può essere modificato con `lv_display_set_rotation(disp, LV_DISPLAY_ROTATION_xxx)` dove xxx è 0, 90, 180 o 270. Questo invertirà internamente la risoluzione orizzontale e verticale in base al grado impostato; tuttavia, non eseguirà la rotazione effettiva. Quando si modifica la rotazione, viene generato l'evento `LV_EVENT_SIZE_CHANGED` per consentire la riconfigurazione hardware. Se il pannello di visualizzazione e/o i relativi chip driver non supportano la rotazione, è possibile utilizzare `lv_draw_sw_rotate()` per ruotare il buffer nella funzione *Callback Flush*.

`lv_display_rotate_area(display, &area)` ruota l'area renderizzata in base alle impostazioni di rotazione correnti del display.

Si noti che in `LV_DISPLAY_RENDER_MODE_DIRECT` le piccole aree modificate vengono renderizzate direttamente nel frame buffer, quindi non possono essere ruotate in un secondo momento. Pertanto, in modalità diretta è possibile ruotare solo l'intero frame buffer.

Nel caso di `LV_DISPLAY_RENDER_MODE_PARTIAL` le piccole aree renderizzate possono essere ruotate autonomamente prima di essere riversate nel frame buffer.

`LV_DISPLAY_RENDER_MODE_FULL` può funzionare con la rotazione se i buffer in cui viene eseguito il rendering sono diversi dai buffer in cui viene eseguito il rendering nella callback flush e i buffer in cui viene eseguito il rendering non hanno un requisito di passo.

Di seguito è riportato un esempio di rotazione quando la modalità di rendering è `LV_DISPLAY_RENDER_MODE_PARTIAL` e l'immagine ruotata deve essere inviata a un **display controller**.

```
/*Rotate a partially rendered area to another buffer and send it*/
void flush_cb(lv_display_t * disp, const lv_area_t * area, uint8_t * px_map)
{
    lv_display_rotation_t rotation = lv_display_get_rotation(disp);
    lv_area_t rotated_area;
    if(rotation != LV_DISPLAY_ROTATION_0) {
        lv_color_format_t cf = lv_display_get_color_format(disp);
        /*Calculate the position of the rotated area*/
        rotated_area = *area;
        lv_display_rotate_area(disp, &rotated_area);
        /*Calculate the source stride (bytes in a line) from the width of the area*/
        uint32_t src_stride = lv_draw_buf_width_to_stride(lv_area_get_width(area), cf);
        /*Calculate the stride of the destination (rotated) area too*/
        uint32_t dest_stride = lv_draw_buf_width_to_stride(lv_area_get_width(&rotated_area), cf);
        /*Have a buffer to store the rotated area and perform the rotation*/
        static uint8_t rotated_buf[500*1014];
        int32_t src_w = lv_area_get_width(area);
        int32_t src_h = lv_area_get_height(area);
        lv_draw_sw_rotate(px_map, rotated_buf, src_w, src_h, src_stride, dest_stride, rotation, cf);
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

/*Use the rotated area and rotated buffer from now on*/
area = &rotated_area;
px_map = rotated_buf;
}
my_set_window(area->x1, area->y1, area->x2, area->y2);
my_send_colors(px_map);
}

```

Di seguito è riportato un esempio di rotazione quando la modalità di rendering è `LV_DISPLAY_RENDER_MODE_PARTIAL` e l'immagine può essere ruotata direttamente in un **frame buffer della periferica LCD**.

```

/*Rotate a partially rendered area to the frame buffer*/
void flush_cb(lv_display_t * disp, const lv_area_t * area, uint8_t * px_map)
{
    lv_color_format_t cf = lv_display_get_color_format(disp);
    uint32_t px_size = lv_color_format_get_size(cf);
    /*Calculate the position of the rotated area*/
    lv_area_t rotated_area = *area;
    lv_display_rotate_area(disp, &rotated_area);
    /*Calculate the properties of the source buffer*/
    int32_t src_w = lv_area_get_width(area);
    int32_t src_h = lv_area_get_height(area);
    uint32_t src_stride = lv_draw_buf_width_to_stride(src_w, cf);
    /*Calculate the properties of the frame buffer*/
    int32_t fb_stride = lv_draw_buf_width_to_stride(disp->hor_res, cf);
    uint8_t * fb_start = my_fb_address;
    fb_start += rotated_area.y1 * fb_stride + rotated_area.x1 * px_size;
    lv_display_rotation_t rotation = lv_display_get_rotation(disp);
    if(rotation == LV_DISPLAY_ROTATION_0) {
        int32_t y;
        for(y = area->y1; y <= area->y2; y++) {
            lv_memcpy(fb_start, px_map, src_stride);
            px_map += src_stride;
            fb_start += fb_stride;
        }
    } else {
        lv_draw_sw_rotate(px_map, fb_start, src_w, src_h, src_stride, fb_stride, rotation, cf);
    }
}

```

API

lv_display.h

7.1.10 Vincoli sull'Area Ridisegnata

Alcuni controller di display hanno requisiti specifici per l'area della finestra in cui può essere inviata l'immagine renderizzata (ad esempio, x1 deve essere pari e x2 deve essere dispari).

Nel caso di display monocromatici, x1 deve essere Nx8 e x2 deve essere Nx8 - 1. (Se il display utilizza `LV_COLOR_FORMAT_I1`, LVGL applica automaticamente questo arrotondamento. Vedere *Display Monocromatici*.)

La dimensione dell'area invalidata (ridisegnata) può essere controllata come segue:

```

void rounder_event_cb(lv_event_t * e)
{
    lv_area_t * a = lv_event_get_invalidate_area(e);

    a->x1 = a->x1 & (~0x1); /* Ensure x1 is even */
    a->x2 = a->x2 | 0x1; /* Ensure x2 is odd */
}

...
lv_display_add_event_cb(disp, rounder_event_cb, LV_EVENT_INVALIDATE_AREA, NULL);

```

API

lv_display.h

7.1.11 Rendering a Riquadri

Quando sono disponibili più core della CPU e un'area estesa deve essere ridisegnata, LVGL deve identificare aree indipendenti che possano essere renderizzate in parallelo.

Ad esempio, se ci sono 4 core della CPU, un core può disegnare lo sfondo dello schermo mentre gli altri 3 devono attendere il completamento. Se ci sono 2 pulsanti sullo schermo, questi 2 pulsanti possono essere renderizzati in parallelo, ma 2 core rimarranno comunque inattivi.

A causa delle dipendenze tra le diverse aree, i core della CPU non possono sempre essere utilizzati completamente.

Per risolvere questo problema, LVGL può suddividere le aree estese che devono essere aggiornate in "tile" [riquadri] più piccoli. Questi riquadri sono indipendenti, rendendo più facile trovare aree che possono essere renderizzate contemporaneamente.

In particolare, se ci sono 4 riquadri e 4 core, ci sarà sempre un'area indipendente per ogni core all'interno di uno dei riquadri.

Il numero massimo di tile può essere impostato utilizzando la funzione `lv_display_set_tile_cnt(disp, cnt)`. Il valore di default è `LV_DRAW_SW_DRAW_UNIT_CNT` (o 1 se il rendering software non è abilitato).

Le aree di piccole dimensioni non vengono ulteriormente suddivise in "tile" più piccoli perché il sovraccarico di 4 core supererebbe i vantaggi.

La dimensione ideale del tile è calcolata come `ideal_tile_size = draw_buf_size / tile_cnt`. Ad esempio, in modalità `LV_DISPLAY_RENDER_MODE_DIRECT` su uno schermo 800x480, il buffer di visualizzazione è 800x480 = 375k pixel. Se ci sono 4 tile, la dimensione ideale del tile è di circa 93k pixel. In base a ciò, l'utilizzo dei core è il seguente:

- 30k pixel: 1 core
- 90k pixel: 1 core
- 95k pixel: 2 core (oltre i 93k pixel, vengono utilizzati 2 core)
- 150k pixel: 2 core
- 200k pixel: 3 core (oltre i 186k pixel, vengono utilizzati 3 core)
- 300k pixel: 4 core (oltre i 279k pixel, vengono utilizzati 4 core)
- 375k pixel: 4 core

In `LV_DISPLAY_RENDER_MODE_DIRECT`, il buffer di disegno, delle dimensioni dello schermo, viene diviso per il numero di tile per determinarne le dimensioni ideali. Se venissero aggiornate aree più piccole, potrebbe essere utilizzato un numero inferiore di core.

In `LV_DISPLAY_RENDER_MODE_FULL`, il numero massimo di riquadri viene sempre creato quando l'intero schermo viene aggiornato.

In `LV_DISPLAY_RENDER_MODE_PARTIAL`, il buffer parziale è suddiviso in riquadri. Ad esempio, se il buffer di disegno è 1/10 delle dimensioni dello schermo e ci sono 2 riquadri, verranno renderizzati contemporaneamente 1/20 + 1/20 dell'area dello schermo.

Il rendering a riquadri influisce solo sul processo di rendering e la `Callback Flush` viene chiamata una volta per ogni area invalidata. Pertanto, il tiling non è visibile dal punto di vista del flushing.

API

`lv_display.h`

7.1.12 Estensione/Combinazione di Display

Mirroring di un Display

Per il mirroring dell'immagine di un display su un altro display, non è necessario utilizzare il supporto multi-display. È sufficiente trasferire il buffer ricevuto nella `Callback Flush` del primo display anche all'altro display.

Suddivisione di un'Immagine

È possibile creare un display virtuale più grande da un array di display più piccoli. È possibile crearlo:

1. impostando la risoluzione dei display alla risoluzione del display più grande;
2. in `Callback Flush`, troncando e modificando il parametro `area` per ciascun display; e
3. inviando il contenuto del buffer a ciascun display reale con l'area troncata.

7.1.13 API

lv_display.h

7.2 Dispositivi di input (lv_indev)

7.2.1 Dispositivi di input

Cos'è un indev?

I dispositivi di input (o indev) si riferiscono a tutti i moduli e le funzionalità relative alla gestione dell'input utente.

Tra questi rientrano touch, mouse, cursore, tastiera, pulsanti esterni, encoder, navigazione e molti altri.

I dispositivi di input possono attivare un'ampia varietà di eventi come pressione, rilascio, clic, doppio o triplo clic, scrolling, passaggio del mouse e altri. Per visualizzare l'elenco completo, consultare [Eventi del Dispositivo di Input](#).

I dispositivi di input possono anche modificare lo [Stato](#) dei widget. Alcune delle modifiche dello stato sono specifiche del tipo indev. Tuttavia, la più elementare è che i widget premuti avranno automaticamente [LV_STATE_PRESSED](#). Consultare [Stili](#) per maggiori dettagli.

Creazione di un Device di Input

Per creare un dispositivo di input, sono necessarie solo due cose:

1. un tipo: puntatore, tastiera, ecc.
2. una callback di lettura: per leggere il punto del touch corrente o il tasto premuto

```
lv_indev_t * indev = lv_indev_create();           /* Create input device */
lv_indev_set_type(indev, LV_INDEV_TYPE_...);       /* Set the device type */
lv_indev_set_read_cb(indev, my_input_read);        /* Set the read callback */
```

Il membro `type` può essere:

- [LV_INDEV_TYPE_POINTER](#): touchpad o mouse
- [LV_INDEV_TYPE_KEYPAD](#): tastiera o tastierino numerico
- [LV_INDEV_TYPE_ENCODER](#): encoder con opzioni di rotazione a sinistra/destra e di pressione
- [LV_INDEV_TYPE_BUTTON](#): pulsanti esterni che premono virtualmente lo schermo

`my_input_read` è un puntatore a funzione che verrà chiamato periodicamente per segnalare lo stato corrente del dispositivo di input a LVGL.

I dispositivi di input sono collegati a un [Display](#), quindi si deve creare un display prima di creare gli indev.

Se si hanno più display, si deve impostare il display di default su quello a cui è "collegato" il dispositivo di input prima di effettuare le chiamate di cui sopra.

Lettura dei Dati

Polling

Per default, viene creato un [Timer](#) (`lv_timer`) che chiama periodicamente `read_cb`. Il periodo di lettura di default è impostato da [LV_DEF_REFR_PERIOD](#) in `lv_conf.h`.

`lv_indev_get_read_timer()` restituisce il timer di lettura in modo che le funzioni relative al [Timer](#) (`lv_timer`) possano essere chiamate su di esso per regolarne il comportamento.

Lettura Bufferizzata

Per default, LVGL chiama `read_cb` periodicamente. A causa di questo polling intermittente, è possibile che alcuni eventi utente vengano persi.

Per risolvere questo problema, è possibile scrivere un driver event-driven per il dispositivo di input che memorizzi i dati misurati nel buffer. In `read_cb`, è possibile segnalare i dati memorizzati nel buffer invece di leggere direttamente

il dispositivo di input. Impostando il flag `data->continue_reading`, LVGL indicherà che ci sono più dati da leggere e che dovrebbe richiamare `read_cb`.

Se il driver è in grado di fornire timestamp precisi per gli eventi memorizzati nel buffer, può sovrascrivere `data->timestamp`. Per default, questo viene inizializzato a `lv_tick_get()` subito prima di richiamare `read_cb`.

Modalità Event-Driven

Normalmente, un dispositivo di input viene letto ogni `LV_DEF_REFR_PERIOD` millisecondi (impostato in `lv_conf.h`). Tuttavia, in alcuni casi, potrebbe essere necessario un maggiore controllo su quando leggere il dispositivo di input. Ad esempio, potrebbe essere necessario leggerlo interrogando un descrittore di file.

È possibile farlo in questo modo:

```
/* Update the input device's running mode to LV_INDEV_MODE_EVENT */
lv_indev_set_mode(indev, LV_INDEV_MODE_EVENT);

...
/* Call this anywhere you want to read the input device */
lv_indev_read(indev);
```

Nota

`lv_indev_read()`, `lv_timer_handler()` e `_lv_display_refr_timer()` non possono essere eseguiti contemporaneamente.

Nota

Per i dispositivi in modalità event-driven, `data->continue_reading` viene ignorato.

Mettere in Pausa il Timer di Indev

Non è sempre possibile effettuare la lettura di un indev direttamente all'interno di un semplice "interrupt handler". In genere, un flag verrebbe impostato all'interno del gestore di interrupt, che verrebbe controllato e reimpostato all'interno della callback di lettura indev, dove la lettura verrebbe effettivamente eseguita. Questo funziona bene, ma la callback di lettura indev interroga costantemente un flag che potrebbe rimanere non impostato per lunghi periodi. Non possiamo usare *Modalità Event-Driven* perché `lv_indev_read()` non dovrebbe essere chiamato in un interrupt handler.

In questa situazione, è possibile utilizzare la callback di lettura indev basata sul timer come di consueto, ma mettere in pausa il timer indev se non si è verificato un interrupt per un po' di tempo. Riprendere un timer è in genere sicuro in un gestore di interrupt. È necessario prestare attenzione per evitare condizioni di "race".

```
volatile bool interrupt_occurred;
lv_timer_t * volatile indev_timer;

void interrupt_handler(void)
{
    interrupt_occurred = true;
    if(indev_timer) lv_timer_resume(indev_timer);
}

uint32_t last_interrupt_tick;

void my_input_read(lv_indev_t * indev, lv_indev_data_t * data)
{
    uint32_t tick_now = lv_tick_get();

    /* If no interrupt has happened in the past 100 ms, pause the indev timer */
    if(lv_tick_diff(tick_now, last_interrupt_tick) > 100) {
        lv_timer_pause(indev_timer);
    }

    if(interrupt_occurred) {
        interrupt_occurred = false;
        last_interrupt_tick = tick_now;

        /*
         * Ensure the timer is running in case an interrupt occurred
         * just after the timer was paused. Without this, a race condition
         * could leave the timer paused and input events would not be processed.
         */
        lv_timer_resume(indev_timer);
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```
/* Perform the reading */
/* ... */
}
```

```
/* In your setup code */
indev_timer = lv_indev_get_read_timer(indev);
```

Ulteriori Informazioni

- [lv_port_indev_template.c](#) per un template per il proprio driver di dispositivo di input.

7.2.2 Touchpad e Mouse

Panoramica

I dispositivi di input che possono cliccare su punti sul display appartengono alla categoria *Puntatore*. Questa categoria include qualsiasi tipo di touchpad o mouse.

I dispositivi di input puntatore possono:

- "scrollare" i widget
- inviare un'ampia varietà di *Eventi* (inclusi clic, pressione prolungata, doppio clic e molti altri)
- attivare e gestire *Gesture*

Esempio

Ecco un esempio di una semplice funzione di callback di lettura del dispositivo di input:

```
lv_indev_t * indev = lv_indev_create();
lv_indev_set_type(indev, LV_INDEV_TYPE_POINTER);
lv_indev_set_read_cb(indev, touch_read);

...

void touch_read(lv_indev_t * indev, lv_indev_data_t * data)
{
    if(my_touch_is_pressed()) {
        data->point.x = touchpad_x;
        data->point.y = touchpad_y;
        data->state = LV_INDEV_STATE_PRESSED;
    } else {
        data->state = LV_INDEV_STATE_RELEASED;
    }
}
```

Cursore

Anche i dispositivi di input puntatore possono avere un cursore.

```
LV_IMAGE_DECLARE(mouse_cursor_icon); /* Declare the image source. */
lv_obj_t * cursor_obj = lv_image_create(lv_screen_active()); /* Create image widget for cursor. */
lv_image_set_src(cursor_obj, &mouse_cursor_icon); /* Set image source. */
lv_indev_set_cursor(mouse_indev, cursor_obj); /* Connect image to input device. */
```

Quando il cursore viene aggiunto, verrà posizionato sul layer di sistema ([lv_layer_sys\(\)](#)) e posizionato sull'ultima coordinata letta.

Multi-touch

Il termine multi-touch può riferirsi a due cose:

1. Elaborazione di gesti, come swipe, pinch, ecc. Per questo, vedere le *Gesture*
2. Supporto di più punti di touch simultanei: è sufficiente creare più dispositivi di input. Ad esempio, per supportare 2 tocchi, creare 2 dispositivi di input puntatore. Utilizzare il primo punto di tocco dal driver touch nel primo indev e il secondo punto nel secondo indev.

Estensione dell'Area di Clic

Per default, i widget possono essere cliccati solo all'interno della loro area di "bounding" [delimitazione]. Tuttavia, soprattutto con i widget di piccole dimensioni, può essere utile ingrandire l'area "cliccabile" di un widget. Lo si può fare con `lv_obj_set_ext_click_area(widget, size)`.

Parametri

I tempi e i limiti dei dispositivi di input possono essere configurati tramite i seguenti campi di `lv_indev_t`:

- `lv_indev_set_long_press_time(indev, ms)`: Tempo richiesto per inviare `LV_EVENT_LONG_PRESSED` (in millisecondi)
- `lv_indev_set_long_press_repeat_time(indev, ms)`: Intervallo tra gli eventi `LV_EVENT_LONG_PRESSED_REPEAT` (in millisecondi)
- `lv_indev_set_scroll_limit(indev, pixels)`: Numero di pixel da spostare prima dello scrolling del widget
- `lv_indev_set_scroll_throw(indev, percent)`: Rallentamento dello scrolling (inerzia) in [%]. Un valore maggiore indica un rallentamento più rapido.

Modalità Crown

Una "Crown" [corona/rotellina] è un dispositivo rotante tipicamente presente sugli smartwatch.

Quando l'utente clicca in un punto qualsiasi e poi ruota la ghiera, l'ultimo widget cliccato verrà fatto scorrere oppure il suo valore verrà incrementato/decrementato (ad esempio, nel caso di uno slider).

Poiché questo comportamento è strettamente correlato all'ultimo widget cliccato, il supporto della corona è un'estensione del dispositivo di input del puntatore. È sufficiente impostare `data->enc_diff` sul numero di passi di rotazione e LVGL invierà automaticamente l'evento `LV_EVENT_ROTARY` o farà scorrere il widget in base al flag `editable` nella classe del widget. I widget non modificabili vengono fatti scorrere e, per quelli modificabili, l'evento viene inviato.

Per ottenere i passaggi in una callback di evento, si usa: `int32_t diff = lv_event_get_rotary_diff(e);`

La sensibilità della rotazione può essere regolata a due livelli:

1. Nel dispositivo di input utilizzando il campo `indev->rotary_sensitivity` (1/256 di unità)
2. Nel widget utilizzando la proprietà di stile `rotary_sensitivity` (1/256 di unità)

La differenza finale viene calcolata come:

```
diff_final = diff_in * (indev_sensitivity / 256) * (widget_sensitivity / 256);
```

Ad esempio, se sia la sensibilità `indev` che quella del widget sono impostate su 128 (0,5), la differenza di input verrà moltiplicata per 0,25. Il valore del widget verrà incrementato di tale importo o il widget verrà fatto scorrere di quel numero di pixel.

7.2.3 Keypad e Tastiera

Panoramica

Le tastiere complete con tutte le lettere o i "keypad" [tastierini] semplici con pochi pulsanti di navigazione appartengono alla categoria *Keypad*.

È possibile controllare completamente l'interfaccia utente senza touchpad o mouse, utilizzando solo un keypad. Funziona in modo simile al tasto *TAB* su un PC per selezionare un elemento in un'applicazione o in una pagina web.

Solo i widget aggiunti a un gruppo possono essere selezionati tramite tastiera. Per saperne di più, visitare [Gruppi](#).

Esempio

```
/*Create a group and add widgets to it so they can be selected by the keypad*/
lv_group_t * g = lv_group_create();

lv_indev_t * indev = lv_indev_create();
lv_indev_set_type(indev, LV_INDEV_TYPE_KEYPAD);
lv_indev_set_read_cb(indev, keyboard_read);
lv_indev_set_group(indev, g);

...

void keyboard_read(lv_indev_t * indev, lv_indev_data_t * data) {
    if(key_pressed()) {
        data->key = my_last_key(); /* Get the last pressed or released key */
        data->state = LV_INDEV_STATE_PRESSED;
    } else {
        data->state = LV_INDEV_STATE_RELEASED;
    }
}
```

Tasti

Ci sono alcuni tasti predefiniti che hanno un significato speciale:

- *LV_KEY_NEXT*: Sposta il focus sull'oggetto successivo
- *LV_KEY_PREV*: Sposta il focus sull'oggetto precedente
- *LV_KEY_ENTER*: Attiva gli eventi *LV_EVENT_PRESSED*, *LV_EVENT_CLICKED* o *LV_EVENT_LONG_PRESSED*
- *LV_KEY_UP*: Aumenta il valore o sposta verso l'alto
- *LV_KEY_DOWN*: Diminuisce il valore o sposta verso il basso
- *LV_KEY_RIGHT*: Aumenta il valore o sposta verso destra
- *LV_KEY_LEFT*: Diminuisce il valore o sposta verso sinistra
- *LV_KEY_ESC*: Chiude o esce (ad esempio, chiude una *Lista Drop-Down*)
- *LV_KEY_DEL*: Elimina (ad esempio, un carattere a destra in una *Text Area*)
- *LV_KEY_BACKSPACE*: Elimina (ad esempio, un carattere a sinistra in una *Text Area*)
- *LV_KEY_HOME*: Va all'inizio/top (ad esempio, in una *Text Area*)
- *LV_KEY_END*: Vai alla fine (ad esempio, in una *Text Area*)

I tasti speciali più importanti nella funzione `read_cb()` sono:

- *LV_KEY_NEXT*
- *LV_KEY_PREV*
- *LV_KEY_ENTER*
- *LV_KEY_UP*
- *LV_KEY_DOWN*
- *LV_KEY_LEFT*
- *LV_KEY_RIGHT*

Si devono tradurre alcuni dei tasti di lettura in questi tasti speciali per supportare la navigazione in un gruppo e interagire con i widget selezionati.

Rimappatura dei tasti

Alcune applicazioni richiedono l'assegnazione di pulsanti fisici a più eventi tasto a seconda del contesto: i tasti freccia possono essere utilizzati in una schermata per navigare tra i widget e per interagire con un widget in un'altra schermata. La di rimappatura dei tasti di indev può essere utilizzata per implementare questo comportamento.

Bisogna prima definire una callback di rimappatura, poi invocare `lv_indev_set_key_remap_cb()` per abilitarla.

```
lv_key_t remap_left_right_into_prev_next(lv_indev_t * indev, lv_key_t key)
{
    LV_UNUSED(indev);

    switch (key) {
    case LV_KEY_LEFT:
        return LV_KEY_PREV;
    case LV_KEY_RIGHT:
        return LV_KEY_NEXT;
    default:
        return key;
    }
}
```

```
/* in your setup code */
lv_indev_set_key_remap_cb(indev, remap_left_right_into_prev_next);

/* later, to disable remapping */
lv_indev_set_key_remap_cb(indev, NULL);
```

i Nota

La callback di rimappatura viene invocata solo per i dispositivi di input tramite tastiera. Non rimapperà i tasti di altri tipi di dispositivi di input.

7.2.4 Encoder

Panoramica

Un encoder è un dispositivo con una manopola girevole che può essere premuta e che indica al dispositivo host:

- quanto è stata ruotata la manopola
- in quale direzione
- e lo stato del pulsante (premuto o rilasciato)

Utilizzando i *Gruppi*, anche le interfacce utente complesse si possono navigare con un singolo encoder.

In breve, i dispositivi di input encoder funzionano in questo modo:

1. Ruotando l'encoder, è possibile mettere il focus sull'oggetto successivo o precedente.
2. Quando si preme e si rilascia l'encoder su un semplice oggetto "non modificabile" (come un pulsante), questo verrà cliccato.
3. Se si preme l'encoder su un oggetto "modificabile" complesso (come una lista, una finestra di messaggio, ecc.), il widget entrerà in *Edit mode*, consentendo di navigare all'interno dell'oggetto ruotando l'encoder.
4. Per uscire dalla modalità di modifica, premere a lungo il pulsante.

Questo significa anche che l'encoder porrà il focus e modificherà solo i widget aggiunti a un *gruppo*.

Esempio

Questo è un esempio di inizializzazione di un encoder:

```
/*Create a group and add widgets to it so they can be selected by the encoder*/
lv_group_t * g = lv_group_create();

lv_indev_t * indev = lv_indev_create();
lv_indev_set_type(indev, LV_INDEV_TYPE_ENCODER);
lv_indev_set_read_cb(indev, encoder_read);

...

void encoder_read(lv_indev_t * indev, lv_indev_data_t * data){
    data->enc_diff = enc_get_new_moves(); /* Negative for left, positive for right turns, 0 for no change */

    if(enc_pressed()) data->state = LV_INDEV_STATE_PRESSED;
    else data->state = LV_INDEV_STATE_RELEASED;
}
```

Stili

Quando un widget riceve il focus cliccandoci sopra tramite touchpad o navigandoci sopra con un encoder o una tastiera, entra nello stato `LV_STATE_FOCUSED`.

Se un widget passa in "edit mode", entra negli stati `LV_STATE_FOCUSED` | `LV_STATE_EDITED`, quindi verranno visualizzate tutte le proprietà di stile assegnate a questi stati.

Pulsanti come Encoder

Oltre al comportamento standard dell'encoder, è possibile utilizzare la sua logica anche per navigare (focus) e modificare i widget tramite i pulsanti. Questo è particolarmente utile se si hanno a disposizione solo pochi pulsanti o per utilizzarne altri oltre alla rotellina dell'encoder.

È necessario disporre di 3 pulsanti:

- `LV_KEY_ENTER`: simula la pressione del pulsante dell'encoder.
- `LV_KEY_LEFT`: simula la rotazione dell'encoder verso sinistra.
- `LV_KEY_RIGHT`: simula la rotazione dell'encoder verso destra.
- Gli altri tasti verranno passati al widget selezionato così come sono.

Se si tengono premuti i tasti, verrà simulato il movimento dell'encoder con il periodo specificato in `indev.long_press_repeat_time`.

In questo caso, in `read_cb`, settare `data->key` sul tasto premuto.

```
/*Create a group and add widgets to it so they can be selected by the encoder*/
lv_group_t * g = lv_group_create();

lv_indev_t * indev = lv_indev_create();
lv_indev_set_type(indev, LV_INDEV_TYPE_ENCODER);
lv_indev_set_read_cb(indev, encoder_with_keys_read);
lv_indev_set_group(indev, g);

...

void encoder_with_keys_read(lv_indev_t * indev, lv_indev_data_t * data){
    if(key_pressed()){
        /* Get the last pressed or released key
         * use LV_KEY_ENTER for encoder press */
        data->key = my_last_key();
        data->state = LV_INDEV_STATE_PRESSED;
    } else {
        data->state = LV_INDEV_STATE_RELEASED;
    }
}
```

7.2.5 Pulsante Hardware

Panoramica

Un *Hardware Button* è un pulsante esterno solitamente posizionato accanto allo schermo e assegnato a coordinate specifiche sullo schermo.. Premendo un pulsante, si simula una pressione sulla coordinata assegnata, in modo simile a un touchpad.

Per assegnare i "Pulsanti Hardware" alle coordinate, si usa `lv_indev_set_button_points(my_indev, points_array)`.

`points_array` dovrebbe apparire così:

```
static const lv_point_t points_array[] = {
    {12,30}, /* First button is assigned to x=12; y=30 */
    {60,90} /* Second button is assigned to x=60; y=90 */
};
```

Importante

`points_array` non deve uscire dallo scope. Dichiaralo come variabile globale o come variabile statica all'interno di una funzione.

L'indice del pulsante premuto deve essere impostato in `data->btn_id` in `read_cb`.

`data->state` deve essere impostato anche su `LV_INDEV_STATE_PRESSED` o `LV_INDEV_STATE_RELEASED`.

Esempio

Ecco un esempio completo:

```
static const lv_point_t points_array[] = {
    {12,30}, /* First button is assigned to x=12; y=30 */
    {60,90} /* Second button is assigned to x=60; y=90 */
};

lv_indev_t * indev = lv_indev_create();
lv_indev_set_type(indev, LV_INDEV_TYPE_BUTTON);
lv_indev_set_button_points(indev, points_array);
lv_indev_set_read_cb(indev, button_read);

...

void button_read(lv_indev_t * indev, lv_indev_data_t * data)
{
    /* Get the ID (0,1,2...) of the pressed button.
     * Let's say it returns -1 if no button was pressed */
    int btn_pr = my_btn_read();

    /* Is there a button press? */
    if(btn_pr >= 0) {
        data->btn_id = btn_pr;           /* Save the ID of the pressed button */
        data->state = LV_INDEV_STATE_PRESSED; /* Set the pressed state */
    } else {
        data->state = LV_INDEV_STATE_RELEASED; /* Set the released state */
    }
}
```

7.2.6 Gruppi

Panoramica

I gruppi sono "gruppi di widget" che possono ottenere il focus dai dispositivi di input *Keypad* o *Encoder*.

Innanzitutto, è necessario creare un gruppo. In seguito, il gruppo deve essere assegnato a un dispositivo di input. Infine, i widget devono essere aggiunti al gruppo.

In pratica, questo significa che quando il dispositivo di input desidera porre il focus sul widget successivo o precedente (`LV_KEY_NEXT` e `LV_KEY_PREV`), cercherà il widget nel gruppo assegnato.

Solo i widget selezionati saranno attivabili e l'ordine di attivazione sarà lo stesso in cui sono stati aggiunti.

L'oggetto con focus riceverà i tasti e altri eventi correlati. Per esempio, se una *Text Area* ha il focus e si preme una lettera sulla tastiera, il tasto verrà inviato e inserito nell'area di testo (come `LV_EVENT_KEY`). Analogamente, se uno *Slider* ha il focus e si premono le frecce sinistra o destra, il valore del cursore verrà modificato.

Utilizzo

Per creare un gruppo, si usa `lv_group_t * g = lv_group_create()`, e per aggiungere un widget al gruppo, si usa `lv_group_add_obj(g, widget)`.

Una volta aggiunto un widget a un gruppo, si può scoprire a quale gruppo appartiene usando `lv_obj_get_group(widget)`.

Per scoprire quale widget in un gruppo ha il focus, se ce n'è uno, si chiama `lv_group_get_focused(group)`. Se un widget in quel gruppo ha il focus, restituirà un puntatore ad esso; altrimenti, restituirà NULL.

Per associare un gruppo a un dispositivo di input, si usa `lv_indev_set_group(indev, g)`.

Modalità Modifica e Navigazione

Poiché un *Keypad* ha molti tasti, è facile navigare tra i widget e modificarli utilizzando tale dispositivo. Tuttavia, gli *Encoder* hanno un numero limitato di "tasti", rendendo difficile la navigazione e la regolazione dei widget.

Le modalità *Navigate* ed *Edit* vengono utilizzate per risolvere questo problema con gli encoder.

In modalità **Navigate**, il passo di un encoder viene tradotto in `LV_KEY_NEXT` o `LV_KEY_PREV`. Pertanto, l'oggetto successivo o precedente verrà selezionato ruotando l'encoder. Premendo il pulsante dell'encoder si passa alla modalità **Edit** se il widget è "editable".

In modalità **Edit**, il passaggio dell'encoder viene solitamente utilizzato per modificare il focus del widget. A seconda del tipo di widget, una pressione breve o prolungata riporta alla modalità *Navigate*. In genere, un widget che non può

essere premuto (come uno *Slider*) esce dalla modalità *Edit* con un clic breve. Ma per i widget in cui un clic breve ha un significato (ad esempio, un *Button*), è richiesta una pressione prolungata.

La modificabilità di un widget è definita nella classe del widget. Vedere, ad esempio, `.editable = LV_OBJ_CLASS_EDITABLE_TRUE` in `lv_slider.c`.

Gruppo di Default

I widget interattivi (come Pulsanti, Checkbox, Slider, ecc.) possono essere aggiunti automaticamente a un gruppo di default. Basta creare un gruppo con `lv_group_t * g = lv_group_create()` e impostare il gruppo di default con `lv_group_set_default(g)`.

Non dimenticare di assegnare uno o più dispositivi di input al gruppo di default usando `lv_indev_set_group(my_indev, g)`.

Gruppi Multipli

È anche possibile creare più gruppi. Immaginare uno strumento (ad esempio, un alimentatore) con 4 canali e pulsanti "sinistra", "destra" e "OK" per ogni canale per regolarli. Sul display, tutti e 4 i canali sono visualizzati uno sotto l'altro.

Per gestire i canali separatamente:

- Creare un gruppo per ogni canale
- Aggiungere gli elementi dell'interfaccia utente di ogni canale al suo gruppo
- Creare 4 dispositivi di input per i 4 canali
- Collegare ogni gruppo al rispettivo dispositivo di input

In questo modo, i canali possono essere gestiti separatamente in modo elegante.

7.2.7 Gestì

Panoramica

I dispositivi di input puntatore possono rilevare gesti semplici (su, giù, sinistra, destra) e multi-touch ("swipe", "pinch" [pizzico], scroll).

Per default, la maggior parte dei widget invia i gesti ai widget principali in modo che possano essere rilevati sul widget Screen sotto forma di un evento `LV_EVENT_GESTURE`. Ad esempio:

Per impedire il passaggio dell'evento "gesture" al genitore da un widget, si usa `lv_obj_remove_flag(widget, LV_OBJ_FLAG_GESTURE_BUBBLE)`.

Si noti che i gesti non vengono attivati se esegue lo scrolling di un widget.

Se si esegue un'azione su un gesto, è possibile chiamare `lv_indev_wait_release(lv_indev_active())` nel gestore eventi per impedire a LVGL di inviare ulteriori eventi relativi al dispositivo di input.

Semplici Gesture

Le gesture semplici sono sempre abilitate e molto facili da usare:

```
void my_event(lv_event_t * e)
{
    lv_obj_t * screen = lv_event_get_current_target(e);
    lv_dir_t dir = lv_indev_get_gesture_dir(lv_indev_active());
    switch(dir) {
        case LV_DIR_LEFT:
            ...
            break;
        case LV_DIR_RIGHT:
            ...
            break;
        case LV_DIR_TOP:
            ...
            break;
        case LV_DIR_BOTTOM:
            ...
            break;
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```
...
lv_obj_add_event_cb(screen1, my_event, LV_EVENT_GESTURE, NULL);
```

Per attivare un gesto, devono verificarsi due cose. Il movimento deve essere:

1. abbastanza veloce: la differenza tra il punto corrente e quello precedente deve essere maggiore di `indev->gesture_min_velocity`
2. abbastanza ampio: la differenza tra il primo punto e quello corrente deve essere maggiore di `indev->gesture_limit`

Gesti Multi-touch

LVGL è in grado di riconoscere i gesti multi-touch. Attualmente, sono supportati i seguenti gesti multi-touch:

- Pizzico con due dita (su e giù)
- Rotazione con due dita
- Scorrimento con due dita (infinito)

Per abilitare il riconoscimento dei gesti multi-touch, impostare l'opzione `LV_USE_GESTURE_RECOGNITION` nel file `lv_conf.h`.

Utilizzo

I riconoscitori possono essere aggiornati per riconoscere i gesti chiamando `lv_indev_gesture_recognizers_update(indev, touches, touch_cnt)`. Questa operazione deve essere eseguita nell'indev definito dall'utente `read_cb`. Questa operazione itererà sui riconoscitori e si fermerà una volta rilevato un gesto riconosciuto o terminato. Attualmente, è possibile riconoscere o terminare un solo gesto multi-touch alla volta.

Una volta aggiornati i riconoscitori, la chiamata a `lv_indev_gesture_recognizers_set_data(indev, data)` aggiorerà la struttura `lv_indev_data_t`. Questa operazione dovrebbe essere eseguita anche nell'indev `read_cb`.

Il modo in cui vengono raccolti i punti di contatto varia in base all'hardware e ai driver utilizzati.

Ecco un esempio generico di `read_cb`:

```
static void touch_read_callback(lv_indev_t * indev, lv_indev_data_t * data)
{
    /* Stores the collected touch events */
    lv_indev_touch_data_t touches[10];

    /* Store the current touch (finger) count */
    int32_t touch_cnt;

    /* Get the touch points */
    touch_cnt = my_read_touch_points(touches);

    lv_indev_gesture_recognizers_update(indev, touches, touch_cnt);
    lv_indev_gesture_recognizers_set_data(indev, data);

    /* Also process normal touch */
    if(touch_cnt > 0) {
        data->point.x = touches[0].point.x;
        data->point.y = touches[0].point.y;
        data->state = LV_INDEV_STATE_PRESSED;
    } else {
        data->state = LV_INDEV_STATE_RELEASED;
    }
}
```

LVGL invia eventi se i gesti si trovano in uno dei seguenti stati:

- `LV_INDEV_GESTURE_STATE_RECOGNIZED`: Il gesto è stato riconosciuto ed è ora attivo.
- `LV_INDEV_GESTURE_STATE_ENDED`: Il gesto è terminato.

Eventi

Una volta riconosciuto o terminato un gesto, viene inviato un `LV_EVENT_GESTURE`. L'utente può utilizzare queste funzioni per raccogliere maggiori informazioni sul gesto:

- `lv_event_get_gesture_type(lv_event_t * e)`: Ottiene il tipo di gesto. Usare questa funzione per verificare quale gesto multi-touch è attualmente segnalato.

- `lv_event_get_gesture_state(lv_event_t *e, lv_indev_gesture_type_t type)`: Ottiene lo stato del gesto come `lv_indev_gesture_state_t`. Può essere uno dei seguenti:

- `LV_INDEV_GESTURE_STATE_NONE`: Il gesto non è attivo.
- `LV_INDEV_GESTURE_STATE_RECOGNIZED`: Il gesto è riconosciuto e può essere utilizzato.
- `LV_INDEV_GESTURE_STATE_ENDED`: Il gesto è terminato.

L'utente può quindi richiedere i valori del gesto con le seguenti funzioni:

- `lv_event_get_pinch_scale(lv_event_t * e)`: Ottiene la scala del "pinch". Rilevante solo per i gesti di "pinch" [pizzico].
- `lv_event_get_rotation(lv_event_t * e)`: Ottiene la rotazione in radianti. Rilevante solo per i gesti di rotazione.
- `lv_event_get_two_fingersSwipeDistance(lv_event_t * e)`: Ottiene la distanza in pixel dal centro di partenza del gesto. Rilevante solo per i gesti di "swipe" [scorrimento] a due dita.
- `lv_event_get_two_fingersSwipeDir(lv_event_t * e)`: Ottiene la direzione dal centro di partenza. Rilevante solo per i gesti di "swipe" [scorrimento] a due dita.

Ciò consente all'utente di reagire ai gesti e di utilizzarne i valori. Un esempio di tale applicazione è disponibile nell'albero sorgente: `examples/others/gestures/lv_example_gestures.c`.

Soglie

I riconoscitori dei gesti possono essere configurati per modificare le soglie dei gesti:

- `lv_indev_set_pinch_up_threshold(lv_indev_t *indev, float threshold)`: Imposta la soglia di pinch-up (zoom in) in pixel.
- `lv_indev_set_pinch_down_threshold(lv_indev_t *indev, float threshold)`: Imposta la soglia di pinch-down (zoom out) in pixel.
- `lv_indev_set_rotation_rad_threshold(lv_indev_t *indev, float threshold)`: Imposta la soglia dell'angolo di rotazione in radianti.

Esempio di libinput

Nel caso di libinput, gli eventi touch vengono ricevuti in modo asincrono. Per gestirli, l'array touch e il conteggio touch devono essere variabili globali:

```
/* Stores the collected touch events */
static lv_indev_touch_data_t touches[10];

/* Store the current touch (finger) count SINCE THE LAST READ */
static int32_t touch_cnt;
```

quindi il flusso è:

1. Salva gli eventi touch in modo asincrono nell'array touches (un evento per dito)
2. Elaborare i dati touch in `read_cb`
3. Ignorare i touch e attendere nuovi eventi touch impostando `touch_cnt = 0` alla fine di `read_cb`

```
/**
 * @brief Convert the libinput event to LVGL's representation of a touch event
 * @param ev a pointer to the libinput event
 */
static void touch_process(struct libinput_event *ev)
{
    int type = libinput_event_get_type(ev);
    struct libinput_event_touch *touch_ev = libinput_event_get_touch_event(ev);
    int id = libinput_event_touch_get_slot(touch_ev);
    uint32_t time = libinput_event_touch_get_time(touch_ev);

    /* Get the last event for the contact point */
    lv_indev_touch_data_t *cur = NULL;

    /* Find if the touch is already stored in the array by its ID.
     * If there are 2 active touches it's enough to check those. */
    for (int i = 0; i < touch_cnt; i++) {
        if (touches[i].id == id) {
            cur = &touches[i];
        }
    }

    /* Add new touch to the array */
    if (cur == NULL) {
        cur = &touches[touch_cnt];
        touch_cnt++;
    }

    /* Set the new touch information */
    cur->id = id;
    cur->time = time;
    cur->type = type;
}
```

(continues on next page)

(continua dalla pagina precedente)

```

        break;
    }

    if (cur && cur->timestamp == time) {
        /* Previous event has the same timestamp - ignore duplicate event */
        return;
    }

    /* Save the new touch */
    if (cur == NULL || 
        type == LIBINPUT_EVENT_TOUCH_UP || 
        type == LIBINPUT_EVENT_TOUCH_DOWN) {

        cur = &touches[touch_cnt];
        cur->id = id; /* It will be needed to identify the touches in the next events */
        touch_cnt++;
    }

    /* Process the event */
    switch (type) {
        case LIBINPUT_EVENT_TOUCH_DOWN:
        case LIBINPUT_EVENT_TOUCH_MOTION:
            cur->point.x = (int) libinput_event_touch_get_x_transformed(touch_ev, SCREEN_WIDTH);
            cur->point.y = (int) libinput_event_touch_get_y_transformed(touch_ev, SCREEN_HEIGHT);
            cur->state = LV_INDEV_STATE_PRESSED;
            break;

        case LIBINPUT_EVENT_TOUCH_UP:
            cur->state = LV_INDEV_STATE_RELEASED;
            cur->point.x = 0;
            cur->point.y = 0;
            break;
    }

    cur->timestamp = time;
}

/**
 * @brief Filter out libinput events that are not related to touches
 * @param ev a pointer to the libinput event
 */
static void process_libinput_event(struct libinput_event *ev)
{
    int type = libinput_event_get_type(ev);

    switch (type) {
        case LIBINPUT_EVENT_TOUCH_MOTION:
        case LIBINPUT_EVENT_TOUCH_DOWN:
        case LIBINPUT_EVENT_TOUCH_UP:
            /* Filter only touch events */
            touch_process(ev);
            break;
        default:
            /* Skip unrelated libinput events */
            return;
    }
}

```

7.2.8 Navigazione a Griglia

Panoramica

La "Grid navigation" [navigazione a griglia] (gridnav in breve) è una funzionalità che sposta il focus tra un insieme di Widget figli tramite la pressione dei tasti freccia.

Se i Widget figli sono in una disposizione a griglia, le frecce su, giù, sinistra e destra spostano il focus sul fratello più vicino nella rispettiva direzione.

Non importa come siano posizionati i figli, poiché vengono considerate solo le coordinate x e y correnti. Ciò significa che gridnav funziona con i figli posizionati manualmente, così come con i layout *Flex* e *Grid*.

Gridnav funziona anche se i figli sono disposti in una singola riga o colonna. Questo è utile, ad esempio, per semplificare la navigazione tra gli elementi in una *Lista di Widget*.

Gridnav presuppone che il Widget a cui viene aggiunto faccia parte di un *gruppo*. In questo modo, se il Widget con gridnav ha il focus, le pressioni dei tasti freccia vengono automaticamente inoltrate al Widget in modo che gridnav possa elaborarle.

Per spostare il focus sul widget successivo del gruppo, si usa *LV_KEY_NEXT* o *LV_KEY_PREV*. Facoltativamente, è possibile utilizzare anche *lv_group_focus_next()* o *lv_group_focus_prev()* o il tasto TAB sulla tastiera, come di consueto.

Se il contenitore è scorrevole e il figlio con focus è fuori dalla vista, gridnav scorrerà automaticamente il figlio nella vista.

Utilizzo

Per aggiungere il comportamento di GridNav a qualsiasi Widget (ad esempio, uno che funge da contenitore per i Widget figli tra cui l'utente finale navigherà utilizzando i tasti freccia), si usa `lv_gridnav_add(container, flags)`.

L'argomento `flags` controlla il comportamento della navigazione:

- `LV_GRIDNAV_CTRL_NONE`: Impostazioni di default
- `LV_GRIDNAV_CTRL_ROLLOVER`: Se non è presente alcun Widget successivo/precedente in una direzione, il focus passa al Widget nella riga successiva/precedente (con i tasti sinistra/destra) o nella prima/ultima riga (con i tasti su/giù)
- `LV_GRIDNAV_CTRL_SCROLL_FIRST`: Se si preme una freccia e il Widget col focus può essere fatto scorrere in quella direzione, verrà fatto scorrere invece di passare al Widget successivo/precedente. Se non c'è più spazio per lo scorrimento, il Widget successivo/precedente riceverà il focus normalmente.
- `LV_GRIDNAV_CTRL_HORIZONTAL_MOVE_ONLY`: Utilizza solo i tasti sinistro/destro per la navigazione nella griglia. Gli eventi dei tasti su/giù verranno inviati al Widget attivo.
- `LV_GRIDNAV_CTRL_VERTICAL_MOVE_ONLY`: Utilizza solo i tasti su/giù per la "grid navigation". Gli eventi dei tasti sinistro/destro verranno inviati al Widget che ha il focus.

Sebbene i comportamenti sopra descritti possano essere combinati tramite l'operazione OR bit per bit dei valori sopra indicati, `LV_GRIDNAV_CTRL_HORIZONTAL_MOVE_ONLY` e `LV_GRIDNAV_CTRL_VERTICAL_MOVE_ONLY` non devono essere utilizzati insieme.

`lv_gridnav_remove(container)` Rimuove il comportamento di Gridnav da un Widget.

Widget con Focus

Un widget deve essere cliccabile o con focus cliccabile (`LV_OBJ_FLAG_CLICKABLE` o `LV_OBJ_FLAG_CLICK_FOCUSABLE`) e non nascosto (`LV_OBJ_FLAG_HIDDEN`) per ricevere il focus tramite gridnav.

Esempi

Grid navigation di base

```
#include "../../lv_examples.h"
#if LV_USE_GRIDNAV && LV_USE_FLEX && LV_BUILD_EXAMPLES

/***
 * Demonstrate a basic grid navigation
 */
void lv_example_gridnav_1(void)
{
    /*It's assumed that the default group is set and
     *there is a keyboard indev*/
    lv_obj_t * cont1 = lv_obj_create(lv_screen_active());
    lv_gridnav_add(cont1, LV_GRIDNAV_CTRL_NONE);

    /*Use flex here, but works with grid or manually placed objects as well*/
    lv_obj_set.flex_flow(cont1, LV_FLEX_FLOW_WRAP);
    lv_obj.set_style.bg_color(cont1, lv.palette_lighten(LV_PALETTE_BLUE, 5), LV_STATE_FOCUSED);
    lv_obj.set_size(cont1, lv_pct(50), lv_pct(100));

    /*Only the container needs to be in a group*/
    lv_group_add_obj(lv_group_get_default(), cont1);

    lv_obj_t * label = lv_label_create(cont1);
    lv_label.set_text_fmt(label, "No rollover");

    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_obj_t * obj = lv_button_create(cont1);
        lv_obj.set_size(obj, 70, LV_SIZE_CONTENT);
        lv_obj.add_flag(obj, LV_OBJ_FLAG_CHECKABLE);
        lv_group.remove_obj(obj);   /*Not needed, we use the gridnav instead*/

        label = lv_label_create(obj);
        lv_label.set_text_fmt(label, "%" LV_PRIu32, i);
        lv_obj_center(label);
    }

    /* Create a second container with rollover grid nav mode.*/
    lv_obj_t * cont2 = lv_obj_create(lv_screen_active());
    lv_gridnav_add(cont2, LV_GRIDNAV_CTRL_ROLLOVER);
    lv_obj.set.style.bg_color(cont2, lv.palette_lighten(LV_PALETTE_BLUE, 5), LV_STATE_FOCUSED);
    lv_obj.set.size(cont2, lv_pct(50), lv_pct(100));
    lv_obj.align(cont2, LV_ALIGN_RIGHT_MID, 0, 0);

    label = lv_label_create(cont2);
    lv_obj.set_width(label, lv_pct(100));
    lv_label.set_text_fmt(label, "Rollover\nUse tab to focus the other container");
}
```

(continues on next page)

(continua dalla pagina precedente)

```

/*Only the container needs to be in a group*/
lv_group_add_obj(lv_group_get_default(), cont2);

/*Add and place some children manually*/
lv_obj_t * ta = lv_textarea_create(cont2);
lv_obj_set_size(ta, lv_pct(100), 80);
lv_obj_set_pos(ta, 0, 80);
lv_group_remove_obj(ta); /*Not needed, we use the gridnav instead*/

lv_obj_t * cb = lv_checkbox_create(cont2);
lv_obj_set_pos(cb, 0, 170);
lv_group_remove_obj(cb); /*Not needed, we use the gridnav instead*/

lv_obj_t * sw1 = lv_switch_create(cont2);
lv_obj_set_pos(sw1, 0, 200);
lv_group_remove_obj(sw1); /*Not needed, we use the gridnav instead/

lv_obj_t * sw2 = lv_switch_create(cont2);
lv_obj_set_pos(sw2, lv_pct(50), 200);
lv_group_remove_obj(sw2); /*Not needed, we use the gridnav instead/
}

#endif

```

Grid navigation in una lista

```

#include "../../lv_examples.h"
#if LV_USE_GRIDNAV && LV_USE_LIST && LV_BUILD_EXAMPLES

/**
 * Grid navigation on a list
 */
void lv_example_gridnav_2(void)
{
    /*It's assumed that the default group is set and
     *there is a keyboard indev*/

    lv_obj_t * list1 = lv_list_create(lv_screen_active());
    lv_gridnav_add(list1, LV_GRIDNAV_CTRL_NONE);
    lv_obj_set_size(list1, lv_pct(45), lv_pct(80));
    lv_obj_align(list1, LV_ALIGN_LEFT_MID, 5, 0);
    lv_obj_set_style_bg_color(list1, lv_palette_lighten(LV_PALETTE_BLUE, 5), LV_STATE_FOCUSED);
    lv_group_add_obj(lv_group_get_default(), list1);

    char buf[32];
    uint32_t i;
    for(i = 0; i < 15; i++) {
        lv_snprintf(buf, sizeof(buf), "File %d", i + 1);
        lv_obj_t * item = lv_list_add_button(list1, LV_SYMBOL_FILE, buf);
        lv_obj_set_style_bg_opa(item, 0, 0);
        lv_group_remove_obj(item); /*Not needed, we use the gridnav instead*/
    }

    lv_obj_t * list2 = lv_list_create(lv_screen_active());
    lv_gridnav_add(list2, LV_GRIDNAV_CTRL_ROLLOVER);
    lv_obj_set_size(list2, lv_pct(45), lv_pct(80));
    lv_obj_align(list2, LV_ALIGN_RIGHT_MID, -5, 0);
    lv_obj_set_style_bg_color(list2, lv_palette_lighten(LV_PALETTE_BLUE, 5), LV_STATE_FOCUSED);
    lv_group_add_obj(lv_group_get_default(), list2);

    for(i = 0; i < 15; i++) {
        lv_snprintf(buf, sizeof(buf), "Folder %d", i + 1);
        lv_obj_t * item = lv_list_add_button(list2, LV_SYMBOL_DIRECTORY, buf);
        lv_obj_set_style_bg_opa(item, 0, 0);
        lv_group_remove_obj(item);
    }
}
#endif

```

Grid navigation nidificate

```

#include "../../lv_examples.h"
#if LV_USE_GRIDNAV && LV_USE_FLEX && LV_BUILD_EXAMPLES

static void cont_sub_event_cb(lv_event_t * e)
{
    uint32_t k = lv_event_get_key(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(k == LV_KEY_ENTER) {
        lv_group_focus_obj(obj);
    } else if(k == LV_KEY_ESC) {
        lv_group_focus_next(lv_obj_get_group(obj));
    }
}

/**
 * Nested grid navigations
 */
void lv_example_gridnav_3(void)
{
    /*It's assumed that the default group is set and
     *there is a keyboard indev*/

    lv_obj_t * cont_main = lv_obj_create(lv_screen_active());
    lv_gridnav_add(cont_main, (lv_gridnav_ctrl_t)(LV_GRIDNAV_CTRL_ROLLOVER | LV_GRIDNAV_CTRL_SCROLL_FIRST));

    /*Only the container needs to be in a group*/
    lv_group_add_obj(lv_group_get_default(), cont_main);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

/*Use flex here, but works with grid or manually placed objects as well*/
lv_obj_set_flex_flow(cont_main, LV_FLEX_FLOW_ROW_WRAP);
lv_obj_set_style_bg_color(cont_main, lv_palette_lighten(LV_PALETTE_BLUE, 5), LV_STATE_FOCUSED);
lv_obj_set_size(cont_main, lv_pct(80), LV_SIZE_CONTENT);

lv_obj_t * btn;
lv_obj_t * label;

btn = lv_button_create(cont_main);
lv_group_remove_obj(btn);
label = lv_label_create(btn);
lv_label_set_text(label, "Button 1");

btn = lv_button_create(cont_main);
lv_group_remove_obj(btn);
label = lv_label_create(btn);
lv_label_set_text(label, "Button 2");

/*Create another container with long text to show how LV_GRIDNAV_CTRL_SCROLL_FIRST works*/
lv_obj_t * cont_sub1 = lv_obj_create(cont_main);
lv_obj_set_size(cont_sub1, lv_pct(100), 100);

label = lv_label_create(cont_sub1);
lv_obj_set_style_bg_color(cont_sub1, lv_palette_lighten(LV_PALETTE_RED, 5), LV_STATE_FOCUSED);
lv_obj_set_width(label, lv_pct(100));
lv_label_set_text(label, "Hello world!\n"
"Hello world!");

/*Create a third container that can be focused with ENTER and contains another grid nav*/
lv_obj_t * cont_sub2 = lv_obj_create(cont_main);
lv_gridnav_add(cont_sub2, LV_GRIDNAV_CTRL_ROLLOVER);
/*Only the container needs to be in a group*/
lv_group_add_obj(lv_group_get_default(), cont_sub2);

lv_obj_add_event_cb(cont_sub2, cont_sub_event_cb, LV_EVENT_KEY, NULL);

/*Use flex here, but works with grid or manually placed objects as well*/
lv_obj_set_flex_flow(cont_sub2, LV_FLEX_FLOW_ROW_WRAP);
lv_obj_set_style_bg_color(cont_sub2, lv_palette_lighten(LV_PALETTE_RED, 5), LV_STATE_FOCUSED);
lv_obj_set_size(cont_sub2, lv_pct(100), LV_SIZE_CONTENT);

label = lv_label_create(cont_sub2);
lv_label_set_text(label, "Use ENTER/ESC to focus/defocus this container");
lv_obj_set_width(label, lv_pct(100));

btn = lv_button_create(cont_sub2);
lv_group_remove_obj(btn);
label = lv_label_create(btn);
lv_label_set_text(label, "Button 3");

btn = lv_button_create(cont_sub2);
lv_group_remove_obj(btn);
label = lv_label_create(btn);
lv_label_set_text(label, "Button 4");

}

#endif

```

Grid navigation semplice su una lista di widget

```

#include "../../lv_examples.h"
#if LV_USE_GRIDNAV && LV_USE_FLEX && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_obj_t * list = lv_obj_get_parent(obj);
    LV_UNUSED(list); /*If logging is disabled*/
    LV_LOG_USER("Clicked: %s", lv_list_get_button_text(list, obj));
}

/**
 * Simple navigation on a list widget
 */
void lv_example_gridnav_4(void)
{
    /*It's assumed that the default group is set
     *there is a keyboard indev/

    lv_obj_t * list = lv_list_create(lv_screen_active());
    lv_gridnav_add(list, LV_GRIDNAV_CTRL_ROLLOVER);
    lv_obj_align(list, LV_ALIGN_LEFT_MID, 10, 0);
    lv_group_add_obj(lv_group_get_default(), list);

    uint32_t i;
    for(i = 0; i < 20; i++) {
        char buf[32];

        /*Add some separators too, they are not focusable by gridnav*/
        if((i % 5) == 0) {
            lv_snprintf(buf, sizeof(buf), "Section %d", i / 5 + 1);
            lv_list_add_text(list, buf);
        }

        lv_snprintf(buf, sizeof(buf), "File %d", i + 1);
        lv_obj_t * item = lv_list_add_button(list, LV_SYMBOL_FILE, buf);
        lv_obj_add_event_cb(item, event_handler, LV_EVENT_CLICKED, NULL);
        lv_group_remove_obj(item); /*The default group adds it automatically*/
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

}
lv_obj_t * btn = lv_button_create(lv_screen_active());
lv_obj_align(btn, LV_ALIGN_RIGHT_MID, -10, 0);
lv_obj_t * label = lv_label_create(btn);
lv_label_set_text(label, "Button");
}

#endif

```

Grid navigation per un solo asse

```

#include "../../lv_examples.h"
#if LV_USE_GRIDNAV && LV_USE_FLEX && LV_BUILD_EXAMPLES

static const char * opts[] = {"0\n1\n2\n3\n4\n5", "0\n1\n2\n3\n4\n5\n6\n7\n8\n9", "s\nm\nh"};
static const int32_t opts_counts[] = {6, 10, 3};

static lv_obj_t * sliders[3];
static lv_obj_t * rollers[3];

static void slider_key_cb(lv_event_t * e)
{
    uint8_t i = (uint32_t)(uintptr_t)lv_event_get_user_data(e);
    lv_roller_set_selected(rollers[i], lv_slider_get_value(sliders[i]), LV_ANIM_ON);
}

static void roller_key_cb(lv_event_t * e)
{
    uint8_t i = (uint32_t)(uintptr_t)lv_event_get_user_data(e);
    lv_slider_set_value(sliders[i], lv_roller_get_selected(rollers[i]), LV_ANIM_ON);
}

/**
 * Grid navigation for only one axis
 */
void lv_example_gridnav_5(void)
{
    /*It's assumed that the default group is set and
     *there is a keyboard indev*/

    lv_group_t * group = lv_group_get_default();
    lv_obj_t * cont;

    cont = lv_obj_create(lv_screen_active());
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(cont, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);
    lv_obj_set_size(cont, lv_pct(100), lv_pct(50));
    lv_obj_align(cont, LV_ALIGN_TOP_MID, 0, 0);
    /* only up/down keys will be used for grid navigation in this container. */
    /* left/right will be sent to the sliders */
    lv_gridnav_add(cont, LV_GRIDNAV_CTRL_VERTICAL_MOVE_ONLY);
    lv_group_add_obj(group, cont);
    for(uint32_t i = 0; i < 3; i++) {
        lv_obj_t * slider = lv_slider_create(cont);
        lv_slider_set_range(slider, 0, opts_counts[i] - 1);
        lv_group_remove_obj(slider);
        lv_obj_set_width(slider, lv_pct(85));
        sliders[i] = slider;
        lv_obj_add_event_cb(slider, slider_key_cb, LV_EVENT_KEY, (void*)(uintptr_t)i);
    }

    cont = lv_obj_create(lv_screen_active());
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW);
    lv_obj_set_flex_align(cont, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);
    lv_obj_set_size(cont, lv_pct(100), lv_pct(50));
    lv_obj_align(cont, LV_ALIGN_BOTTOM_MID, 0, 0);
    /* only left/right keys will be used for grid navigation in this container. */
    /* up/down will be sent to the rollers */
    lv_gridnav_add(cont, LV_GRIDNAV_CTRL_HORIZONTAL_MOVE_ONLY);
    lv_group_add_obj(group, cont);
    for(uint32_t i = 0; i < 3; i++) {
        lv_obj_t * roller = lv_roller_create(cont);
        lv_roller_set_options(roller, opts[i], LV_ROLLER_MODE_INFINITE);
        lv_obj_set_size(roller, lv_pct(30), lv_pct(100));
        lv_group_remove_obj(roller);
        rollers[i] = roller;
        lv_obj_add_event_cb(roller, roller_key_cb, LV_EVENT_KEY, (void*)(uintptr_t)i);
    }
}

#endif

```

API

[lv_gridnav.h](#)

7.3 Font (lv_font)

7.3.1 Panoramica

Cos'è un Font?

In LVGL, i font sono raccolte di bitmap e altre informazioni necessarie per riprodurre le immagini delle singole lettere (glifi). Un font è memorizzato in una variabile `lv_font_t` e può essere impostato nel campo `text_font` di uno stile. Ad esempio:

```
lv_style_set_text_font(&my_style, &lv_font_montserrat_28); /* Set a larger font */
```

Motori dei Font

Un "font engine" è un codice C che consente a LVGL di estrarre varie informazioni dai font, come informazioni sui caratteri (glifi) o bitmap.

Il "font engine" integrato in LVGL è adatto alla maggior parte dei casi tipici. Può gestire diverse impostazioni di bit per pixel (1, 2, 3, 4, 8) nelle bitmap, la crenatura, la selezione di intervalli di caratteri da più font, la compressione delle bitmap e molte altre.

Il "font engine" integrato è anche il più semplice da usare:

1. Andare su <https://lvgl.io/tools/fontconverter>
2. Caricare i font e impostare gli intervalli e/o specificare un elenco di caratteri da includere e altri parametri
3. Cliccare sul pulsante "Submit" e copiare il file generato nel progetto
4. In un file C si aggiunge `LV_FONT_DECLARE(font_name)` per dichiarare il font
5. Usare il font come `lv_style_set_text_font(&my_style, &font_name);` o `lv_obj_set_style_text_font(label1, &font_name, 0);`

LVGL supporta anche diversi altri "font engine":

- `fmt_txt`: Questo è il "font engine" nativo che memorizza i font come array C
- `binfont`: Simile al formato nativo, ma il font viene memorizzato come file, quindi può essere caricato anche in fase di esecuzione
- `tiny_ttf`: Piccolo motore di grafica vettoriale per caricare file TTF in fase di esecuzione di qualsiasi dimensione
- `freetype`: Nota libreria di rendering di font che carica e renderizza font TTF in fase di esecuzione. Supporta anche gli "strok" [tratti] delle lettere.

Supporto Unicode

LVGL supporta i caratteri Unicode codificati **UTF-8**. L'editor deve essere configurato per salvare il codice/testo come UTF-8 (solitamente è il default) e assicurarsi che `LV_TXT_ENC` sia impostato su `LV_TXT_ENC_UTF8` in `lv_conf.h`. (Questo è il valore di default).

Per testarlo, provare

```
lv_obj_t * label1 = lv_label_create(lv_screen_active(), NULL);
lv_label_set_text(label1, LV_SYMBOL_OK);
```

Se tutto funzionasse correttamente, dovrebbe essere visualizzato un carattere '✓'.

Composizione

Sebbene LVGL possa decodificare e visualizzare qualsiasi carattere Unicode (a condizione che il font li supporti), non è in grado di visualizzare correttamente tutte le lingue complesse.

Le lingue standard basate sul latino (ad esempio, inglese, spagnolo, tedesco) e le lingue dell'Asia orientale come cinese, giapponese e coreano (CJK) sono relativamente semplici, poiché i loro caratteri vengono scritti semplicemente da sinistra a destra.

Anche lingue come l'arabo, il persiano e l'ebraico, che utilizzano direzioni di scrittura da destra a sinistra (RTL) o miste, sono supportate in LVGL. Per saperne di più [qui](#).

Per caratteri come 'é', 'ü', 'ö', 'Á' e 'Ñ', si consiglia di utilizzare il formato Unicode singolo (NFC) anziché scomporli in una lettera base e segni diacritici (ad esempio u + 'ñ').

Le lingue complesse in cui i caratteri successivi si combinano in un unico glifo e in cui il glifo risultante non ha una rappresentazione Unicode individuale (ad esempio, il devanagari), hanno un supporto limitato in LVGL.

Crenatura

I font solitamente forniscono informazioni sul kerning per regolare la spaziatura tra caratteri specifici.

- Il [Convertitore online](#) genera tabelle di kerning.
- Il [Convertitore offline](#) genera tabelle di kerning a meno che non venga specificato `--no-kerning`.
- L'integrazione con FreeType attualmente non supporta il "kerning" [crenatura].
- Il motore di font Tiny TTF supporta le tabelle GPOS (Glyph Positioning) e Kern.

Per configurare il kerning in fase di esecuzione, si usa `lv_font_set_kerning()`.

Utilizzo del Font Fallback

Se il font in uso non ha un glifo necessario per un'attività di rendering del testo, è possibile specificare un font fallback da utilizzare in `lv_font_t`.

`fallback` può essere concatenato, quindi tenterà di risolvere il problema finché non verrà più impostato alcun `fallback`.

```
/* Roboto font doesn't have support for CJK glyphs */
lv_font_t *robot = my_font_load_function();
/* Droid Sans Fallback has more glyphs but its typeface doesn't look good as Roboto */
lv_font_t *droid_sans_fallback = my_font_load_function();
/* So now we can display Roboto for supported characters while having wider characters set support */
robot->fallback = droid_sans_fallback;
```

Simboli

LVGL supporta alcuni "simboli" predefiniti. Un simbolo è un carattere Unicode specifico in un font con un'immagine simile a un'icona. I simboli hanno nomi come `LV_SYMBOL_OK`, `LV_SYMBOL_HOME`, ecc. Consultare l'elenco completo dei simboli predefiniti qui sotto:

♫ LV_SYMBOL_AUDIO	⚠ LV_SYMBOL_WARNING
🎥 LV_SYMBOL_VIDEO	🔀 LV_SYMBOL_SHUFFLE
☰ LV_SYMBOL_LIST	^K LV_SYMBOL_UP
✓ LV_SYMBOL_OK	⌄ LV_SYMBOL_DOWN
✗ LV_SYMBOL_CLOSE	🔁 LV_SYMBOL_LOOP
⚡ LV_SYMBOL_POWER	📁 LV_SYMBOL_DIRECTORY
⚙️ LV_SYMBOL_SETTINGS	📤 LV_SYMBOL_UPLOAD
🗑 LV_SYMBOL_TRASH	📞 LV_SYMBOL_CALL
🏡 LV_SYMBOL_HOME	✖️ LV_SYMBOL_CUT
⬇️ LV_SYMBOL_DOWNLOAD	📄 LV_SYMBOL_COPY
📀 LV_SYMBOL_DRIVE	💾 LV_SYMBOL_SAVE
⟳ LV_SYMBOL_REFRESH	⚡ LV_SYMBOL_CHARGE
🔇 LV_SYMBOL_MUTE	📄 LV_SYMBOL_PASTE
🔉 LV_SYMBOL_VOLUME_MID	🔔 LV_SYMBOL_BELL
🔊 LV_SYMBOL_VOLUME_MAX	⌨️ LV_SYMBOL_KEYBOARD
🖼️ LV_SYMBOL_IMAGE	📍 LV_SYMBOL_GPS
📝 LV_SYMBOL_EDIT	📄 LV_SYMBOL_FILE
⏪ LV_SYMBOL_PREV	📶 LV_SYMBOL_WIFI
⏩ LV_SYMBOL_PLAY	🔋 LV_SYMBOL_BATTERY_FULL
⏸ LV_SYMBOL_PAUSE	🔋 LV_SYMBOL_BATTERY_3
⏹ LV_SYMBOL_STOP	🔋 LV_SYMBOL_BATTERY_2
⏭️ LV_SYMBOL_NEXT	🔋 LV_SYMBOL_BATTERY_1
⏏️ LV_SYMBOL_EJECT	🔋 LV_SYMBOL_EMPTY
◀️ LV_SYMBOL_LEFT	🔌 LV_SYMBOL_USB
▶️ LV_SYMBOL_RIGHT	Bluetooth LV_SYMBOL_BLUETOOTH
➕ LV_SYMBOL_PLUS	✖️ LV_SYMBOL_BACKSPACE
➖ LV_SYMBOL_MINUS	_SD CARD LV_SYMBOL_SD_CARD
👁️ LV_SYMBOL_EYE_OPEN	➡️ LV_SYMBOL_NEW_LINE
👁️ LV_SYMBOL_EYE_CLOSE	

I simboli nei *font built-in* sono creati a partire dal font FontAwesome.

Utilizzare questi simboli è molto semplice:

```
lv_label_set_text(label, LV_SYMBOL_OK); /*Just a symbol*/
lv_label_set_text(label, LV_SYMBOL_OK "Apply"); /*Concatenate with a string*/
```

Per aggiungere un nuovo simbolo in un font personalizzato:

1. Cercare un simbolo su <https://fontawesome.com>. Ad esempio, il simbolo USB. Copiare il suo ID Unicode, che è `0xf287`.
2. Aprire il [Convertitore di font online](#). Aggiungere `FontAwesome.woff`.
3. Impostare i parametri come Nome, Dimensione, BPP. Si userà questo nome per dichiarare e utilizzare il font nel codice.
4. Aggiungere l'ID Unicode del simbolo al campo "range". Ad esempio, `0xf287` per il simbolo USB. È possibile enumerare altri simboli con , [virgola].
5. Convertire il font e copiare il codice sorgente generato nel progetto. Assicurarsi di compilare il file .c del proprio font.

- Dichiarare il font utilizzando `LV_FONT_DECLARE(my_font_name)`.

Uso del simbolo

- Convertire il valore Unicode in UTF8, ad esempio su [questo sito](#). Per `0xf287` i byte *Hex UTF-8* sono `EF 8A 87`.
- Creare una stringa `#define` dai valori UTF8: `#define MY_USB_SYMBOL "\xEF\x8A\x87"`
- Creare un'etichetta e impostare il testo. Es. `lv_label_set_text(label, MY_USB_SYMBOL)`

nota

`lv_label_set_text(label, MY_USB_SYMBOL)` cerca questo simbolo nel font definito nella proprietà `text.font` dello stile. Per utilizzare il simbolo, è necessario impostare il font del testo dello stile in modo che utilizzi il font generato, ad esempio `lv_style_set_text_font(&my_style, &my_font_name)` o `lv_obj_set_style_text_font(label, &my_font_name, 0)`.

Naturalmente è possibile utilizzare qualsiasi altro font, assicurandosi solo che definisca i simboli necessari.

API

7.3.2 Font Integrati

Panoramica

Sono disponibili diversi font integrati in diverse dimensioni, che possono essere abilitati in `lv_conf.h` con le definizioni `LV_FONT_....`

I font integrati sono **variabili globali** con nomi come `lv_font_montserrat_16` per un font di 16 px di altezza. Per utilizzarli in uno stile, basta aggiungere un puntatore a una variabile font in questo modo:

```
lv_style_set_text_font(&my_style, &lv_font_montserrat_28);
```

I font predefiniti con `bpp = 4` contengono i caratteri ASCII e utilizzano il font Montserrat.

Oltre al range ASCII, tutti i *simboli predefiniti* vengono aggiunti ai font integrati dal font FontAwesome.

Font Normali

I seguenti font contengono tutti i caratteri ASCII, il simbolo del grado (`U+00B0`), il simbolo del punto elenco (`U+2022`) e i simboli integrati (vedere di seguito).

- `LV_FONT_MONTSERRAT_12`: 12 px font
- `LV_FONT_MONTSERRAT_14`: 14 px font
- `LV_FONT_MONTSERRAT_16`: 16 px font
- `LV_FONT_MONTSERRAT_18`: 18 px font
- `LV_FONT_MONTSERRAT_20`: 20 px font
- `LV_FONT_MONTSERRAT_22`: 22 px font
- `LV_FONT_MONTSERRAT_24`: 24 px font
- `LV_FONT_MONTSERRAT_26`: 26 px font
- `LV_FONT_MONTSERRAT_28`: 28 px font
- `LV_FONT_MONTSERRAT_30`: 30 px font
- `LV_FONT_MONTSERRAT_32`: 32 px font
- `LV_FONT_MONTSERRAT_34`: 34 px font
- `LV_FONT_MONTSERRAT_36`: 36 px font
- `LV_FONT_MONTSERRAT_38`: 38 px font
- `LV_FONT_MONTSERRAT_40`: 40 px font

- `LV_FONT_MONTserrat_42`: 42 px font
- `LV_FONT_MONTserrat_44`: 44 px font
- `LV_FONT_MONTserrat_46`: 46 px font
- `LV_FONT_MONTserrat_48`: 48 px font

Font Speciali

- `LV_FONT_MONTserrat_28_COMPRESSED`: Uguale al font normale da 28 px, ma memorizzato come *Font Compressi* con 3 bpp
- `LV_FONT_DEJAVU_16_PERSIAN_HEBREW`: Font da 16 px con intervallo normale + lettere ebraiche, arabe, persiane e tutte le loro forme
- `LV_FONT_SOURCE_HAN_SANS_SC_16_CJK`: Font da 16 px con intervallo normale più 1000 dei radicali CJK più comuni
- `LV_FONT_UNSCII_8`: Font pixel perfect da 8 px con solo caratteri ASCII
- `LV_FONT_UNSCII_16`: Font pixel perfect da 16 px con solo caratteri ASCII

Aggiungere un Nuovo Font

Esistono diversi modi per aggiungere un nuovo font al progetto:

1. Il metodo più semplice è utilizzare il [Convertitore di font online](#). Basta impostare i parametri, cliccare sul pulsante *Convert*, copiare il font nel progetto e utilizzarlo. **Assicurarsi di leggere attentamente i passaggi forniti su quel sito o si riceverà un errore durante la conversione.**
2. Utilizzo del [Convertitore di font offline](#). (Richiede l'installazione di Node.js)
3. Per creare qualcosa di simile ai font nativi (font e simboli Montserrat) ma con dimensioni e/o intervalli diversi, si può utilizzare lo script `built_in_font_gen.py` nella cartella `lvgl/scripts/built_in_font`. (Questo richiede l'installazione di Python e di https://github.com/lvgl/lv_font_conv/.)

Per dichiarare un font in un file, si usa `LV_FONT_DECLARE(my_font_name)`.

Per rendere i font disponibili globalmente (come quelli nativi), si aggiungono a `LV_FONT_CUSTOM_DECLARE` in `lv_conf.h`.

Font Compressi

Il motore integrato dei font supporta bitmap compresse. I font compressi possono essere generati

- selezionando la check box `Compressed` nel convertitore online
- non passando il flag `--no-compress` al convertitore offline (la compressione viene applicata per default)

La compressione è più efficace con font più grandi e un bpp più elevato. Tuttavia, il rendering dei font compressi è circa il 30% più lento. Pertanto, si consiglia di comprimere solo i font più grandi di un'interfaccia utente, perché

- richiedono più memoria
- possono essere compressi meglio
- e, dato che probabilmente vengono utilizzati meno frequentemente rispetto ai font di medie dimensioni, il costo in termini di prestazioni sarà inferiore.

I font compressi supportano anche `bpp=3`.

7.3.3 Loader BinFont

Panoramica

`lv_binfont_create()` può essere utilizzato per caricare un font da un file. Il font deve avere un formato binario speciale. (Non TTF o WOFF). Si usa `lv_font_conv` con l'opzione `--format bin` per generare un file font compatibile con LVGL.

Caricamento da File

nota

Per caricare un font, *Il file system di LVGL* deve essere abilitato e deve essere aggiunto un driver.

Esempio

```
lv_font_t *my_font = lv_binfont_create("X:/path/to/my_font.bin");
if(my_font == NULL) return;

/* Use the font */

/* Free the font if not required anymore */
lv_binfont_destroy(my_font);
```

Caricamento dalla Memoria

`lv_binfont_create_from_buffer()` può essere utilizzato per caricare un font da un buffer di memoria. Questa funzione può essere utile per caricare un font da un file system esterno, non supportato da LVGL. Il font deve essere nello stesso formato in cui si troverebbe se fosse caricato da un file.

nota

Per caricare un font da un buffer, *Il file system di LVGL* deve essere abilitato e deve essere aggiunto il driver MEMFS.

Esempio

```
lv_font_t *my_font;
uint8_t *buf;
uint32_t bufsize;

/* Read font file into the buffer from the external file system */
...

/* Load font from the buffer */
my_font = lv_binfont_create_from_buffer((void *)buf, bufsize);
if(my_font == NULL) return;
/* Use the font */

/* Free the font if not required anymore */
lv_binfont_destroy(my_font);
```

7.3.4 Motore del Font Tiny TTF

L'estensione `lv_tiny_ttf` consente di utilizzare font TrueType in LVGL tramite la Libreria di Terze Parti `stb_truetype`.

Per un'introduzione dettagliata, vedere: <https://github.com/nothings/stb>.

Utilizzo

Se abilitata in `lv_conf.h` impostando `LV_USE_TINY_TTF` a 1, `lv_tiny_ttf_create_data(data, data_size, font_size)` può essere utilizzata per creare un'istanza di font TTF con l'altezza di riga specificata. È poi possibile utilizzare quel font ovunque sia accettato `lv_font_t`.

Per default, il file TTF o OTF deve essere incorporato come array, in un header o caricato nella RAM per funzionare.

Tuttavia, se `LV_TINY_TTF_FILE_SUPPORT` è abilitato (ad esempio 1), sarà disponibile anche `lv_tiny_ttf_create_file(path, font_size)`, consentendo a `tiny_ttf` di eseguire lo streaming da un file. Il file deve rimanere aperto per tutto il tempo in cui il font viene utilizzato.

Dopo aver creato un font, è possibile modificarne la dimensione in pixel utilizzando `lv_tiny_ttf_set_size(font, font_size)`.

Per default, un font memorizza nella cache i dati per un massimo di 256 elementi glifo per velocizzare il rendering. Questo valore massimo può essere modificato utilizzando `lv_tiny_ttf_create_data_ex(data, data_size, font_size, kerning, cache_size)` o `lv_tiny_ttf_create_file_ex(path, font_size, kerning, cache_size)` (se disponibile). La dimensione della cache è indicata in numero di voci. L'argomento kerning sarà uno dei valori `LV_FONT_KERNING_...`, che indica se consentire o meno il kerning, se supportato.

Esempio

Aprire un font con Tiny TTF da un array di dati

```
#include "../../lv_examples.h"
#if LV_USE_TINY_TTF && LV_BUILD_EXAMPLES

/***
 * Load a font with Tiny_TTF
 */
void lv_example_tiny_ttf_1(void)
{
    extern const uint8_t ubuntu_font[];
    extern const int ubuntu_font_size;

    /*Create style with the new font*/
    static lv_style_t style;
    lv_style_init(&style);
    lv_font_t * font = lv_tiny_ttf_create_data(ubuntu_font, ubuntu_font_size, 30);
    lv_style_set_text_font(&style, font);
    lv_style_set_text_align(&style, LV_TEXT_ALIGN_CENTER);

    /*Create a label with the new style*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_add_style(label, &style, 0);
    lv_label_set_text(label, "Hello world\nI'm a font\ncreated\nwith Tiny TTF");
    lv_obj_center(label);
}
#endif
```

Caricare un font con Tiny_TTF da un file

```
#include "../../lv_examples.h"
#if LV_USE_TINY_TTF && LV_TINY_TTF_FILE_SUPPORT && LV_BUILD_EXAMPLES

/***
 * Load a font with Tiny_TTF from file
 */
void lv_example_tiny_ttf_2(void)
{
    /*Create style with the new font*/
    static lv_style_t style;
    lv_style_init(&style);
    lv_font_t * font = lv_tiny_ttf_create_file("A:lvgl/examples/libs/tiny_ttf/Ubuntu-Medium.ttf", 30);
    lv_style_set_text_font(&style, font);
    lv_style_set_text_align(&style, LV_TEXT_ALIGN_CENTER);

    /*Create a label with the new style*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_add_style(label, &style, 0);
    lv_label_set_text(label, "Hello world\nI'm a font\ncreated\nwith Tiny TTF");
    lv_obj_center(label);
}
#endif
```

Modificare la dimensione del font con Tiny_TTF

```
#include "../../lv_examples.h"
#if LV_USE_TINY_TTF && LV_BUILD_EXAMPLES && LV_USE_OBSERVER

static void font_size_observer_cb(lv_observer_t * observer, lv_subject_t * subject);

static lv_subject_t subject_font;

/***
 * Change font size with Tiny_TTF
 */
void lv_example_tiny_ttf_3(void)
{
    extern const uint8_t ubuntu_font[];
    extern const int ubuntu_font_size;

    lv_subject_init_int(&subject_font, 25);

    /*Create style with the new font*/
    static lv_style_t style;
    lv_style_init(&style);
    lv_font_t * font = lv_tiny_ttf_create_data(ubuntu_font, ubuntu_font_size, 25);
    lv_style_set_text_font(&style, font);
    lv_style_set_text_align(&style, LV_TEXT_ALIGN_CENTER);

    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_obj_center(slider);
    lv_slider_set_range(slider, 5, 50);
    lv_obj_align(slider, LV_ALIGN_BOTTOM_MID, 0, -50);
    lv_slider_bind_value(slider, &subject_font);

    lv_obj_t * slider_label = lv_label_create(lv_screen_active());
    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
    lv_label_bind_text(slider_label, &subject_font, "%d");

    /*Create a label with the new style*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_add_style(label, &style, 0);
    lv_obj_set_size(label, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_label_set_text(label, "Hello world!");
    lv_obj_center(label);

    lv_subject_add_observer(&subject_font, font_size_observer_cb, &style);
}

static void font_size_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
```

(continues on next page)

(continua dalla pagina precedente)

```
{
    lv_style_t * style = (lv_style_t *) lv_observer_get_user_data(observer);
    lv_style_value_t v;
    lv_style_get_prop(style, LV_STYLE_TEXT_FONT, &v);
    lv_font_t * font = (lv_font_t *) v.ptr;
    int32_t size = lv_subject_get_int(subject);

    lv_tiny_ttf_set_size(font, size);
    lv_obj_report_style_change(style);
}
#endif
```

API

lv_tiny_ttf.h

7.3.5 Motore del Font FreeType

FreeType è una libreria software gratuita per il rendering di font.

L'estensione LVGL FreeType è un'interfaccia alla libreria FreeType, che consente di generare bitmap di font in fase di esecuzione dalla maggior parte dei formati di file di font vettoriali e bitmap.

Per un'introduzione dettagliata, consultare: <https://freetype.org/>.

Aggiungere FreeType al Progetto

Per prima cosa, scaricare FreeType dalla cartella `freetype2` (e facoltativamente `freetype-docs` e `freetype-demos`) dal suo [repository ufficiale](#). (Si consiglia la versione più recente).

Se non è stato già fatto, ora è il momento giusto per familiarizzare con l'installazione e la configurazione di questa libreria. Il sito web sopra indicato è un buon punto di partenza, così come il file `README` nella directory principale della versione scaricata.

Esistono due modi per utilizzare FreeType:

Per UNIX

Per i sistemi UNIX-like, si consiglia di compilare e installare le librerie FreeType come segue.

- Accedere alla directory del codice sorgente di FreeType
- `make`
- `sudo make install`
- Aggiungere il path degli include: `/usr/include/freetype2` (per GCC: `-I/usr/include/freetype2 -L/usr/local/lib`)
- Link alla libreria: `freetype` (per GCC: `-L/usr/local/lib -lfreetype`)

Per i Dispositivi Embedded

Per i dispositivi embedded, si consiglia di utilizzare i file di configurazione FreeType forniti da LVGL:

- `lvgl/src/libs/freetype/ftmodule.h`
- `lvgl/src/libs/freetype/ftoption.h`

che includono solo i moduli e le opzioni più comunemente utilizzati, il che è importante per risparmiare spazio FLASH limitato.

- Copiare il codice sorgente di FreeType nella directory del progetto.
- Fare riferimento al seguente [Makefile](#) per la configurazione:

```
# FreeType custom configuration header file
CFLAGS += -DFT2_BUILD_LIBRARY
CFLAGS += -DFT_CONFIG_MODULES_H=<lvgl/src/libs/freetype/ftmodule.h>
CFLAGS += -DFT_CONFIG_OPTIONS_H=<lvgl/src/libs/freetype/ftoption.h>

# FreeType include path
CFLAGS += -Ilibftype/include

# FreeType C source file
FT_CSRCS += freetype/src/base/ftbase.c
FT_CSRCS += freetype/src/base/ftbitmap.c
FT_CSRCS += freetype/src/base/ftdebug.c
FT_CSRCS += freetype/src/base/ftglyph.c
FT_CSRCS += freetype/src/base/ftinit.c
FT_CSRCS += freetype/src/base/ftstroke.c
FT_CSRCS += freetype/src/cache/ftcache.c
FT_CSRCS += freetype/src/gzip/tgzip.c
FT_CSRCS += freetype/src/sfnt/sfnt.c
FT_CSRCS += freetype/src/smooth/smooth.c
FT_CSRCS += freetype/src/truetype/truetype.c
CSRC += $(FT_CSRCS)
```

Utilizzo

Impostare `LV_USE_FREETYPE` in `lv_conf.h` a 1.

Configurazione della cache:

- `LV_FREETYPE_CACHE_FT_GLYPH_CNT` Numero massimo di glifi memorizzati nella cache, ecc.

Per default, l'estensione FreeType non utilizza il file system di LVGL. È sufficiente passare il path del font come di consueto sul sistema operativo o sulla piattaforma.

Se si desidera che FreeType utilizzi l'allocazione di memoria e l'interfaccia del file system di LVGL, è possibile abilitare `LV_FREETYPE_USE_LVGL_PORT` in `lv_conf.h`, utile per una gestione unificata. In questo caso, sarà necessario escludere dalla compilazione il file `ftsystem.c` della libreria FreeType configurata, poiché `lv_ftsystem.c` di LVGL contiene versioni personalizzate delle funzioni definite al suo interno.

Lo stile del font supporta il font **Corsivo** e il **Grassetto** elaborati dal software e può essere impostato utilizzando i seguenti valori, laddove richiesti:

- `LV_FREETYPE_FONT_STYLE_NORMAL`: Stile di default.
- `LV_FREETYPE_FONT_STYLE_ITALIC`: Stile corsivo.
- `LV_FREETYPE_FONT_STYLE_BOLD`: Stile grassetto.

Questi valori possono essere combinati, ad esempio `LV_FREETYPE_FONT_STYLE_BOLD | LV_FREETYPE_FONT_STYLE_ITALIC`.

L'estensione FreeType supporta anche glifi bitmap colorati, come gli emoji. Si noti che al momento sono supportati solo i bitmap. Non è possibile eseguire il rendering di grafica vettoriale colorata. Di seguito è riportato un esempio su come disegnare un glifo bitmap colorato.

Utilizzare la funzione `lv_ftype_font_create()` per creare un font. Per eliminare un font, si usa `lv_ftype_font_delete()`. Per un utilizzo più dettagliato, fare riferimento al codice di esempio riportato di seguito.

Ulteriori Informazioni

Tutorial su FreeType

Il rendering dei font vettoriali è supportato da VGLite o ThorVG; quando si utilizzano font vettoriali con ThorVG, è possibile impostare un contorno delle lettere di un colore diverso.

Ciò si ottiene impostando gli attributi di stile con le funzioni `lv_style_set_text_outline_width()` e `lv_style_set_text_outline_color()`

Bisognerà tenere conto dell'aumento di larghezza e altezza delle lettere dovuto all'aggiunta del contorno; per evitare sovrapposizioni, distanziarle utilizzando `lv_style_set_text_letter_space()`

Per utilizzare font vettoriali con ThorVG, è necessario abilitare `LV_USE_VECTOR_GRAPHICS` in `lv_conf.h`

i Nota

Questa funzionalità è attualmente sperimentale e presenta problemi di clipping, soprattutto quando si utilizzano font di grandi dimensioni.

Vedere la funzione `lv_example_freetype_2_vector_font()` per un esempio di utilizzo.

Esempi

Creare un font con FreeType

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_FREETYPE

#if LV_FREETYPE_USE_LVGL_PORT
#define PATH_PREFIX "A:"
#else
#define PATH_PREFIX "./"
#endif

/***
 * Load a font with FreeType
 */
void lv_example_freetype_1(void)
{
    /*Create a font*/
    lv_font_t * font = lv_freetype_font_create(PATH_PREFIX "lvgl/examples/libs/freetype/Lato-Regular.ttf",
                                                LV_FREETYPE_FONT_RENDER_MODE_BITMAP,
                                                24,
                                                LV_FREETYPE_FONT_STYLE_NORMAL);

    if(!font) {
        LV_LOG_ERROR("freetype font create failed.");
        return;
    }

    /*Create style with the new font*/
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_text_font(&style, font);
    lv_style_set_text_align(&style, LV_TEXT_ALIGN_CENTER);

    /*Create a label with the new style*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_add_style(label, &style, 0);
    lv_label_set_text(label, "Hello world\nI'm a font created with FreeType");
    lv_obj_center(label);
}

void lv_example_freetype_1(void)
{
    /*TODO
     *fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "FreeType is not installed");
    lv_obj_center(label);
}

#endif
#endif
```

Utilizzare un font bitmap per disegnare Emoji con FreeType

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_FREETYPE

#if LV_FREETYPE_USE_LVGL_PORT
#define PATH_PREFIX "A:"
#else
#define PATH_PREFIX "./"
#endif

/*
 * Load a vector font
 * ThorVG needs to be enabled, LV_USE_VECTOR_GRAPHICS=1
 */
void lv_example_freetype_2_vector_font(uint32_t font_size, uint32_t border_width)
{
    lv_font_t * font = lv_freetype_font_create(PATH_PREFIX "lvgl/examples/libs/freetype/Lato-Regular.ttf",
                                                LV_FREETYPE_FONT_RENDER_MODE_OUTLINE,
                                                font_size,
                                                LV_FREETYPE_FONT_STYLE_NORMAL);

    if(!font) {
        LV_LOG_ERROR("Freetype font create failed.");
        return;
    }

    /*Create style with the new font*/
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_text_font(&style, font);
```

(continues on next page)

(continua dalla pagina precedente)

Crenatura dei font FreeType

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_FREETYPE

#if LV_FREETYPE_USE_LVGL_PORT
    #define PATH_PREFIX "A:"
#else
    #define PATH_PREFIX "./"
#endif

#define KERNING_TEST_TEXT "AWAY,ToTaTe"

static void create_label(lv_font_kerning_t kerning, int32_t y_ofs, const char * text)
```

(continues on next page)

(continua dalla pagina precedente)

```
{
    lv_font_info_t font_info;
    lv_freetype_init_font_info(&font_info);
    font_info.name = PATH_PREFIX "lvgl/examples/libs/freetype/Lato-Regular.ttf";
    font_info.size = 32;
    font_info.kerning = kerning;

    lv_font_t * font = lv_freetype_font_create_with_info(&font_info);
    if(!font) {
        LV_LOG_ERROR("freetype font create failed.");
        return;
    }

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_style_text_font(label, font, 0);
    lv_label_set_text(label, text);
    lv_obj_align(label, LV_ALIGN_CENTER, 0, y_ofs);
}

/**
 * FreeType kerning example
 */
void lv_example_freetype_3(void)
{
    create_label(LV_FONT_KERNING_NONE, -32, "Kerning None: " KERNING_TEST_TEXT);
    create_label(LV_FONT_KERNING_NORMAL, 32, "Kerning Normal: " KERNING_TEST_TEXT);
}
#else

void lv_example_freetype_3(void)
{
    /*TODO
     *fallback for online examples*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "FreeType is not installed");
    lv_obj_center(label);
}

#endif
#endif
}
```

API

lv_freetype.h

lv_freetype_private.h

7.3.6 Font immagine

Disegna l'immagine in un oggetto **label** o **span** con **lv_imgfont**. Questo viene spesso utilizzato per visualizzare le icone emoji Unicode nel testo.

Formati immagine supportati: determinati dai *decoder di immagini* LVGL abilitati.

Utilizzo

Abilitare **LV_USE_IMGFONT** in **lv_conf.h**.

Per creare un nuovo *imgfont*:

```
static lv_font_t * imgfont;
...
imgfont = lv_imgfont_create(height, path_cb, user_data);
```

- **height** Dimensione del font.
- **path_cb** Funzione per ottenere il path dell'immagine di un carattere. Si passa **NULL** se non deve essere visualizzata alcuna immagine, ma il carattere stesso.
- **user_data** Puntatore ai dati utente.

Per utilizzare *imgfont* in un'etichetta, si fa riferimento ad esso: **lv_obj_set_style_text_font(label, imgfont, LV_PART_MAIN)**

Per eliminare *imgfont* che non viene più utilizzato, si usa **lv_imgfont_destroy(imgfont)**.

Esempio

Utilizzare emoji in un testo.

```
#include "../../lv_examples.h"

#if LV_BUILD_EXAMPLES
#if LV_USE_IMGFONT

static const void * get_imgfont_path(const lv_font_t * font, uint32_t unicode, uint32_t unicode_next,
                                     int32_t * offset_y, void * user_data)
{
    LV_UNUSED(font);
    LV_UNUSED(unicode_next);
    LV_UNUSED(offset_y);
    LV_UNUSED(user_data);

    LV_IMAGE_DECLARE(emoji_F617);

    if(unicode < 0x0000) return NULL;

    if(unicode == 0xF617) {
        return &emoji_F617;
    }
    else if(unicode == 0xF600) {
#if LV_USE_FFMPEG
        return "lvgl/examples/assets/emoji/F600.png";
#else
        return "A:lvgl/examples/assets/emoji/F600.png";
#endif
    }

    return NULL;
}

/**
 * draw img in label or span obj
 */
void lv_example_imgfont_1(void)
{
    lv_font_t * imgfont = lv_imgfont_create(80, get_imgfont_path, NULL);
    if(imgfont == NULL) {
        LV_LOG_ERROR("imgfont init error");
        return;
    }

    imgfont->fallback = LV_FONT_DEFAULT;

    lv_obj_t * label1 = lv_label_create(lv_screen_active());
    lv_label_set_text(label1, "12\uF600\uF617AB");
    lv_obj_set_style_text_font(label1, imgfont, LV_PART_MAIN);
    lv_obj_center(label1);
}
#else

void lv_example_imgfont_1(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "imgfont is not installed");
    lv_obj_center(label);
}
#endif
#endif

```

API

lv_imgfont.h

7.3.7 Font BDF

Panoramica

I display piccoli a bassa risoluzione non sono gradevoli da vedere con i font renderizzati automaticamente. Un font bitmap offre la soluzione, ma è necessario convertire il font bitmap (BDF) in un TTF.

Convertire BDF in TTF

I BDF sono font bitmap in cui i font non sono descritti in contorni, ma in pixel. È possibile utilizzare file BDF, ma devono essere convertiti in formato TTF utilizzando `mkttf`, disponibile in questo repository GitHub: <https://github.com/Tblue/mkttf>. Questo strumento utilizza "potrace" per generare contorni dalle informazioni bitmap. Anche la bitmap stessa verrà incorporata nel TTF. `lv_font_conv` utilizza la bitmap incorporata, ma necessita anche dei contorni. Si potrebbe pensare di poter usare un finto MS Bitmap solo sfnt (ttf) (TTF senza contorni) creato da fontforge, ma non funzionerà.

Installare imagemagick, python3, python3-fontforge e potrace

Sui Sistemi Ubuntu, digitare semplicemente

```
sudo apt install imagemagick python3-fontforge potrace
```

Clonare mkttf

```
git clone https://github.com/Tblue/mkttf
```

Leggere la documentazione di mkttf.

Le versioni precedenti di imagemagick richiedono la chiamata a imagemagick prima di convert, identify e così via. Ma le versioni più recenti non lo richiedono. Quindi si potrebbero dover modificare 2 righe in potrace-wrapper.sh: aprire potrace-wrapper.sh e rimuovere imagemagick dalla riga 55 e dalla riga 64:

riga 55

```
wh=$(identify -format '%[width]pt %[height]pt' "${input?}")
```

riga 64

```
convert "${input?}" -sample '1000%' - \
```

Potrebbe essere necessario modificare lo script mkttf.py.

riga 1

```
#!/usr/bin/env python3
```

Esempio

```
cd mkttf
./mkttf.py ./TerminusMedium-12-12.bdf
Importing bitmaps from 0 additional fonts...
Importing font './TerminusMedium-12-12.bdf' into glyph background...
Processing glyphs...
Saving TTF file...
Saving SFD file...
Done!
```

Il TTF TerminusMedium-001.000.ttf verrà creato da ./TerminusMedium-12-12.bdf.

Per creare un font per LVGL:

```
lv_font_conv --bpp 1 --size 12 --no-compress --font TerminusMedium-001.000.ttf --range 0x20-0x7e,0xa1-0xff --format lvgl -o terminus_1bpp_12px.c
```

nota

Si usa 1 bpp perché non usiamo l'anti-aliasing. Non risulta nitido sui display a bassa risoluzione.

7.3.8 Supporto Bidirezionale

Panoramica

La maggior parte delle lingue utilizza la direzione di scrittura da sinistra a destra (LTR in breve), tuttavia alcune lingue (come l'ebraico, il persiano o l'arabo) utilizzano la direzione da destra a sinistra (RTL in breve).

LVGL non supporta solo il testo RTL, ma supporta anche il rendering misto (noto anche come bidirezionale, BiDi). Alcuni esempi:

The names of these states in Arabic
are الكويت and مصر respectively.

The title is مفتاح معايير الويب! in Arabic.

Il supporto BiDi è abilitato impostando `LV_USE_BIDI` su un valore diverso da zero in `lv_conf.h`.

Tutto il testo ha una direzione di base (LTR o RTL) che determina alcune regole di rendering e l'allineamento di default del testo (sinistra o destra). Tuttavia, in LVGL, la direzione di base non viene applicata solo alle "label"

[etichette]. È una proprietà generale che può essere impostata per ogni Widget. Se non impostata, verrà ereditata dal padre. Ciò significa che è sufficiente impostare la direzione di base di una schermata e i Widget figli la erediteranno.

La direzione di base di default per le schermate può essere impostata tramite `LV_BIDI_BASE_DIR_DEF` in `lv_conf.h` e gli altri Widget ereditano la direzione di base dal loro genitore.

Per impostare la direzione di base di un Widget, si usa `lv_obj_set_style_base_dir(widget, base_dir, selector)`. Le possibili direzioni di base sono:

- `LV_BASE_DIR_LTR`: Direzione di base da sinistra a destra
- `LV_BASE_DIR_RTL`: Direzione di base da destra a sinistra
- `LV_BASE_DIR_AUTO`: Rilevamento automatico della direzione di base

Questo elenco riassume l'effetto della direzione di base RTL sui widget:

- Creare widget per impostazione di default a destra
- `lv_tabview`: Visualizza le schede da destra a sinistra
- `lv_checkbox`: Mostra la casella a destra
- `lv_buttonmatrix`: Ordina i pulsanti da destra a sinistra
- `lv_list`: Mostra le icone a destra
- `lv_dropdown`: Allinea le opzioni a destra
- Le stringhe di testo in `lv_table`, `lv_buttonmatrix`, `lv_keyboard`, `lv_tabview`, `lv_dropdown`, `lv_roller` sono "elaborati BiDi" per essere visualizzati correttamente

Supporto per arabo e persiano

Esistono alcune regole speciali per visualizzare i caratteri arabi e persiani: la *forma* di un carattere dipende dalla sua posizione nel testo. È necessario utilizzare una forma diversa della stessa lettera quando è isolata, all'inizio, al centro o alla fine. Oltre a queste, è necessario tenere conto anche di alcune regole di congiunzione.

LVGL supporta queste regole se `LV_USE_ARABIC_PERSIAN_CHARS` è abilitato in `lv_conf.h`.

Tuttavia, ci sono alcune limitazioni:

- È supportata solo la visualizzazione del testo (ad esempio sulle etichette), gli input di testo (ad esempio, Text Area) non supportano questa funzionalità.
- Il testo statico (ad esempio, `const`) non viene elaborato. Ad esempio Il testo impostato da `lv_label_set_text()` sarà "elaborato in arabo", ma `lv_label_set_text_static()` non lo sarà.
- Le funzioni di acquisizione del testo (ad esempio `lv_label_get_text()`) restituiranno il testo elaborato.

7.3.9 Aggiungere un Nuovo Motore di Font

Panoramica

I font hanno una proprietà **format**. Descrive come vengono memorizzati i dati del glifo. Al momento in cui scriviamo, questo campo può assumere diversi valori, suddivisi in due categorie:

1. **Bitmap based**: 1, 2, 4 o 8 bpp e formato immagine, e
2. vettore **Vector based**, SVG; per quest'ultimo, l'utente fornisce la logica di rendering.

Per i formati semplici:

- il font è memorizzato come un array di bitmap, una bitmap per glifo;
- il valore memorizzato per ciascun pixel determina l'opacità del pixel, consentendo bordi più smussati: valori bpp più elevati producono bordi più smussati.

Per i formati avanzati, le informazioni sul font vengono memorizzate nel rispettivo formato.

La proprietà **format** influisce anche sulla quantità di memoria necessaria per memorizzare un font. Ad esempio, `format = LV_FONT_GLYPH_FORMAT_A4` rende un font quasi quattro volte più grande rispetto a `format = LV_FONT_GLYPH_FORMAT_A1`.

Esempio

L'interfaccia dei font di LVGL è progettata per essere molto flessibile, ma, nonostante ciò, è possibile aggiungere un proprio motore di font al posto di quello interno di LVGL. Ad esempio, è possibile utilizzare [FreeType](#) per il rendering in tempo reale dei glifi dei font TTF o utilizzare una flash esterna per memorizzare la bitmap del font e leggerla quando la libreria ne ha bisogno. FreeType può essere utilizzato in LVGL come descritto in [Freetype](#).

Per aggiungere un nuovo motore di font, è necessario creare una variabile personalizzata `lv_font_t`:

```
/* Describe the properties of a font */
lv_font_t my_font;
my_font.get_glyph_dsc = my_get_glyph_dsc_cb; /* Set a callback to get info about glyphs */
my_font.get_glyph_bitmap = my_get_glyph_bitmap_cb; /* Set a callback to get bitmap of a glyph */
my_font.line_height = height; /* The real line height where any text fits */
my_font.base_line = base_line; /* Base line measured from the top of line_height */
my_font.dsc = something_required; /* Store any implementation specific data here */
my_font.user_data = user_data; /* Optionally some extra user data */

...
/* Get info about glyph of 'unicode_letter' in 'font' font.
 * Store the result in 'dsc_out'.
 * The next letter ('unicode_letter_next') might be used to calculate the width required by this glyph (kerning)
 */
bool my_get_glyph_dsc_cb(const lv_font_t * font, lv_font_glyph_dsc_t * dsc_out, uint32_t unicode_letter, uint32_t unicode_letter_next)
{
    /* Your code here */

    /* Store the result.
     * For example ...
     */
    dsc_out->adv_w = 12; /* Horizontal space required by the glyph in [px] */
    dsc_out->box_h = 8; /* Height of the bitmap in [px] */
    dsc_out->box_w = 6; /* Width of the bitmap in [px] */
    dsc_out->ofs_x = 0; /* X offset of the bitmap in [px] */
    dsc_out->ofs_y = 3; /* Y offset of the bitmap measured from the as line */
    dsc_out->format= LV_FONT_GLYPH_FORMAT_A2;

    return true; /* true: glyph found; false: glyph was not found */
}

/* Get the bitmap of 'unicode_letter' from 'font'. */
const uint8_t * my_get_glyph_bitmap_cb(const lv_font_t * font, uint32_t unicode_letter)
{
    /* Your code here */

    /* The bitmap should be a continuous bitstream where
     * each pixel is represented by `bpp` bits */

    return bitmap; /* Or NULL if not found */
}
```

7.3.10 Font Manager

Il Font Manager è una incapsulazione secondaria del motore dei font LVGL, che facilita l'uso e la gestione dei font per le applicazioni. Le funzioni di gestione font attualmente disponibili includono:

- Conteggio dei riferimenti alle risorse font (riduce la creazione ripetuta di risorse font).
- Concatenazione delle risorse dei font (font alternativi).
- Meccanismo di riciclo delle risorse font (memorizza nel buffer le risorse font eliminate di recente per ridurre il tempo di creazione ripetuto).

Utilizzo

Si abilitano FreeType e Font Manager in `lv_conf.h` impostando le macro `LV_USE_FONT_MANAGER` su valori diversi da zero e configurando `LV_FONT_MANAGER_NAME_MAX_LEN` per impostare la lunghezza massima del nome del font.

Inizializzare Font Manager

Si usa `lv_font_manager_create()` per creare un font manager, dove il parametro `recycle_cache_size()` viene utilizzato per impostare il numero di cache di riciclo dei font, il che può migliorare l'efficienza della creazione dei font.

Si usa `lv_font_manager_add_src_static()` per aggiungere una mappatura tra i nomi dei font e le risorse font per indicare al font manager come accedere alle risorse font. Nota: se la struttura di descrizione delle risorse font non è allocata staticamente (ad esempio, allocata da una variabile locale), si usa `lv_font_manager_add_src()` per aggiungere la risorsa. Questa funzione copierà il contenuto della struttura stessa.

Utilizzare `lv_font_manager_remove_src()` per rimuovere la mappatura delle risorse font.

È importante notare che il parametro `src` deve corrispondere rigorosamente a `class_p`. `class_p` influenzera il modo in cui il gestore dei font interpreta `src`. Se viene passato un parametro incompatibile, il programma potrebbe non funzionare. Per le classi di font attualmente supportate, fare riferimento al codice di esempio.

```
static lv_font_manager_t * g_font_manager = NULL;

void font_manager_init(void)
{
    /* Create font manager, with 8 fonts recycling buffers */
    g_font_manager = lv_font_manager_create(8);

    /* Add font path mapping to font manager */
    lv_font_manager_add_src_static(g_font_manager, "Lato-Regular", "./lvgl/examples/libs/freetype/Lato-Regular.ttf", &lv_freetype_font_class);

    char path[] = "/path/to/myfont.ttf";
    lv_font_manager_add_src(g_font_manager, "MyFont", path, &lv_freetype_font_class);
}
```

Creare Font da Font Manager

I parametri verranno passati alla funzione di creazione font del backend font, come `lv_freetype_font_create()` e `lv_tiny_ttf_create_file()`. Il backend font selezionerà autonomamente i parametri supportati e ignorerà quelli non supportati.

Il parametro `font_family` può essere compilato con i nomi di più font (separati da , [virgola]) per ottenere la concatenazione dei font (quando il glifo corrispondente non viene trovato in un file font, la ricerca verrà eseguita automaticamente a partire dal font concatenato successivo).

```
static lv_font_t * g_font = NULL;

/* Create font from font manager */
lv_font_t * g_font = lv_font_manager_create_font(g_font_manager,
                                                "Lato-Regular,MyFont",
                                                LV_FREETYPE_FONT_RENDER_MODE_BITMAP,
                                                24,
                                                LV_FREETYPE_FONT_STYLE_NORMAL,
                                                LV_FONT_KERNING_NONE);

/* Handle error case */
if(g_font == NULL) {
    g_font = (lv_font_t *)LV_FONT_DEFAULT;
}

/* Create label with the font */
lv_obj_t * label = lv_label_create(lv_screen_active());
lv_obj_set_style_text_font(label, g_font, 0);
lv_label_set_text(label, "Hello World!");
```

Eliminare Font

Si usa `lv_font_manager_delete_font()` per eliminare il font quando non è più necessario. Il font manager contrasseggerà la risorsa font come font riciclabile in modo che possa essere creata più rapidamente la prossima volta.

Si noti che è necessario eliminare prima tutti i Widget che utilizzavano il font, poi eliminare il font per evitare di accedere a puntatori indesiderati.

```
/* Delete label and font */
lv_obj_delete(label);
lv_font_manager_delete_font(g_font_manager, g_font);
g_font = NULL;
```

Eliminare il Font Manager

Usare `lv_font_manager_delete()` per eliminare l'intero font manager. Prima di procedere all'eliminazione, è necessario assicurarsi che l'applicazione abbia eliminato tutti i font utilizzando `lv_font_manager_delete_font()`. Il font manager verificherà lo stato di riferimento di tutti i font allocati. Se sono ancora presenti font referenziati, il gestore font non verrà eliminato e la funzione restituirà false.

Esempio

Creare un Font FreeType

```
#include "../../lv_examples.h"
#if LV_USE_FONT_MANAGER && LV_USE_FREETYPE && LV_BUILD_EXAMPLES

#if LV_FREETYPE_USE_LVGL_PORT
#define PATH_PREFIX "A:"
#else
#define PATH_PREFIX "./"
#endif

static lv_font_manager_t * g_font_manager = NULL;

void lv_example_font_manager_1(void)
{
    /* Create font manager, with 8 fonts recycling buffers */
    g_font_manager = lv_font_manager_create(8);

    /* Add font path mapping to font manager */
    lv_font_manager_add_src_static(g_font_manager,
        "Lato-Regular",
        PATH_PREFIX "lvgl/examples/libs/freetype/Lato-Regular.ttf",
        &lv_freetype_font_class);

    /* Create font from font manager */
    lv_font_t * font = lv_font_manager_create_font(g_font_manager,
        "Lato-Regular",
        LV_FREETYPE_FONT_RENDER_MODE_BITMAP,
        24,
        LV_FREETYPE_FONT_STYLE_NORMAL,
        LV_FONT_KERNING_NONE);

    if(!font) {
        LV_LOG_ERROR("Could not create font");
        return;
    }

    /* Create label with the font */
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_style_text_font(label, font, 0);
    lv_label_set_text(label, "Hello Font Manager!");
    lv_obj_center(label);
}

#else

void lv_example_font_manager_1(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Font Manager or FreeType is not enabled");
    lv_obj_center(label);
}

#endif
```

Creare una Famiglia di Font

```
#include "../../lv_examples.h"
#if LV_USE_FONT_MANAGER && LV_BUILD_EXAMPLES

#if defined(LV_FREETYPE_USE_LVGL_PORT) && LV_FREETYPE_USE_LVGL_PORT
#define PATH_PREFIX "A:"
#else
#define PATH_PREFIX "./"
#endif

static lv_font_manager_t * g_font_manager = NULL;

void lv_example_font_manager_2(void)
{
    /* Create font manager, with 8 fonts recycling buffers */
    g_font_manager = lv_font_manager_create(8);

#if LV_FONT_MONTserrat_14 && LV_FONT_MONTserrat_32

    /* Register built-in font sources */
    static lv_builtin_font_src_t builtin_font_src[3] = { 0 };
    builtin_font_src[0].font_p = &lv_font_montserrat_14;
    builtin_font_src[0].size = 14;
    builtin_font_src[1].font_p = &lv_font_montserrat_32;
    builtin_font_src[1].size = 32;

    /* IMPORTANT! Marking the end of the array */
    builtin_font_src[2].font_p = NULL;
    builtin_font_src[2].size = 0;

    lv_font_manager_add_src_static(g_font_manager,
        "Montserrat",
        builtin_font_src,
        &lv_builtin_font_class);
#endif

#if LV_USE_FREETYPE
    /* Register FreeType font source */
    lv_font_manager_add_src_static(g_font_manager,
        "NotoColorEmoji",
        PATH_PREFIX "lvgl/examples/libs/freetype/NotoColorEmoji-32.subset.ttf",
        &lv_freetype_font_class);
#endif

#if LV_USE_TINY_TTF && LV_TINY_TTF_FILE_SUPPORT
    /* Register TinyTTF font source */
    static const lv_tiny_ttf_font_src_t tiny_ttf_font_src = {
        .path = ":lvgl/examples/libs/tiny_ttf/Ubuntu-Medium.ttf",
        .data = NULL,

```

(continues on next page)

(continua dalla pagina precedente)

```

    .data_size = 0,
    .cache_size = 0,
};

lv_font_manager_add_src_static(g_font_manager,
    "Ubuntu-Medium",
    &tiny_ttf_font_src,
    &lv_tiny_ttf_font_class);

#endif

/* Create font from font manager */
lv_font_t * font = lv_font_manager_create_font(g_font_manager,
    "Ubuntu-Medium,NotoColorEmoji,Montserrat",
    LV_FREETYPE_FONT_RENDER_MODE_BITMAP,
    32,
    LV_FREETYPE_FONT_STYLE_NORMAL,
    LV_FONT_KERNING_NONE);

if(!font) {
    LV_LOG_ERROR("Could not create font");
    return;
}

/* Create label with the font */
lv_obj_t * label = lv_label_create(lv_screen_active());
lv_obj_set_style_text_font(label, font, 0);
lv_label_set_text(label, "Hello Font Manager! □ " LV_SYMBOL_OK);
lv_obj_center(label);
}

#else

void lv_example_font_manager_2(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Font Manager is not enabled");
    lv_obj_center(label);
}

#endif

```

Creare un Font Immagine Custom

```

#include "../../lv_examples.h"
#if LV_USE_FONT_MANAGER && LV_USE_FREETYPE && LV_USE_IMGFONT && LV_BUILD_EXAMPLES

#ifndef PATH_PREFIX
#define PATH_PREFIX "A/"
#else
#define PATH_PREFIX "./"
#endif

typedef struct {
    uint16_t match_size_min;
    uint16_t match_size_max;
    lv_imgfont_get_path_cb_t get_path_cb;
} imgfont_src_t;

static lv_font_manager_t * g_font_manager = NULL;

static const void * get_imgfont_path(const lv_font_t * font, uint32_t unicode, uint32_t unicode_next,
    int32_t * offset_y, void * user_data)
{
    LV_UNUSED(font);
    LV_UNUSED(unicode_next);
    LV_UNUSED(offset_y);
    LV_UNUSED(user_data);

    LV_IMAGE_DECLARE(emoji_F617);

    if(unicode == 0xF617) {
        return &emoji_F617;
    }

    return NULL;
}

static lv_font_t * imgfont_create_cb(const lv_font_info_t * info, const void * src)
{
    const imgfont_src_t * imgfont_src = (imgfont_src_t *) src;

    if(info->size < imgfont_src->match_size_min
        || info->size > imgfont_src->match_size_max) {
        return NULL;
    }

    return lv_imgfont_create(info->size, imgfont_src->get_path_cb, NULL);
}

static void imgfont_delete_cb(lv_font_t * font)
{
    lv_imgfont_destroy(font);
}

static void * imgfont_dup_src_cb(const void * src)
{
    imgfont_src_t * imgfont_src = (imgfont_src_t *) lv_malloc(sizeof(imgfont_src_t));
    LV_ASSERT_MALLOC(imgfont_src);
    lv_memcpy(imgfont_src, src, sizeof(imgfont_src_t));
    return imgfont_src;
}

static void imgfont_free_src_cb(void * src)
{
    lv_free(src);
}

static const lv_font_class_t imgfont_class = {

```

(continues on next page)

(continua dalla pagina precedente)

```

.create_cb = imgfont_create_cb,
.delete_cb = imgfont_delete_cb,
.dup_src_cb = imgfont_dup_src_cb,
.free_src_cb = imgfont_free_src_cb,
};

void lv_example_font_manager_3(void)
{
    /* Create font manager, with 8 fonts recycling buffers */
    g_font_manager = lv_font_manager_create(8);

    /* Add font path mapping to font manager */
    lv_font_manager_add_src_static(g_font_manager,
        "Lato-Regular",
        PATH_PREFIX "lvgl/examples/libs/freetype/Lato-Regular.ttf",
        &lv_freetype_font_class);

    /* Add custom imgfont source to font manager */
    static const imgfont_src_t imgfont_src = {
        .match_size_min = 70,
        .match_size_max = 80,
        .get_path_cb = get_imgfont_path,
    };
    lv_font_manager_add_src_static(g_font_manager,
        "Emoji",
        &imgfont_src,
        &imgfont_class);

    /* Create font from font manager */
    lv_font_t * font = lv_font_manager_create_font(g_font_manager,
        "Lato-Regular", "Emoji",
        LV_FREETYPE_FONT_RENDER_MODE_BITMAP,
        75,
        LV_FREETYPE_FONT_STYLE_NORMAL,
        LV_FONT_KERNING_NONE);

    if(!font) {
        LV_LOG_ERROR("Could not create font");
        return;
    }

    /* Create label with the font */
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_style_text_font(label, font, 0);
    lv_label_set_text(label, "Quiet\uF617");
    lv_obj_center(label);
}

#else

void lv_example_font_manager_3(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Font Manager or Freetype or Imgfont is not enabled");
    lv_obj_center(label);
}

#endif

```

API

lv_font_manager.h

lv_font_manager_recycle.h

7.4 Immagini (lv_image)

Un'immagine può essere un file o una variabile che memorizza la bitmap stessa e alcuni metadati.

7.4.1 Immagini Memorizzate

È possibile memorizzare le immagini in due posizioni:

- come variabile nella memoria interna (RAM o ROM)
- come file

Variabili

Le immagini memorizzate internamente in una variabile sono composte principalmente da una struttura *lv_image_dsc_t* con i seguenti campi:

- **header:**
 - *cf*: Formato colore. Vedere *di seguito*
 - *w*: larghezza in pixel (<= 2048)
 - *h*: altezza in pixel (<= 2048)

- *always zero*: 3 bit che devono essere sempre zero
- *reserved*: riservato per uso futuro
- **data**: puntatore a un array in cui è memorizzata l'immagine stessa
- **data_size**: lunghezza di **data** in byte

Queste vengono solitamente memorizzate all'interno di un progetto come file C. Sono linkate all'eseguibile risultante come qualsiasi altro dato costante.

File

Per gestire i file è necessario aggiungere un *Drive* di archiviazione a LVGL. In breve, un *Drive* è un insieme di funzioni (*open*, *read*, *close*, ecc.) registrate in LVGL per eseguire operazioni sui file. È possibile aggiungere un'interfaccia a un file system standard (FAT32 su scheda SD) oppure creare un file system semplice per leggere i dati da una memoria Flash SPI. In ogni caso, un *Drive* è solo un'astrazione per leggere e/o scrivere dati in memoria. Consultare la sezione [File system](#) per ulteriori informazioni.

Le immagini memorizzate come file non sono linkate all'eseguibile risultante e devono essere lette nella RAM prima di essere disegnate. Di conseguenza, non consumano così tante risorse come le immagini linkate in fase di compilazione. Tuttavia, sono più facili da sostituire senza dover ricostruire il programma principale.

7.4.2 Formati Colore

Sono supportati vari formati nativi di colore:

- ***LV_COLOR_FORMAT_NATIVE***: Memorizza semplicemente i colori RGB (in qualsiasi profondità di colore per cui è configurato LVGL).
- ***LV_COLOR_FORMAT_NATIVE_WITH_ALPHA***: Come ***LV_COLOR_FORMAT_NATIVE*** ma aggiunge anche un byte alfa (trasparenza) per ogni pixel.
- ***LV_COLOR_FORMAT_I1*, *LV_COLOR_FORMAT_I2*, *LV_COLOR_FORMAT_I4*, *LV_COLOR_FORMAT_I8***: Utilizza una paletta con 2, 4, 16 o 256 colori e memorizza ogni pixel in 1, 2, 4 o 8 bit.

Poiché i colori nella paletta possono essere qualsiasi, i dati pixel per questi formati vengono memorizzati come ARGB8888.

Quando si utilizza il rendering SW di LVGL, assicurarsi che **LV_DRAW_SW_SUPPORT_ARGB8888** sia impostato a 1 nella configurazione di LVGL.

- ***LV_COLOR_FORMAT_A1*, *LV_COLOR_FORMAT_A2*, *LV_COLOR_FORMAT_A4*, *LV_COLOR_FORMAT_A8***: **Memorizzano il valore alfa solo con 1, 2, 4 o 8 bit**. I pixel assumono il colore di **style.img_recolor** e l'opacità imposta. L'immagine sorgente deve essere un canale alfa. Questo è ideale per bitmap simili ai font in cui l'intera immagine è un unico colore modificabile.

I byte delle immagini ***LV_COLOR_FORMAT_NATIVE*** vengono memorizzati nel seguente ordine.

- Profondità colore a 32 bit:
 - **Byte 0**: Blu
 - **Byte 1**: Verde
 - **Byte 2**: Rosso
 - **Byte 3**: Alfa (solo con ***LV_COLOR_FORMAT_NATIVE_WITH_ALPHA***)
- Profondità colore a 16 bit:
 - **Byte 0**: Verde 3 bit inferiori, Blu 5 bit
 - **Byte 1**: Rosso 5 bit, Verde 3 bit superiori
 - **Byte 2**: Alfa (solo con ***LV_COLOR_FORMAT_NATIVE_WITH_ALPHA***)
- Profondità colore a 8 bit:
 - **Byte 0**: Rosso 3 bit, Verde 3 bit, Blu 2 bit

– **Byte 2:** Alfa (solo con `LV_COLOR_FORMAT_NATIVE_WITH_ALPHA`)

È possibile memorizzare le immagini in formato *Raw* per indicare che non sono codificate con uno dei formati colore nativi ed è necessario utilizzare un *Decoder di immagini*.

- `LV_COLOR_FORMAT_RAW`: Indica un'immagine raw di base (ad esempio, un'immagine PNG o JPG).
- `LV_COLOR_FORMAT_RAW_ALPHA`: Indica che un'immagine ha un canale alfa e che per ogni pixel viene aggiunto un byte.

7.4.3 Aggiunta e Utilizzo di Immagini

È possibile aggiungere immagini a LVGL in due modi:

- utilizzando il convertitore online
- creando manualmente le immagini

Convertitore online

Il convertitore di immagini online è disponibile qui: <https://lvgl.io/tools/imageconverter>

Aggiungere un'immagine a LVGL tramite il convertitore online è semplice.

1. È necessario selezionare prima un'immagine *BMP*, *PNG* o *JPG*.
2. Assegnare all'immagine un nome che verrà utilizzato in LVGL.
3. Selezionare il *Formato colore*.
4. Selezionare il tipo di immagine desiderato. Scegliendo un binario verrà generato un file `.bin` che deve essere salvato separatamente e letto utilizzando il *supporto dei file*. Scegliendo una variabile verrà generato un file C standard che può essere linkato al progetto.
5. Premere il pulsante *Convert*. Una volta completata la conversione, il browser scaricherà automaticamente il file risultante.

Negli array C generati (variabili), le bitmap per tutte le profondità di colore (1, 8, 16 o 32) sono incluse nel file C, ma solo la profondità di colore che corrisponde a `LV_COLOR_DEPTH` in `lv_conf.h` verrà effettivamente linkata all'eseguibile risultante.

Nel caso di file binari, è necessario specificare il formato colore desiderato:

- RGB332 per profondità di colore a 8 bit
- RGB565 per profondità di colore a 16 bit
- RGB565 Swap per profondità di colore a 16 bit (vengono scambiati due byte)
- RGB888 per profondità colore a 24 bit (canali a 8 bit senza canale alfa)

Creare manualmente un'immagine

Se si genera un'immagine a run-time, è possibile creare una variabile immagine per visualizzarla utilizzando LVGL. Ad esempio:

```
uint8_t my_img_data[] = {0x00, 0x01, 0x02, ...};

static lv_image_dsc_t my_img_dsc = {
    .header.always_zero = 0,
    .header.w = 80,
    .header.h = 60,
    .data_size = 80 * 60 * LV_COLOR_DEPTH / 8,
    .header.cf = LV_COLOR_FORMAT_NATIVE,           /* Set the color format */
    .data = my_img_data,
};
```

Un'altra opzione (forse più semplice) per creare e visualizzare un'immagine in fase di esecuzione è utilizzare il Widget *Canvas*.

Utilizzare le immagini

Il modo più semplice per utilizzare un'immagine in LVGL è visualizzarla con un Widget *Image* (*lv_image*):

```
lv_obj_t * icon = lv_image_create(lv_screen_active(), NULL);
/* From variable */
lv_image_set_src(icon, &my_icon_dsc);
/* From file */
lv_image_set_src(icon, "S:my_icon.bin");
```

Se l'immagine è stata ricavata con il convertitore online, è necessario utilizzare `LV_IMAGE_DECLARE(my_icon_dsc)` per dichiarare l'immagine nel file in cui si desidera utilizzarla.

7.4.4 Decoder dell'Immagine

Come si può vedere nella sezione *Formati Colore*, LVGL supporta diversi formati di immagine. In molti casi, questi saranno tutto ciò di cui si ha bisogno. LVGL non supporta direttamente formati di immagine generici come PNG o JPG.

Per gestire formati di immagine non integrati, è necessario utilizzare librerie esterne e allegarle a LVGL tramite l'interfaccia *Image Decoder*.

Un image decoder è composto da 4 callback:

info

ottiene alcune informazioni di base sull'immagine (larghezza, altezza e formato del colore).

open

apertura di un'immagine: - memorizza un'immagine decodificata - la imposta su `NULL` per indicare che l'immagine può essere letta riga per riga.

get_area

se `open` non ha aperto completamente un'immagine, questa funzione dovrebbe restituire parte dell'immagine come dati decodificati.

close

chiude un'immagine aperta, libera le risorse allocate.

È possibile aggiungere un numero qualsiasi di decodificatori di immagini. Quando è necessario disegnare un'immagine, la libreria proverà tutti i decodificatori di immagini registrati finché non ne trova uno in grado di aprirla, ovvero uno che riconosce quel formato.

Il decoder integrato riconosce i seguenti formati:

- `LV_COLOR_FORMAT_I1`
- `LV_COLOR_FORMAT_I2`
- `LV_COLOR_FORMAT_I4`
- `LV_COLOR_FORMAT_I8`
- `LV_COLOR_FORMAT_RGB888`
- `LV_COLOR_FORMAT_XRGB8888`
- `LV_COLOR_FORMAT_ARGB8888`
- `LV_COLOR_FORMAT_RGB565`
- `LV_COLOR_FORMAT_RGB565A8`

Formati immagine personalizzati

Il modo più semplice per creare un'immagine personalizzata è utilizzare il convertitore di immagini online e selezionare il formato Raw o Raw with alpha. Prenderà ogni byte del file binario caricato e lo scriverà come un'immagine "bitmap". Si dovrà poi collegare un decoder di immagini che analizzerà la bitmap e genererà la bitmap reale, renderizzabile.

`header.cf` sarà `LV_COLOR_FORMAT_RAW`, `LV_COLOR_FORMAT_RAW_ALPHA` di conseguenza. Si deve scegliere il formato corretto in base alle proprie esigenze: un'immagine completamente opaca, che utilizza un canale alfa.

Il formato decodificato di un'immagine RAW dipende dal decodificatore. Esempio: le immagini JPG vengono decodificate in RGB888 e le immagini PNG in ARGB8888. Vedere [Formati Colore](#) per maggiori dettagli.

Registrazione di un decoder di immagini

Ecco un esempio di come far funzionare LVGL con le immagini PNG.

Per prima cosa, è necessario creare un nuovo decoder di immagini e impostare alcune funzioni per aprire/chiudere i file PNG. Dovrebbe apparire così:

```
/* Create a new decoder and register functions */
lv_image_decoder_t * dec = lv_image_decoder_create();
lv_image_decoder_set_info_cb(dec, decoder_info);
lv_image_decoder_set_open_cb(dec, decoder_open);
lv_image_decoder_set_get_area_cb(dec, decoder_get_area);
lv_image_decoder_set_close_cb(dec, decoder_close);

/**
 * Get info about a PNG image
 * @param decoder pointer to the decoder where this function belongs
 * @param src can be file name or pointer to a C array
 * @param header image information is set in header parameter
 * @return LV_RESULT_OK: no error; LV_RESULT_INVALID: can't get the info
 */
static lv_result_t decoder_info(lv_image_decoder_t * decoder, const void * src, lv_image_header_t * header)
{
    /* Check whether the type 'src' is known by the decoder */
    if(is_png(src) == false) return LV_RESULT_INVALID;

    /* Read the PNG header and find 'width' and 'height' */
    ...

    header->cf = LV_COLOR_FORMAT_ARGB8888;
    header->w = width;
    header->h = height;
}

/**
 * Open a PNG image and decode it into dscdecoded
 * @param decoder pointer to the decoder where this function belongs
 * @param dsc image descriptor
 * @return LV_RESULT_OK: no error; LV_RESULT_INVALID: can't open the image
 */
static lv_result_t decoder_open(lv_image_decoder_t * decoder, lv_image_decoder_dsc_t * dsc)
{
    (void) decoder; /* Unused */

    /* Check whether the type 'src' is known by the decoder */
    if(is_png(dsc->src) == false) return LV_RESULT_INVALID;

    /* Decode and store the image. If 'dsc->decoded' is 'NULL', the 'decoder_get_area' function will be called to get the image data line-by-line */
    dsc->decoded = my_png_decoder(dsc->src);

    /* Change the color format if decoded image format is different than original format. For PNG it's usually decoded to ARGB8888 format */
    dsc->decoded.header.cf = LV_COLOR_FORMAT_...

    /* Call a binary image decoder function if required. It's not required if 'my_png_decoder' opened the image in ARGB8888 format. */
    lv_result_t res = lv_bin_decoder_open(decoder, dsc);

    return res;
}

/**
 * Decode an area of image
 * @param decoder pointer to the decoder where this function belongs
 * @param dsc image decoder descriptor
 * @param full_area input parameter. the full area to decode after enough subsequent calls
 * @param decoded_area input/output parameter. set the values to 'LV_COORD_MIN' for the first call and to reset decoding.
 * @param return LV_RESULT_OK: ok; LV_RESULT_INVALID: failed or there is nothing left to decode
 */
static lv_result_t decoder_get_area(lv_image_decoder_t * decoder, lv_image_decoder_dsc_t * dsc,
                                    const lv_area_t * full_area, lv_area_t * decoded_area)
{
    /**
     * If 'dsc->decoded' is always set in 'decoder_open' then 'decoder_get_area' does not need to be implemented.
     * If 'dsc->decoded' is only sometimes set or never set in 'decoder_open' then 'decoder_get_area' is used to
     * incrementally decode the image by calling it repeatedly until it returns 'LV_RESULT_INVALID'.
     * In the example below the image is decoded line-by-line but the decoded area can have any shape and size
     * depending on the requirements and capabilities of the image decoder.
     */

    my_decoder_data_t * my_decoder_data = dsc->user_data;

    /* if 'decoded_area' has a field set to 'LV_COORD_MIN' then reset decoding */
    if(decoded_area->y1 == LV_COORD_MIN) {
        decoded_area->x1 = full_area->x1;
        decoded_area->x2 = full_area->x2;
        decoded_area->y1 = full_area->y1;
        decoded_area->y2 = decoded_area->y1; /* decode line-by-line, starting with the first line */

        /* create a draw buf the size of one line */
        bool reshape_success = NULL != lv_draw_buf_reshape(my_decoder_data->partial,
                                                          dsc->decoded.header.cf,
                                                          lv_area_get_width(full_area),
                                                          1,
                                                          LV_STRIDE_AUTO);

        if(!reshape_success) {
            lv_draw_buf_destroy(my_decoder_data->partial);
            my_decoder_data->partial = lv_draw_buf_create(lv_area_get_width(full_area),
                                                          1,
                                                          LV_STRIDE_AUTO);
        }
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

        dsc->decoded.header.cf,
        LV_STRIDE_AUTO);

    my_png_decode_line_reset(full_area);
}

/* otherwise decoding is already in progress, decode the next line */
else {
    /* all lines have already been decoded. indicate completion by returning `LV_RESULT_INVALID` */
    if (decoded_area->y1 >= full_area->y2) return LV_RESULT_INVALID;
    decoded_area->y1++;
    decoded_area->y2++;
}

my_png_decode_line(my_decoder_data->partial);

return LV_RESULT_OK;
}

/**
 * Close PNG image and free data
 * @param decoder pointer to the decoder where this function belongs
 * @param dsc     image decoder descriptor
 * @return       LV_RESULT_OK: no error; LV_RESULT_INVALID: can't open the image
 */
static void decoder_close(lv_image_decoder_t * decoder, lv_image_decoder_dsc_t * dsc)
{
    /* Free all allocated data */
    my_png_cleanup();

    my_decoder_data_t * my_decoder_data = dsc->user_data;
    lv_draw_buf_destroy(my_decoder_data->partial);

    /* Call the built-in close function if the built-in open/get_area was used */
    lv_bin_decoder_close(decoder, dsc);
}

```

In sintesi:

- In `decoder_info`, si devono raccogliere alcune informazioni di base sull'immagine e memorizzarle in header.
- In `decoder_open`, si deve provare ad aprire la sorgente dell'immagine puntata da `dsc->src`. Il suo tipo è già in `dsc->src_type == LV_IMG_SRC_FILE/VARIABLE`. Se questo formato/tipo non è supportato dal decoder, restituire `LV_RESULT_INVALID`. Tuttavia, se è possibile aprire l'immagine, è necessario impostare un puntatore all'immagine decodificata in `dsc->decoded`. Se il formato è noto, ma non si desidera decodificare l'intera immagine (ad esempio, non si dispone di memoria), si imposta `dsc->decoded = NULL` e si usa `decoder_get_area` per ottenere i pixel dell'area dell'immagine.
- In `decoder_close` è necessario liberare tutte le risorse allocate.
- `decoder_get_area` è facoltativo. In questo caso, è necessario decodificare l'intera immagine nella funzione `decoder_open` e memorizzare i dati dell'immagine in `dsc->decoded`. La decodifica dell'intera immagine richiede memoria aggiuntiva e un certo overhead computazionale.

Utilizzo manuale di un decoder di immagini

LVGL utilizzerà automaticamente i decodificatori di immagini registrati se si tenta di disegnare un'immagine raw (ad esempio utilizzando il Widget `lv_image`), ma è possibile utilizzarli anche manualmente. Creare una variabile `lv_image_decoder_dsc_t` per descrivere la sessione di decodifica e richiamare `lv_image_decoder_open()`.

Il parametro `color` viene utilizzato solo con le immagini `LV_COLOR_FORMAT_A1/2/4/8` per indicare il colore dell'immagine.

```

lv_result_t res;
lv_image_decoder_dsc_t dsc;
lv_image_decoder_args_t args = { 0 }; /* Custom decoder behavior via args */
res = lv_image_decoder_open(&dsc, &my_img_dsc, &args);

if(res == LV_RESULT_OK) {
    /* Do something with `dsc->decoded`. You can copy out the decoded image by `lv_draw_buf_dup(dsc.decoded)` */
    lv_image_decoder_close(&dsc);
}

```

Post-processing delle immagini

Considerando che alcuni hardware hanno requisiti speciali per i formati immagine, come la premoltiplicazione alfa e la "stride alignment", la maggior parte dei decoder di immagini (come i decoder PNG) potrebbe non produrre direttamente dati immagine che soddisfano i requisiti hardware.

Per questo motivo, LVGL fornisce una soluzione per la post-elaborazione delle immagini. Innanzitutto, si richiama una funzione di post-elaborazione personalizzata dopo `lv_image_decoder_open` per regolare i dati nella ca-

che delle immagini, poi si contrassegna lo stato di elaborazione in `cache_entry->process_state` (per evitare post-elaborazione ripetute).

Vedere il codice dettagliato di seguito:

- Esempio di post-elaborazione con "Stride alignment" e premoltiplicazione:

```
/* Define post-processing state */
typedef enum {
    IMAGE_PROCESS_STATE_NONE = 0,
    IMAGE_PROCESS_STATE_STRIDE_ALIGNED = 1 << 0,
    IMAGE_PROCESS_STATE_PREMULTIPLIED_ALPHA = 1 << 1,
} image_process_state_t;

lv_result_t my_image_post_process(lv_image_decoder_dsc_t * dsc)
{
    lv_color_format_t color_format = dsc->header.cf;
    lv_result_t res = LV_RESULT_OK;

    if(color_format == LV_COLOR_FORMAT_ARGB8888) {
        lv_cache_lock();
        lv_cache_entry_t * entry = dsc->cache_entry;

        if(!(entry->process_state & IMAGE_PROCESS_STATE_PREMULTIPLIED_ALPHA)) {
            lv_draw_buf_premultiply(dsc->decoded);
            LV_LOG_USER("premultiplied alpha OK");

            entry->process_state |= IMAGE_PROCESS_STATE_PREMULTIPLIED_ALPHA;
        }

        if(!(entry->process_state & IMAGE_PROCESS_STATE_STRIDE_ALIGNED)) {
            uint32_t stride_expect = lv_draw_buf_width_to_stride(decoded->header.w, decoded->header.cf);
            if(decoded->header.stride != stride_expect) {
                LV_LOG_WARN("Stride mismatch");
                lv_draw_buf_t * aligned = lv_draw_buf_adjust_stride(decoded, stride_expect);
                if(aligned == NULL) {
                    LV_LOG_ERROR("No memory for Stride adjust.");
                    return NULL;
                }

                decoded = aligned;
            }

            entry->process_state |= IMAGE_PROCESS_STATE_STRIDE_ALIGNED;
        }
    }

    alloc_failed:
    lv_cache_unlock();
}

return res;
}
```

- Esempio di unità di disegno GPU:

```
void gpu_draw_image(lv_draw_unit_t * draw_unit, const lv_draw_image_dsc_t * draw_dsc, const lv_area_t * coords)
{
    ...
    lv_image_decoder_dsc_t decoder_dsc;
    lv_result_t res = lv_image_decoder_open(&decoder_dsc, draw_dsc->src, NULL);
    if(res != LV_RESULT_OK) {
        LV_LOG_ERROR("Failed to open image");
        return;
    }

    res = my_image_post_process(&decoder_dsc);
    if(res != LV_RESULT_OK) {
        LV_LOG_ERROR("Failed to post-process image");
        return;
    }
    ...
}
```

7.4.5 Memorizzazione nella cache delle immagini

A volte l'apertura di un'immagine richiede molto tempo. Decodificare continuamente un'immagine PNG/JPEG o caricare immagini da una memoria esterna lenta sarebbe inefficiente e dannoso per l'esperienza utente.

Pertanto, LVGL memorizza nella cache i dati delle immagini. La memorizzazione nella cache implica che alcune immagini rimarranno aperte, quindi LVGL può accedervi rapidamente da `dsc->decoded` invece di doverle decodificare nuovamente.

Naturalmente, la memorizzazione nella cache delle immagini richiede molte risorse, poiché utilizza più RAM per memorizzare l'immagine decodificata. LVGL cerca di ottimizzare il processo il più possibile (vedere di seguito), ma sarà comunque necessario valutare se questo sia vantaggioso per la propria piattaforma o meno. La memorizzazione nella cache delle immagini potrebbe non essere utile se si dispone di un target profondamente integrato che decodifica immagini di piccole dimensioni da un supporto di memorizzazione relativamente veloce.

Dimensione della cache

La dimensione della cache (in byte) può essere definita con `LV_CACHE_DEF_SIZE` in `lv_conf.h`. Il valore di default è 0, quindi nessuna immagine viene memorizzata nella cache.

La dimensione della cache può essere modificata in fase di esecuzione con `lv_cache_set_max_size(size_t size)`, e ottenuta con `lv_cache_get_max_size()`.

Valore delle immagini

Quando si utilizzano più immagini rispetto alla dimensione disponibile della cache, LVGL non può memorizzarle tutte nella cache. Al contrario, la libreria chiuderà una delle immagini memorizzate nella cache per liberare spazio.

Per decidere quale immagine chiudere, LVGL utilizza una misurazione effettuata in precedenza sul tempo impiegato per aprirla. Le voci della cache che contengono immagini più lente da aprire sono considerate più preziose e vengono mantenute nella cache il più a lungo possibile.

Per sovrascrivere la misurazione di LVGL, è possibile impostare manualmente il valore *weight* nella voce della cache in `cache_entry->weight = time_ms` per ottenere un valore maggiore o minore. (Lasciarlo invariato per consentire a LVGL di gestirlo).

Ogni voce nella cache ha un valore *"life"*. Ogni volta che un'immagine viene aperta tramite la cache, il valore *life* di tutte le voci viene aumentato dei rispettivi valori *weight* per renderle più vecchie. Quando viene utilizzata un'immagine nella cache, il suo valore *usage_count* viene aumentato per renderla più *"live"*.

Se non c'è più spazio nella cache, la voce con *usage_count == 0* e il valore di *life* più basso verrà eliminata.

Utilizzo della memoria

Si noti che un'immagine memorizzata nella cache potrebbe consumare memoria in modo continuo. Ad esempio, se tre immagini PNG sono memorizzate nella cache, consumeranno memoria mentre sono aperte.

Pertanto, è responsabilità dell'utente assicurarsi che ci sia abbastanza RAM per memorizzare nella cache anche le immagini più grandi contemporaneamente.

Pulire la cache

Supponiamo di aver caricato un'immagine PNG in una variabile `lv_image_dsc_t my_png` e di averla utilizzata in un Widget `lv_image`. Se l'immagine è già memorizzata nella cache e si modifica il file PNG sottostante, è necessario notificare a LVGL di memorizzare nuovamente l'immagine nella cache. In caso contrario, non esiste un modo semplice per rilevare che il file sottostante è cambiato e LVGL continuerà a estrarre la vecchia immagine dalla cache.

Per farlo, si usa `lv_cache_invalidate(lv_cache_find(&my_png, LV_CACHE_SRC_TYPE_PTR, 0, 0))`.

Algoritmo di cache personalizzato

Per implementare un proprio algoritmo di cache, si può fare riferimento al seguente codice per sostituire il gestore di cache integrato in LVGL:

```
static lv_cache_entry_t * my_cache_add_cb(size_t size)
{
    ...
}

static lv_cache_entry_t * my_cache_find_cb(const void * src, lv_cache_src_type_t src_type, uint32_t param1, uint32_t param2)
{
    ...
}

static void my_cache_invalidate_cb(lv_cache_entry_t * entry)
{
    ...
}

static const void * my_cache_get_data_cb(lv_cache_entry_t * entry)
{
    ...
}

static void my_cache_release_cb(lv_cache_entry_t * entry)
{
    ...
}

static void my_cache_set_max_size_cb(size_t new_size)
{
```

(continues on next page)

(continua dalla pagina precedente)

```

...
}

static void my_cache_empty_cb(void)
{
...
}

void my_cache_init(void)
{
/* Initialize new cache manager. */
lv_cache_manager_t my_manager;
my_manager.add_cb = my_cache_add_cb;
my_manager.find_cb = my_cache_find_cb;
my_manager.invalidate_cb = my_cache_invalidate_cb;
my_manager.get_data_cb = my_cache_get_data_cb;
my_manager.release_cb = my_cache_release_cb;
my_manager.set_max_size_cb = my_cache_set_max_size_cb;
my_manager.empty_cb = my_cache_empty_cb;

/* Replace existing cache manager with the new one. */
lv_cache_lock();
lv_cache_set_manager(&my_manager);
lv_cache_unlock();
}

```

7.4.6 API

lv_draw_buf.h
lv_draw_image.h
lv_draw_image_private.h
lv_image_decoder.h
lv_image_decoder_private.h
lv_image_dsc.h
lv_api_map_v8.h
lv_api_map_v9_0.h
lv_api_map_v9_1.h
lv_image_cache.h
lv_image_header_cache.h
lv_style.h
lv_image.h
lv_image_private.h
lv_obj_property_names.h

7.5 Colore (lv_color)

Il modulo colore gestisce tutte le funzioni relative al colore, come la modifica della profondità di colore, la creazione di colori da codice esadecimale, la conversione tra profondità di colore, la miscelazione dei colori, ecc.

Il tipo *lv_color_t* viene utilizzato per memorizzare un colore in formato RGB888. Questo tipo e formato sono utilizzati in quasi tutte le API, indipendentemente da LV_COLOR_DEPTH.

7.5.1 Creazione di Colori

RGB

Crea colori dai valori dei canali Rosso, Verde e Blu:

```

/* All channels are 0-255 */
lv_color_t c = lv_color_make(red, green, blue);

/* Same but can be used for const initialization as well */
lv_color_t c = LV_COLOR_MAKE(red, green, blue);

```

(continues on next page)

(continua dalla pagina precedente)

```
/* From hex code 0x000000..0xFFFFFFF interpreted as RED + GREEN + BLUE */
lv_color_t c = lv_color_hex(0x123456);

/* From 3 digits. Same as lv_color_hex(0x112233) */
lv_color_t c = lv_color_hex3(0x123);
```

HSV

Crea colori dai valori di Tonalità, Saturazione e Valore:

```
//h = 0..359, s = 0..100, v = 0..100
lv_color_t c = lv_color_hsv_to_rgb(h, s, v);

//All channels are 0-255
lv_color_hsv_t c_hsv = lv_color_rgb_to_hsv(r, g, b);

//From lv_color_t variable
lv_color_hsv_t c_hsv = lv_color_to_hsv(color);
```

Palette

LVGL include la [paletta di Material Design](#) di colori. In questo sistema, tutti i colori denominati hanno un colore principale nominale e quattro varianti più scure e cinque più chiare.

I nomi dei colori sono i seguenti:

- LV_PALETTE_RED
- LV_PALETTE_PINK
- LV_PALETTE_PURPLE
- LV_PALETTE_DEEP_PURPLE
- LV_PALETTE_INDIGO
- LV_PALETTE_BLUE
- LV_PALETTE_LIGHT_BLUE
- LV_PALETTE_CYAN
- LV_PALETTE_TEAL
- LV_PALETTE_GREEN
- LV_PALETTE_LIGHT_GREEN
- LV_PALETTE_LIME
- LV_PALETTE_YELLOW
- LV_PALETTE_AMBER
- LV_PALETTE_ORANGE
- LV_PALETTE_DEEP_ORANGE
- LV_PALETTE_BROWN
- LV_PALETTE_BLUE_GREY
- LV_PALETTE_GREY

Per ottenere il colore principale, si usa `lv_color_t c = lv_palette_main(LV_PALETTE_...)`.

Per le varianti più chiare di un colore della tavolozza, si usa `lv_color_t c = lv_palette_lighten(LV_PALETTE_..., v)`. `v` può essere compreso tra 1 e 5. Per le varianti più scure di un colore della tavolozza, si usa `lv_color_t c = lv_palette_darken(LV_PALETTE_..., v)`. `v` può essere compreso tra 1 e 4.

Modificare e mescolare i colori

Le seguenti funzioni possono modificare un colore:

```
// Lighten a color. 0: no change, 255: white
lv_color_t c = lv_color_lighten(c, lvl);

// Darken a color. 0: no change, 255: black
lv_color_t c = lv_color_darken(lv_color_t c, lv_opa_t lvl);

// Lighten or darken a color. 0: black, 128: no change 255: white
lv_color_t c = lv_color_change_lightness(lv_color_t c, lv_opa_t lvl);

// Mix two colors with a given ratio 0: full c2, 255: full c1, 128: half c1 and half c2
lv_color_t c = lv_color_mix(c1, c2, ratio);
```

Colori predefiniti

`lv_color_white()` e `lv_color_black()` restituiscono rispettivamente 0xFFFFFFF e 0x000000.

7.5.2 Opacità

Per descrivere l'opacità, il tipo `lv_opa_t` viene creato da `uint8_t`. Sono state introdotte anche alcune definizioni speciali:

- `LV_OPA_TRANSP` Valore: 0, significa nessuna opacità, rendendo il colore completamente trasparente
- `LV_OPA_10` Valore: 25, significa che il colore copre solo parzialmente
- `LV_OPA_20` ... `OPA_80` seguono logicamente
- `LV_OPA_90` Valore: 229, significa che il colore copre quasi completamente
- `LV_OPA_COVER` Valore: 255, significa che il colore copre completamente (opacità completa)

È anche possibile utilizzare le definizioni `LV_OPA_*` in `lv_color_mix()` come *rapporto* di miscelazione.

7.5.3 API

`lv_draw_ppa_private.h`

`lv_color.h`

`lv_color_op.h`

7.6 Timer (`lv_timer`)

LVGL ha un sistema di Timer integrato. È possibile registrare una funzione per farla chiamare periodicamente. I Timer vengono gestiti e chiamati in `lv_timer_handler()`, che deve essere chiamato ogni pochi millisecondi. Vedere [Gestione del Timer](#) per maggiori informazioni.

Per default, LVGL utilizza i Timer per:

- aggiornare ogni display: durante la creazione di ogni *Display* (`lv_display`), viene creato un Timer per quel display. Il Timer aggiorna la visualizzazione in base al valore configurato di `LV_DEF_REFR_PERIOD` e invia anche tutti gli eventi relativi alla visualizzazione, come `LV_EVENT_REFR_START`, `LV_EVENT_REFR_READY`, ecc.
- Dispositivi di input di lettura: durante la creazione di ogni *Dispositivi di input* (`lv_indev`), viene creato un Timer per quel Dispositivo di Input in base al valore configurato di `LV_DEF_REFR_PERIOD`. Quel Timer determina la lettura del dispositivo di input e invia anche tutti gli eventi relativi al dispositivo di input, come `LV_EVENT_CLICKED`, `LV_EVENT_PRESSED`, ecc.
- Aggiorna i valori del monitor di sistema: se `LV_USE_SYSMON` è impostato a 1 in `lv_conf.h`, vengono creati uno o più timer per calcolare e monitorare periodicamente le statistiche sulle prestazioni del sistema e l'utilizzo della memoria di LVGL.

I Timer non sono "preemptive", il che significa che un Timer non può interrompere un altro Timer. Pertanto, è possibile chiamare qualsiasi funzione relativa a LVGL in un Timer.

7.6.1 Creazione di un Timer

Per creare un nuovo Timer, si usa `lv_timer_create(timer_cb, period_ms, user_data)`. Restituisce un `lv_timer_t *` che può essere utilizzato in seguito per modificare i parametri del Timer, metterlo in pausa o eliminarlo quando non è più necessario. `lv_timer_create_basic()` può anche essere utilizzato per creare un nuovo Timer senza specificare alcun parametro.

Una callback del Timer dovrebbe avere questo prototipo: `void (*lv_timer_cb_t)(lv_timer_t *)`.

Ad esempio:

```
void my_timer(lv_timer_t * timer)
{
    /* Use the user_data */
    uint32_t * user_data = lv_timer_get_user_data(timer);
    printf("my_timer called with user data: %d\n", *user_data);

    /* Do something with LVGL */
    if(something_happened) {
        something_happened = false;
        lv_button_create(lv_screen_active());
    }
}

...
static uint32_t user_data = 10;
lv_timer_t * timer = lv_timer_create(my_timer, 500, &user_data);
```

7.6.2 Ready e Reset

`lv_timer_ready(timer)` fa sì che un Timer venga eseguito alla successiva chiamata di `lv_timer_handler()`.

`lv_timer_reset(timer)` azzera il periodo di un Timer. Verrà richiamato nuovamente dopo che il periodo attualmente impostato (in millisecondi) sarà trascorso.

Guardare cosa succede quando *nessun timer è pronto* e viene chiamato `lv_timer_handler()`.

7.6.3 Impostazione dei Parametri

È possibile modificare i parametri del Timer in qualsiasi momento durante il suo ciclo di vita:

- `lv_timer_set_cb(timer, new_cb)`
- `lv_timer_set_period(timer, new_period_ms)`
- `lv_timer_set_user_data(timer, user_data)`

7.6.4 Conteggio Ripetizioni

Quando viene creato un Timer, il suo conteggio ripetizioni viene impostato su -1 per farlo ripetere all'infinito. È possibile impostare un timer in modo che si ripeta solo un determinato numero di volte con `lv_timer_set_repeat_count(timer, count)`. Per default, una volta che il timer è stato eseguito `count` volte, verrà automaticamente eliminato.

È possibile utilizzare `lv_timer_set_auto_delete(timer, false)` se si desidera che il timer venga messo in pausa dopo essere stato eseguito `count` volte. Questo può essere utile se si riutilizza ripetutamente quel timer e si desidera evitare il sovraccarico di CPU e `lv_malloc()` dovuto alla creazione e all'eliminazione ripetute di un timer. Se si utilizza questa opzione, sarà necessario impostarne il conteggio delle ripetizioni (su -1 o su un conteggio positivo, poiché sarà decrementato a 0) e `resume` per riattivarlo.

7.6.5 Pausa e Ripristino

`lv_timer_pause(timer)` mette in pausa il Timer specificato.

`lv_timer_resume(timer)` riprende il Timer specificato.

7.6.6 Misurazione del Tempo di Inattività

È possibile ottenere la percentuale di tempo di inattività di `lv_timer_handler()` con `lv_timer_get_idle()`. Si noti che non misura il tempo di inattività dell'intero sistema, ma solo di `lv_timer_handler()`. Questo può essere fuorviante se si utilizza un sistema operativo e DMA e/o GPU vengono utilizzati durante il rendering, poiché non misura effettivamente il tempo trascorso dal sistema operativo in un thread inattivo.

Se si utilizza un sistema operativo e si desidera ottenere il tempo trascorso dalla CPU in un thread inattivo, un modo per farlo è configurare `LV_USE_SYSMON` e `LV_USE_PERF_MONITOR` a 1 in `lv_conf.h` (se non lo sono già) e impostare la macro `LV_SYSMON_GET_IDLE` con il nome di una funzione che recupera la percentuale di tempo della CPU trascorso nel thread inattivo del sistema operativo. Un esempio di tale funzione è `lv_os_get_idle_percent()` in `lv_freertos.c`. Mentre la configurazione è impostata in questo modo, alcune statistiche sulle prestazioni del sistema (incluso il carico della CPU) appariranno sul display in un'etichetta parzialmente trasparente la cui posizione è impostata dalla macro `LV_USE_PERF_MONITOR_POS`.

7.6.7 Abilitazione e Disabilitazione

È possibile disabilitare temporaneamente la gestione dei Timer con `lv_timer_enable(false)`. Attenzione: questo sospende anche la gestione dei Timer che aggiornano i Display e leggono dai dispositivi di input, quindi non dimenticare di riabilitarla con `lv_timer_enable(true)` appena la necessità della pausa è terminata.

7.6.8 Callback di Ripristino del Gestore dei Timer

Quando il sistema dei timer è stato disabilitato (causando il ritorno anticipato di `lv_timer_handler()` prima di aver elaborato i timer), per intraprendere un'azione quando il sistema dei timer viene riabilitato, impostare una callback di ripristino utilizzando `lv_timer_handler_set_resume_cb(cb, user_data)`. La callback dovrebbe avere questo prototipo: `void (*lv_timer_handler_resume_cb_t)(void*)`.

7.6.9 Chiamate asincrone

Ci sono diversi casi in cui si potrebbe non voler eseguire un'azione immediatamente. Alcuni esempi sono:

- non si può eliminare un Widget perché qualcos'altro lo sta ancora utilizzando,
- non si vuole bloccare l'esecuzione ora, oppure
- si rileva la necessità di eliminare un Widget in un thread diverso da quello che effettua le chiamate LVGL (ad esempio, nel caso in cui si utilizzi un *Gateway Thread* per effettuare tutte le chiamate LVGL in un ambiente multi-thread).

In questi casi, `lv_async_call(my_function, data_p)` può essere utilizzata per chiamare `my_function` alla successiva invocazione di `lv_timer_handler()`. Come effetto collaterale, questo garantisce anche che venga chiamato in un thread in cui è sicuro effettuare chiamate LVGL. `data_p` verrà passato alla funzione quando viene chiamata. Si noti che viene salvato solo il puntatore ai dati, quindi qualsiasi cosa punti deve rimanere valida fino alla chiamata della funzione, in modo che possa puntare a dati statici, globali o allocati dinamicamente. Per annullare una chiamata asincrona, si chiama `lv_async_call_cancel(my_function, data_p)`, che rimuoverà tutte le chiamate asincrone corrispondenti a `my_function` e a `data_p`.

Si noti che, se `lv_async_call(my_function, data_p)` viene chiamato da un thread diverso da quello che normalmente effettua chiamate LVGL, si è comunque obbligati a proteggere la struttura dati LVGL utilizzando un MUTEX.

Ad esempio:

```
void my_screen_clean_up(void * scr)
{
    /* Free some resources related to `scr` */

    /* Finally delete the screen */
    lv_obj_delete(scr);
}

...
/* Do something with the Widget on the current screen */

/* Delete screen on next call of 'lv_timer_handler', not right now. */
lv_lock();
lv_async_call(my_screen_clean_up, lv_screen_active());
```

(continues on next page)

(continua dalla pagina precedente)

```
lv_unlock();
/* The screen is still valid so you can do other things with it */
```

Per eliminare un widget e non c'è bisogno di pulire nulla in `my_screen_cleanup`, si può semplicemente usare `lv_obj_delete_async(widget)`, che eliminerà il Widget alla successiva chiamata a `lv_timer_handler()`.

7.6.10 API

`lv_api_map_v8.h`

`lv_timer.h`

`lv_timer_private.h`

7.7 Animazione (lv_anim)

Le animazioni consentono di definire il modo in cui qualcosa dovrebbe muoversi o cambiare nel tempo, lasciando che LVGL si occupi del grosso del lavoro. Ciò che lo rende così potente è che l'oggetto da modificare può essere praticamente qualsiasi cosa nel sistema. È molto comodo applicarlo ai Widget LVGL nell'interfaccia utente (UI), per modificarne l'aspetto, le dimensioni o la posizione nel tempo. Ma poiché è --- in sostanza --- un gestore generico di modifiche nel tempo, completo di una varietà di callback di eventi opzionali, la sua applicazione può essere più ampia rispetto ai soli componenti dell'UI.

Per ogni Animazione creata, LVGL realizza quanto sopra fornendo un metodo generico per variare un intero con segno da un valore iniziale a un valore finale in un periodo di tempo specificato. Consente di specificare a quale oggetto si applica (la "variabile"), che è disponibile nelle funzioni di callback che vengono chiamate durante la riproduzione dell'animazione.

Questa variazione nel tempo può essere lineare (impostazione predefinita), può essere lungo un percorso (curva) specificato dall'utente ed è persino possibile specificare una varietà di effetti non lineari di uso comune.

La callback principale chiamata durante un'Animazione (durante la riproduzione) è chiamata funzione *animator*, che ha il seguente prototipo:

```
void func(void *var , int32_t value);
```

Questo prototipo semplifica l'utilizzo della maggior parte delle funzioni *set* LVGL direttamente o tramite un wrapper banale. Include:

- la maggior parte delle proprietà dei widget
- funzioni che impostano le *proprietà di stile locali* direttamente sugli oggetti (necessita di un wrapper per impostare il *selector*)
- impostano le proprietà sugli oggetti `lv_style_t` (ad esempio, *stili condivisi*) (`lv_obj_report_style_change` deve essere chiamato per notificare i widget che hanno lo stile)
- `lv_style_set_<property_name>(&style, <value>)`
- `lv_obj_set_<property_name>(widget, <value>)`

Per via del primo, un'animazione su un singolo oggetto `lv_style_t` condiviso tra più oggetti può modificare simultaneamente l'aspetto di tutti gli oggetti che lo utilizzano. Consultare *Stili* per maggiori dettagli.

Esempi del secondo sono: `lv_obj_set_x(widget, value)` o `lv_obj_set_width(widget, value)`.

Questo lo rende molto comodo da applicare all'aspetto (e ad altri attributi) dei componenti dell'interfaccia utente. Ma è possibile fornire le proprie funzioni "set", quindi l'applicazione delle Animazioni è limitata solo dall'immaginazione.

Il numero di Animazioni riproducibili contemporaneamente per un dato oggetto con una data callback *animator* è uno (1). Tuttavia, il numero di animazioni che possono essere riprodotte contemporaneamente è limitato solo dalla RAM e dal tempo di CPU disponibili per:

- un dato oggetto con diverse callback *animator* ; e

- oggetti diversi.

Quindi, è possibile che la larghezza di un Pulsante venga modificata da un'Animazione e la sua altezza da un'altra Animazione.

7.7.1 Creare un'Animazione

Per creare un'Animazione, si inizia creando un *template* di Animazione in una variabile `lv_anim_t`. Deve essere inizializzato e configurato con le funzioni `lv_anim_set_...()`.

```
/* INITIALIZE AN ANIMATION
*-----*/
static lv_anim_t anim_template;
static lv_anim_t * running_anim;

lv_anim_init(&anim_template);

/* MANDATORY SETTINGS
*-----*/
/* Set the "animator" function */
lv_anim_set_exec_cb(&anim_template, (lv_anim_exec_xcb_t) lv_obj_set_x);

/* Set target of the Animation */
lv_anim_set_var(&anim_template, widget);

/* Length of the Animation [ms] */
lv_anim_set_duration(&anim_template, duration_in_ms);

/* Set start and end values. E.g. 0, 150 */
lv_anim_set_values(&anim_template, start, end);

/* OPTIONAL SETTINGS
*-----*/
/* Time to wait before starting the Animation [ms] */
lv_anim_set_delay(&anim_template, delay);

/* Set path (curve). Default is linear */
lv_anim_set_path_cb(&anim_template, lv_anim_path_ease_in);

/* Set anim_template callback to indicate when the Animation is completed. */
lv_anim_set_completed_cb(&anim_template, completed_cb);

/* Set anim_template callback to indicate when the Animation is deleted (idle). */
lv_anim_set_deleted_cb(&anim_template, deleted_cb);

/* Set anim_template callback to indicate when the Animation is started (after delay). */
lv_anim_set_start_cb(&anim_template, start_cb);

/* When ready, play the Animation backward with this duration. Default is 0 (disabled) [ms] */
lv_anim_set_reverse_duration(&anim_template, time);

/* Delay before reverse play. Default is 0 (disabled) [ms] */
lv_anim_set_reverse_delay(&anim_template, delay);

/* Number of repetitions. Default is 1. LV_ANIM_REPEAT_INFINITE for infinite repetition */
lv_anim_set_repeat_count(&anim_template, cnt);

/* Delay before repeat. Default is 0 (disabled) [ms] */
lv_anim_set_repeat_delay(&anim_template, delay);

/* true (default): apply the start value immediately, false: apply start value after delay when the Anim. really starts. */
lv_anim_set_early_apply(&anim_template, true/false);

/* START THE ANIMATION
*-----*/
running_anim = lv_anim_start(&anim_template); /* Start the Animation */
```

7.7.2 Path dell'Animazione

È possibile controllare il Percorso (curva) di un'Animazione. Il caso più semplice è lineare, ovvero il valore corrente tra *inizio* e *fine* viene modificato alla stessa velocità (ovvero con incrementi fissi) per tutta la durata dell'animazione. Un *Path* è una funzione che calcola il valore successivo da impostare in base allo stato corrente dell'Animazione. Sono disponibili diversi *Path* predefiniti:

- `lv_anim_path_linear()`: Animazione lineare (predefinita)
- `lv_anim_path_step()`: modifica di un passo alla fine
- `lv_anim_path_ease_in()`: lenta all'inizio
- `lv_anim_path_ease_out()`: lenta alla fine
- `lv_anim_path_ease_in_out()`: lenta all'inizio e alla fine
- `lv_anim_path_overshoot()`: supera il valore finale
- `lv_anim_path_bounce()`: rimbalza leggermente dal valore finale (come se colpisce un muro)

In alternativa, si può fornire una propria funzione Path.

`lv_anim_init(&my_anim)` imposta il Path su `lv_anim_path_linear()` per default. Per utilizzare un Path diverso (inclusa una funzione Path personalizzata fornita), è possibile impostarlo utilizzando `lv_anim_set_path_cb(&anim_template, path_cb)`.

Se si fornisce una funzione Path personalizzata, il suo prototipo è:

```
int32_t calculate_value(lv_anim_t * anim);
```

7.7.3 Velocità vs Tempo

Normalmente, si imposta la durata dell'animazione direttamente tramite `lv_anim_set_duration(&anim_template, duration_in_ms)`. In alcuni casi, tuttavia, la *velocità* è nota, ma la durata non lo è. Dati i valori `start` e `end` di un'Animazione, `rate` qui indica il numero di unità di variazione al secondo, ovvero la velocità (unità al secondo) con cui il valore dell'animazione deve cambiare tra il valore `start` e `end`. In questi casi esiste una funzione di utilità `lv_anim_speed_to_time()` utilizzabile per calcolare la durata dell'animazione, quindi, la si può impostare in questo modo:

```
uint32_t change_per_sec = 20;
uint32_t duration_in_ms = lv_anim_speed_to_time(change_per_sec, 0, 100);
/* `duration_in_ms` will be 5000 */
lv_anim_set_duration(&anim_template, duration_in_ms);
```

7.7.4 Animazione in Entrambe le Direzioni

A volte un'animazione deve essere riprodotta in avanti e poi all'indietro, invertendo di fatto il percorso, animando dal valore `end` al valore `start` di nuovo. Per fare ciò, si passa un valore diverso da zero a questa funzione per impostare la durata della parte inversa dell'Animazione: `lv_anim_set_reverse_duration(&anim_template, duration_in_ms)`.

Facoltativamente, si può anche introdurre un ritardo tra la direzione avanti e quella indietro usando `lv_anim_set_reverse_delay(&anim_template, delay_in_ms)`

7.7.5 Avvio di un'Animazione

Dopo aver impostato l'oggetto `lv_anim_t`, è importante rendersi conto che ciò che si è impostato è un "template" per un'animazione in esecuzione che non è ancora stata creata. Quando si chiama `lv_anim_start(&anim_template)` passando il `template` impostato, questa utilizza il template per allocare dinamicamente un oggetto interno che è un'Animazione *in esecuzione*. Questa funzione restituisce un puntatore a tale oggetto.

```
static lv_anim_t anim_template;
static lv_anim_t * running_anim;

/* Set up template... */
lv_anim_init(&anim_template);
/* ...and other set-up functions above. */

/* Later... */
running_anim = lv_anim_start(&anim_template);
```

Nota

`lv_anim_start(&anim_template)` crea una propria copia del template dell'Animazione, quindi, se non serve in seguito, non è necessario conservarne il contenuto dopo questa chiamata.

Una volta avviata un'animazione *in esecuzione*, questa prosegue fino al suo completamento o fino a quando non viene eliminata (vedere di seguito), a seconda di quale evento si verifica per primo. Un'Animazione è completata quando:

- il suo "valore" ha raggiunto il valore `end` designato;
- se l'Animazione ha un valore di durata *inversa* diverso da zero, il suo valore è stato eseguito dal valore `end` al valore `start`;
- se è stato impostato un conteggio di ripetizioni diverso da zero, l'Animazione è stata ripetuta per quel numero di volte.

Una volta completata, l'Animazione *live, in esecuzione* viene automaticamente eliminata dall'elenco delle Animazioni in esecuzione. Ciò non influisce sul template dell'Animazione

i Nota

Se `lv_anim_set_repeat_count(&anim_template, cnt)` è stata chiamata passando `LV_ANIM_REPEAT_INFINITE`, l'animazione non raggiunge mai lo stato di "completamento". In questo caso, è necessario eliminarla per terminare l'Animazione.

7.7.6 Eliminazione di Animazioni

È necessario eliminare un'Animazione utilizzando `lv_anim_delete(var, func)` se si verifica una di queste due condizioni:

- l'oggetto (variabile) animato viene eliminato (e non è un Widget) oppure
- un'animazione in esecuzione deve essere interrotta prima del completamento.

i Nota

Se l'oggetto (variabile) eliminato è un tipo di Widget, il codice di gestione coinvolto nella sua eliminazione elimina anche tutte le animazioni in esecuzione ad esso collegate. Quindi `lv_anim_delete(var, func)` deve essere chiamato solo se l'oggetto eliminato *non* è uno dei Widget.

Se è stata conservata una copia del puntatore restituito da `lv_anim_start()` come `running_anim`, si può eliminare l'animazione in esecuzione in questo modo:

```
lv_anim_delete(running_anim->var, running_anim->exec_cb);
```

Nel caso in cui l'animazione venga completata *dopo* aver determinato che deve essere eliminata e prima che venga effettuata la chiamata a `lv_anim_delete()`, non c'è alcun problema a richiamarla una seconda volta: non si verificherà alcun danno.

Questa funzione restituisce un valore Booleano che indica se sono state eliminate Animazioni *live, running*.

7.7.7 Messa in Pausa delle Animazioni

Se è stata conservata una copia del puntatore restituito da `lv_anim_start()`, si può mettere in pausa l'animazione in esecuzione utilizzando `lv_anim_pause(animation)` e poi riprenderla utilizzando `lv_anim_resume(animation)`.

`lv_anim_pause_for(animation, milliseconds)` è disponibile anche per riprendere automaticamente in seguito l'animazione.

7.7.8 Timeline

È possibile creare una serie di animazioni correlate collegate tra loro utilizzando una Timeline di Animazione. Una Timeline è una raccolta di più animazioni che semplifica la creazione di Animazioni composite complesse. Per creare e utilizzare una Timeline di Animazione:

- Creare un template di Animazione, ma non chiamare `lv_anim_start()` su di esso.
- Creare un oggetto Timeline di Animazione chiamando `lv_anim_timeline_create()`.
- Aggiungere template di Animazione alla Timeline chiamando `lv_anim_timeline_add(timeline, start_time, &anim_template)`. `start_time` è il tempo di inizio dell'Animazione sulla Timeline. Si noti che `start_time` sovrascriverà qualsiasi valore specificato in `lv_anim_set_delay(&anim_template, delay)`.
- Chiamare `lv_anim_timeline_start(timeline)` per avviare la Timeline dell'Animazione.

i Nota

`lv_anim_timeline_add(timeline, start_time, &anim_template)` crea una propria copia del contenuto del template dell'Animazione, quindi, se non ce n'è bisogno in seguito, il suo contenuto non deve essere conservato dopo questa chiamata.

Supporta la riproduzione in avanti e all'indietro dell'intero gruppo di Animazione, utilizzando `lv_anim_timeline_set_reverse(timeline, reverse)`. Si noti che per riprodurre al contrario dalla fine della Timeline, è necessario chiamare `lv_anim_timeline_set_progress(timeline, LV_ANIM_TIMELINE_PROGRESS_MAX)` dopo aver aggiunto tutte le animazioni e prima di avviare la riproduzione.

Si chiama `lv_anim_timeline_pause(timeline)` per mettere in pausa la Timeline dell'Animazione. Nota: questo non ne preserva lo stato. L'unico modo per riavviarla è chiamare `lv_anim_timeline_start(timeline)`, che avvia la Timeline dall'inizio o dal punto impostato da `lv_anim_timeline_set_progress(timeline, progress)`.

Chiamare la funzione `lv_anim_timeline_set_progress(timeline, progress)` per impostare lo stato della Timeline di Animazione in base al valore `progress`. `progress` è un valore compreso tra 0 e 32767 (`LV_ANIM_TIMELINE_PROGRESS_MAX`) per indicare la porzione di Timeline che è stata "riprodotta". Esempio: un valore `progress` di `LV_ANIM_TIMELINE_PROGRESS_MAX / 2` imposterebbe la riproduzione della Timeline a metà.

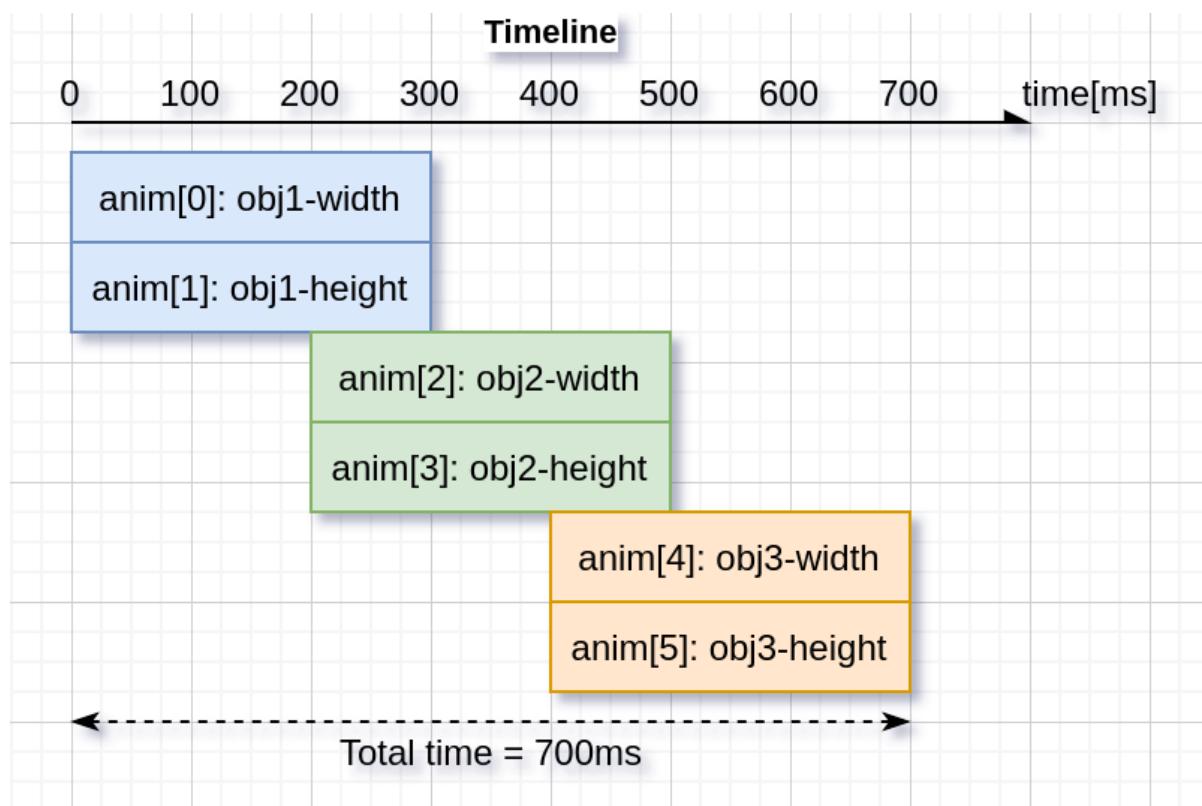
Chiamare la funzione `lv_anim_timeline_get_playtime(timeline)` per ottenere la durata totale (in millisecondi) dell'intera Timeline di Animazione.

Chiamare la funzione `lv_anim_timeline_get_reverse(timeline)` per verificare se la Timeline dell'Animazione viene riprodotta anche al contrario dopo il completamento della riproduzione in avanti.

Chiamare la funzione `lv_anim_timeline_delete(timeline)` per eliminare la Timeline dell'Animazione.

⚠ Attenzione

Per eliminare un Widget durante l'Animazione, assicurarsi di eliminare la Timeline dell'animazione prima di eliminare il widget. In caso contrario, il programma potrebbe bloccarsi o comportarsi in modo anomalo.



7.7.9 Esempi

Avviare l'animazione su un evento

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_x((lv_obj_t *) var, v);
}

static void sw_event_cb(lv_event_t * e)
{
    lv_obj_t * sw = lv_event_get_target_obj(e);
    lv_obj_t * label = (lv_obj_t *) lv_event_get_user_data(e);

    if(lv_obj_has_state(sw, LV_STATE_CHECKED)) {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), 100);
        lv_anim_set_duration(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_overshoot);
        lv_anim_start(&a);
    }
    else {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), -lv_obj_get_width(label));
        lv_anim_set_duration(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_ease_in);
        lv_anim_start(&a);
    }
}

/***
 * Start animation on an event
 */
void lv_example_anim_1(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Hello animations!");
    lv_obj_set_pos(label, 100, 10);

    lv_obj_t * sw = lv_switch_create(lv_screen_active());
    lv_obj_center(sw);
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_event_cb(sw, sw_event_cb, LV_EVENT_VALUE_CHANGED, label);
}
#endif
```

Riproduzione di animazione

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_x((lv_obj_t *) var, v);
}

static void anim_size_cb(void * var, int32_t v)
{
    lv_obj_set_size((lv_obj_t *) var, v, v);
}

/***
 * Create a playback animation
 */
void lv_example_anim_2(void)
{
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_remove_flag(obj, LV_OBJ_FLAG_SCROLLABLE);
    lv_obj_set_style_bg_color(obj, lv_palette_main(LV_PALETTE_RED), 0);
    lv_obj_set_style_radius(obj, LV_RADIUS_CIRCLE, 0);

    lv_obj_align(obj, LV_ALIGN_LEFT_MID, 10, 0);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, obj);
    lv_anim_set_values(&a, 10, 50);
    lv_anim_set_duration(&a, 1000);
    lv_anim_set_reverse_delay(&a, 100);
    lv_anim_set_reverse_duration(&a, 300);
    lv_anim_set_repeat_delay(&a, 500);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_set_path_cb(&a, lv_anim_path_ease_in_out);

    lv_anim_set_exec_cb(&a, anim_size_cb);
    lv_anim_start(&a);
    lv_anim_set_exec_cb(&a, anim_x_cb);
    lv_anim_set_values(&a, 10, 240);
    lv_anim_start(&a);
}

#endif
```

Animazione di cubica di Bézier

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SLIDER && LV_USE_CHART && LV_USE_BUTTON && LV_USE_GRID

/**
 * the example show the use of cubic-bezier3 in animation.
 * the control point P1,P2 of cubic-bezier3 can be adjusted by slider.
 * and the chart shows the cubic-bezier3 in real time. you can click
 * run button see animation in current point of cubic-bezier3.
 */

#define CHART_POINTS_NUM 256

struct {
    lv_obj_t * anim_obj;
    lv_obj_t * chart;
    lv_chart_series_t * ser1;
    lv_obj_t * pl_slider;
    lv_obj_t * pl_label;
    lv_obj_t * p2_slider;
    lv_obj_t * p2_label;
    lv_obj_t * run_btn;
    int32_t p1;
    int32_t p2;
    lv_anim_t a;
} ginfo;

static int32_t anim_path_bezier3_cb(const lv_anim_t * a);
static void refer_chart_cubic_bezier(void);
static void run_button_event_handler(lv_event_t * e);
static void slider_event_cb(lv_event_t * e);
static void page_obj_init(lv_obj_t * par);
static void anim_x_cb(void * var, int32_t v);

/**
 * create an animation
 */
void lv_example_anim_3(void)
{
    static int32_t col_dsc[] = {LV_GRID_FR(1), 200, LV_GRID_FR(1), LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {30, 10, 10, LV_GRID_FR(1), LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_style_pad_all(cont, 2, LV_PART_MAIN);
    lv_obj_set_style_pad_column(cont, 10, LV_PART_MAIN);
    lv_obj_set_style_pad_row(cont, 10, LV_PART_MAIN);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);
    lv_obj_set_size(cont, 320, 240);
    lv_obj_center(cont);

    page_obj_init(cont);

    lv_anim_init(&ginfo.a);
    lv_anim_set_var(&ginfo.a, ginfo.anim_obj);
    int32_t end = lv_obj_get_style_width(cont, LV_PART_MAIN) -
        lv_obj_get_style_width(ginfo.anim_obj, LV_PART_MAIN) - 10;
    lv_anim_set_values(&ginfo.a, 5, end);
    lv_anim_set_duration(&ginfo.a, 2000);
    lv_anim_set_exec_cb(&ginfo.a, anim_x_cb);
    lv_anim_set_path_cb(&ginfo.a, anim_path_bezier3_cb);

    refer_chart_cubic_bezier();
}

static int32_t anim_path_bezier3_cb(const lv_anim_t * a)
{
    int32_t t = lv_map(a->act_time, 0, a->duration, 0, 1024);
    int32_t step = lv_bezier3(t, 0, ginfo.p1, ginfo.p2, 1024);
    int32_t new_value;
    new_value = step * (a->end_value - a->start_value);
    new_value = new_value >> 10;
    new_value += a->start_value;
    return new_value;
}

static void refer_chart_cubic_bezier(void)
{
    for(uint16_t i = 0; i <= CHART_POINTS_NUM; i++) {
        int32_t t = i * (1024 / CHART_POINTS_NUM);
        int32_t step = lv_bezier3(t, 0, ginfo.p1, ginfo.p2, 1024);
        lv_chart_set_series_value_by_id2(ginfo.chart, ginfo.ser1, i, t, step);
    }
    lv_chart_refresh(ginfo.chart);
}

static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_style_translate_x((lv_obj_t *)var, v, LV_PART_MAIN);
}

static void run_button_event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_CLICKED) {
        lv_anim_start(&ginfo.a);
    }
}

static void slider_event_cb(lv_event_t * e)
{
    char buf[16];
    lv_obj_t * label;
    lv_obj_t * slider = lv_event_get_target_obj(e);

    if(slider == ginfo.pl_slider) {
        label = ginfo.pl_label;
        ginfo.p1 = lv_slider_get_value(slider);
        lv_snprintf(buf, sizeof(buf), "p1:%d", ginfo.p1);
    }
    else {
        label = ginfo.p2_label;
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

ginfo.p2 = lv_slider_get_value(slider);
lv_snprintf(buf, sizeof(buf), "p2:%d", ginfo.p2);
}

lv_label_set_text(label, buf);
refer_chart_cubic_bezier();
}

static void page_obj_init(lv_obj_t * par)
{
    ginfo.anim_obj = lv_obj_create(par);
    lv_obj_set_size(ginfo.anim_obj, 30, 30);
    lv_obj_set_align(ginfo.anim_obj, LV_ALIGN_TOP_LEFT);
    lv_obj_remove_flag(ginfo.anim_obj, LV_OBJ_FLAG_SCROLLABLE);
    lv_obj_set_style_bg_color(ginfo.anim_obj, lv_palette_main(LV_PALETTE_RED), LV_PART_MAIN);
    lv_obj_set_grid_cell(ginfo.anim_obj, LV_GRID_ALIGN_START, 0, 1, LV_GRID_ALIGN_START, 0, 1);

    ginfo.p1_label = lv_label_create(par);
    ginfo.p2_label = lv_label_create(par);
    lv_label_set_text(ginfo.p1_label, "p1:0");
    lv_label_set_text(ginfo.p2_label, "p2:0");
    lv_obj_set_grid_cell(ginfo.p1_label, LV_GRID_ALIGN_START, 0, 1, LV_GRID_ALIGN_START, 1, 1);
    lv_obj_set_grid_cell(ginfo.p2_label, LV_GRID_ALIGN_START, 0, 1, LV_GRID_ALIGN_START, 2, 1);

    ginfo.p1_slider = lv_slider_create(par);
    ginfo.p2_slider = lv_slider_create(par);
    lv_slider_set_range(ginfo.p1_slider, 0, 1024);
    lv_slider_set_range(ginfo.p2_slider, 0, 1024);
    lv_obj_set_style_pad_all(ginfo.p1_slider, 2, LV_PART_KNOB);
    lv_obj_set_style_pad_all(ginfo.p2_slider, 2, LV_PART_KNOB);
    lv_obj_add_event_cb(ginfo.p1_slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_add_event_cb(ginfo.p2_slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_set_grid_cell(ginfo.p1_slider, LV_GRID_ALIGN_STRETCH, 1, 1, LV_GRID_ALIGN_START, 1, 1);
    lv_obj_set_grid_cell(ginfo.p2_slider, LV_GRID_ALIGN_STRETCH, 1, 1, LV_GRID_ALIGN_START, 2, 1);

    ginfo.run_btn = lv_button_create(par);
    lv_obj_add_event_cb(ginfo.run_btn, run_button_event_handler, LV_EVENT_CLICKED, NULL);
    lv_obj_t * btn_label = lv_label_create(ginfo.run_btn);
    lv_label_set_text(btn_label, LV_SYMBOL_PLAY);
    lv_obj_center(btn_label);
    lv_obj_set_grid_cell(ginfo.run_btn, LV_GRID_ALIGN_STRETCH, 2, 1, LV_GRID_ALIGN_STRETCH, 1, 2);

    ginfo.chart = lv_chart_create(par);
    lv_obj_set_style_pad_all(ginfo.chart, 0, LV_PART_MAIN);
    lv_obj_set_style_size(ginfo.chart, 0, 0, LV_PART_INDICATOR);
    lv_chart_set_type(ginfo.chart, LV_CHART_TYPE_SCATTER);
    ginfo.ser1 = lv_chart_add_series(ginfo.chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_set_axis_range(ginfo.chart, LV_CHART_AXIS_PRIMARY_Y, 0, 1024);
    lv_chart_set_axis_range(ginfo.chart, LV_CHART_AXIS_PRIMARY_X, 0, 1024);
    lv_chart_set_point_count(ginfo.chart, CHART_POINTS_NUM);
    lv_obj_set_grid_cell(ginfo.chart, LV_GRID_ALIGN_STRETCH, 0, 3, LV_GRID_ALIGN_STRETCH, 3, 1);
}
#endif

```

Pausa animazione

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_x((lv_obj_t *)var, v);
}

static void timer_cb(lv_timer_t * timer)
{
    lv_anim_t * anim = (lv_anim_t *)lv_timer_get_user_data(timer);
    lv_anim_pause_for(anim, 1000);
    lv_timer_delete(timer);
}

static void sw_event_cb(lv_event_t * e)
{
    lv_obj_t * sw = lv_event_get_target_obj(e);
    lv_obj_t * label = (lv_obj_t *)lv_event_get_user_data(e);

    if(lv_obj_has_state(sw, LV_STATE_CHECKED)) {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), 100);
        lv_anim_set_duration(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_overshoot);
        lv_timer_create(timer_cb, 200, lv_anim_start(&a));
    }
    else {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), -lv_obj_get_width(label));
        lv_anim_set_duration(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_ease_in);
        lv_timer_create(timer_cb, 200, lv_anim_start(&a));
    }
}

/**
 * Start animation on an event
 */
void lv_example_anim_4(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Hello animations!");
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_pos(label, 100, 10);

lv_obj_t * sw = lv_switch_create(lv_screen_active());
lv_obj_center(sw);
lv_obj_add_state(sw, LV_STATE_CHECKED);
lv_obj_add_event_cb(sw, sw_event_cb, LV_EVENT_VALUE_CHANGED, label);
}

#endif

```

Timeline animazione

```

#include "../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

static const int32_t obj_width = 90;
static const int32_t obj_height = 70;

static void set_width(lv_anim_t * var, int32_t v)
{
    lv_obj_set_width((lv_obj_t *) var->var, v);
}

static void set_height(lv_anim_t * var, int32_t v)
{
    lv_obj_set_height((lv_obj_t *) var->var, v);
}

static void set_slider_value(lv_anim_t * var, int32_t v)
{
    lv_slider_set_value((lv_obj_t *) var->var, v, LV_ANIM_OFF);
}

static void btn_start_event_handler(lv_event_t * e)
{
    lv_obj_t * btn = lv_event_get_current_target_obj(e);
    lv_anim_timeline_t * anim_timeline = (lv_anim_timeline_t *) lv_event_get_user_data(e);

    bool reverse = lv_obj_has_state(btn, LV_STATE_CHECKED);
    lv_anim_timeline_set_reverse(anim_timeline, reverse);
    lv_anim_timeline_start(anim_timeline);
}

static void btn_pause_event_handler(lv_event_t * e)
{
    lv_anim_timeline_t * anim_timeline = (lv_anim_timeline_t *) lv_event_get_user_data(e);
    lv_anim_timeline_pause(anim_timeline);
}

static void slider_prg_event_handler(lv_event_t * e)
{
    lv_obj_t * slider = lv_event_get_current_target_obj(e);
    lv_anim_timeline_t * anim_timeline = (lv_anim_timeline_t *) lv_event_get_user_data(e);
    int32_t progress = lv_slider_get_value(slider);
    lv_anim_timeline_set_progress(anim_timeline, (uint16_t) progress);
}

/***
 * Create an animation timeline
 */
void lv_example_anim_timeline_1(void)
{
    /* Create anim timeline */
    lv_anim_timeline_t * anim_timeline = lv_anim_timeline_create();

    lv_obj_t * par = lv_screen_active();
    lv_obj_set.flex_flow(par, LV_FLEX_FLOW_ROW);
    lv_obj.set.flex_align(par, LV_FLEX_ALIGN_SPACE_AROUND, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    /* create btn_start */
    lv_obj_t * btn_start = lv_button_create(par);
    lv_obj.add.event_cb(btn_start, btn_start_event_handler, LV_EVENT_VALUE_CHANGED, anim_timeline);
    lv_obj.add.flag(btn_start, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj.align(btn_start, LV_ALIGN_TOP_MID, -100, 20);

    lv_obj_t * label_start = lv_label_create(btn_start);
    lv_label.set.text(label_start, "Start");
    lv_obj_center(label_start);

    /* create btn_pause */
    lv_obj_t * btn_pause = lv_button_create(par);
    lv_obj.add.event_cb(btn_pause, btn_pause_event_handler, LV_EVENT_CLICKED, anim_timeline);
    lv_obj.add.flag(btn_pause, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj.align(btn_pause, LV_ALIGN_TOP_MID, 100, 20);

    lv_obj_t * label_pause = lv_label_create(btn_pause);
    lv_label.set.text(label_pause, "Pause");
    lv_obj_center(label_pause);

    /* create slider_prg */
    lv_obj_t * slider_prg = lv_slider_create(par);
    lv_obj.add.event_cb(slider_prg, slider_prg_event_handler, LV_EVENT_VALUE_CHANGED, anim_timeline);
    lv_obj.add.flag(slider_prg, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj.align(slider_prg, LV_ALIGN_BOTTOM_MID, 0, -20);
    lv_slider.set.range(slider_prg, 0, LV_ANIM_TIMELINE_PROGRESS_MAX);

    /* create 3 objects */
    lv_obj_t * obj1 = lv_obj_create(par);
    lv_obj.set.size(obj1, obj_width, obj_height);
    lv_obj.set.scrollbar_mode(obj1, LV_SCROLLBAR_MODE_OFF);

    lv_obj_t * obj2 = lv_obj_create(par);
    lv_obj.set.size(obj2, obj_width, obj_height);
    lv_obj.set.scrollbar_mode(obj2, LV_SCROLLBAR_MODE_OFF);

    lv_obj_t * obj3 = lv_obj_create(par);
    lv_obj.set.size(obj3, obj_width, obj_height);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_scrollbar_mode(obj3, LV_SCROLLBAR_MODE_OFF);

/* anim-slider */
lv_anim_t a_slider;
lv_anim_init(&a_slider);
lv_anim_set_var(&a_slider, slider_prg);
lv_anim_set_values(&a_slider, 0, LV_ANIM_TIMELINE_PROGRESS_MAX);
lv_anim_set_custom_exec_cb(&a_slider, set_slider_value);
lv_anim_set_path_cb(&a_slider, lv_anim_path_linear);
lv_anim_set_duration(&a_slider, 700);

/* anim-obj1 */
lv_anim_t a1;
lv_anim_init(&a1);
lv_anim_set_var(&a1, obj1);
lv_anim_set_values(&a1, 0, obj_width);
lv_anim_set_custom_exec_cb(&a1, set_width);
lv_anim_set_path_cb(&a1, lv_anim_path_overshoot);
lv_anim_set_duration(&a1, 300);

lv_anim_t a2;
lv_anim_init(&a2);
lv_anim_set_var(&a2, obj1);
lv_anim_set_values(&a2, 0, obj_height);
lv_anim_set_custom_exec_cb(&a2, set_height);
lv_anim_set_path_cb(&a2, lv_anim_path_ease_out);
lv_anim_set_duration(&a2, 300);

/* anim-obj2 */
lv_anim_t a3;
lv_anim_init(&a3);
lv_anim_set_var(&a3, obj2);
lv_anim_set_values(&a3, 0, obj_width);
lv_anim_set_custom_exec_cb(&a3, set_width);
lv_anim_set_path_cb(&a3, lv_anim_path_overshoot);
lv_anim_set_duration(&a3, 300);

lv_anim_t a4;
lv_anim_init(&a4);
lv_anim_set_var(&a4, obj2);
lv_anim_set_values(&a4, 0, obj_height);
lv_anim_set_custom_exec_cb(&a4, set_height);
lv_anim_set_path_cb(&a4, lv_anim_path_ease_out);
lv_anim_set_duration(&a4, 300);

/* anim-obj3 */
lv_anim_t a5;
lv_anim_init(&a5);
lv_anim_set_var(&a5, obj3);
lv_anim_set_values(&a5, 0, obj_width);
lv_anim_set_custom_exec_cb(&a5, set_width);
lv_anim_set_path_cb(&a5, lv_anim_path_overshoot);
lv_anim_set_duration(&a5, 300);

lv_anim_t a6;
lv_anim_init(&a6);
lv_anim_set_var(&a6, obj3);
lv_anim_set_values(&a6, 0, obj_height);
lv_anim_set_custom_exec_cb(&a6, set_height);
lv_anim_set_path_cb(&a6, lv_anim_path_ease_out);
lv_anim_set_duration(&a6, 300);

/* add animations to timeline */
lv_anim_timeline_add(anim_timeline, 0, &a_slider);
lv_anim_timeline_add(anim_timeline, 0, &a1);
lv_anim_timeline_add(anim_timeline, 0, &a2);
lv_anim_timeline_add(anim_timeline, 200, &a3);
lv_anim_timeline_add(anim_timeline, 200, &a4);
lv_anim_timeline_add(anim_timeline, 400, &a5);
lv_anim_timeline_add(anim_timeline, 400, &a6);

lv_anim_timeline_set_progress(anim_timeline, LV_ANIM_TIMELINE_PROGRESS_MAX);
}

#endif

```

7.7.10 API

7.8 File System (lv_fs_drv)

LVGL dispone di un modulo di astrazione del "File system" che consente di collegare qualsiasi tipo di file system. Un file system è identificato da una lettera identificativa assegnata. Ad esempio, se una scheda SD è associata alla lettera 'S', un file può essere raggiunto utilizzando "S:/path/to/file.txt". Vedere i dettagli in [Lettere Identificative](#).

Nota

Per ignorare il prefisso della lettera di unità nei percorsi di tipo Unix, è possibile utilizzare il parametro di configurazione `LV_FS_DEFAULT_DRIVER LETTER`.

7.8.1 Driver Pronti all'Uso

LVGL contiene driver pronti per le API di POSIX, C standard, Windows e [FATFS](#). Per saperne di più [qui](#).

7.8.2 Lettere Identificative

Come accennato in precedenza, un file system è identificato da una lettera identificativa assegnata. Questo identificativo è semplicemente un modo per la logica di astrazione del file system LVGL di cercare il driver del file system registrato appropriato per un determinato percorso.

Come Funziona:

Si registra un driver per il file system e gli si assegna una lettera identificativa. Questa lettera deve essere univoca tra tutti i driver del file system registrati e compresa nell'intervallo [A-Z] o nel carattere '/'. Vedere [Aggiunta di un Driver](#) per informazioni su come procedere.

In seguito, quando si utilizzano path per i file sul file system, si antepone al percorso quel carattere identificativo più due punti (':').

Nota

Non confondere questa con una lettera di unità di Windows o DOS.

Esempio:

Usiamo la lettera 'Z' come carattere identificativo e "path_to_file" come percorso; le stringhe di percorso passate alle funzioni `lv_fs_...` saranno simili a queste:

```
"Z:path_to_file"
^ ~~~~~
|   |
|   +-- This part gets passed to the OS-level file-system functions.
|
+-- This part LVGL strips from path string, and uses it to find the appropriate
    driver (i.e. set of functions) that apply to that file system.
```

Si noti inoltre che il path può essere un percorso relativo o un "rooted path" (che inizia con /), i path rooted sono consigliati poiché il driver non fornisce ancora un modo per impostare la directory di default.

Esempi per file system di tipo Unix:

- "Z:/etc/images/splash.png"
- "Z:/etc/images/left_button.png"
- "Z:/etc/images/right_button.png"
- "Z:/home/users/me/wip/proposal.txt"

Esempi per file system di tipo Windows/DOS:

- "Z:C:/Users/me/wip/proposal.txt"
- "Z:/Users/me/wip/proposal.txt" (se l'unità predefinita è C:)
- "Z:C:/Users/Public/Documents/meeting_notes.txt"
- "Z:D:/to_print.docx"

Promemoria: Si noti che il prefisso "Z:" non ha nulla a che fare con le lettere di unità Windows/DOS "C:" e "D:" in 3 dei file esempi sopra, che fanno parte del path. "Z:" viene utilizzato per cercare il driver per quel file system nell'elenco di tutti i driver di file system registrati con LVGL.

7.8.3 Aggiunta di un Driver

Registrazione di un driver

Per aggiungere un driver, è necessario inizializzare e registrare un oggetto `lv_fs_drv_t` in modo simile al codice seguente. La variabile `lv_fs_drv_t` deve essere statica, globale o allocata dinamicamente e non locale, poiché il suo contenuto deve rimanere valido finché il driver è in uso.

```
static lv_fs_drv_t drv;           /* Needs to be static or global */
lv_fs_drv_init(&drv);           /* Basic initialization */

drv.letter = 'S';                /* An uppercase letter to identify the drive */
drv.cache_size = my_cache_size;   /* Cache size for reading in bytes. 0 to not cache. */

drv.ready_cb = my_ready_cb;      /* Callback to tell if the drive is ready to use */
drv.open_cb = my_open_cb;         /* Callback to open a file */
drv.close_cb = my_close_cb;       /* Callback to close a file */
drv.read_cb = my_read_cb;         /* Callback to read a file */
drv.write_cb = my_write_cb;       /* Callback to write a file */
drv.seek_cb = my_seek_cb;         /* Callback to seek in a file (Move cursor) */
drv.tell_cb = my_tell_cb;         /* Callback to tell the cursor position */

drv.dir_open_cb = my_dir_open_cb; /* Callback to open directory to read its content */
drv.dir_read_cb = my_dir_read_cb; /* Callback to read a directory's content */
drv.dir_close_cb = my_dir_close_cb; /* Callback to close a directory */

drv.user_data = my_user_data;    /* Any custom data if required */

lv_fs_drv_register(&drv);        /* Finally register the drive */
```

Qualsiasi callback può essere `NULL` per indicare che l'operazione non è supportata.

Implementazione delle callback

Callback Open

Il prototipo di `open_cb` si presenta così:

```
void * (*open_cb)(lv_fs_drv_t * drv, const char * path, lv_fs_mode_t mode);
```

`path` è il percorso dopo la lettera di unità (ad esempio "S:path/to/file.txt" -> "path/to/file.txt"). `mode` può essere `LV_FS_MODE_WR` o `LV_FS_MODE_RD` per aprire in scrittura o lettura.

Il valore restituito è un puntatore a un *oggetto file* che descrive il file aperto oppure `NULL` in caso di problemi (ad esempio, il file non è stato trovato). L'oggetto file restituito verrà passato ad altre callback relative al file system. (Vedere di seguito).

Altre callback

Le altre callback sono piuttosto simili. Ad esempio, `write_cb` si presenta così:

```
lv_fs_res_t (*write_cb)(lv_fs_drv_t * drv, void * file_p, const void * buf, uint32_t btw, uint32_t * bw);
```

Per `file_p`, LVGL passa il valore di ritorno di `open_cb`, `buf` sono i dati da scrivere, `btw` è il numero di "byte da scrivere", `bw` è il numero di "byte scritti" (scritti durante la chiamata alla funzione).

Per un elenco di prototipi per queste callback, vedere `lv_fs_template.c`. Questo file fornisce anche un template per i nuovi driver del file system che è possibile utilizzare se quello necessario non è già disponibile.

Driver inclusi in LVGL

Al momento della stesura di questo documento, l'elenco dei driver del file system già disponibili può essere abilitato impostando una o più delle seguenti macro su un valore diverso da zero in `lv_conf.h`. I driver sono quelli indicati dai nomi delle macro.

Se ne viene usato più di uno, ogni lettera identificativa associata deve essere univoca.

- `LV_USE_FS_FATFS`
- `LV_USE_FS_STUDIO`
- `LV_USE_FS_POSIX`
- `LV_USE_FS_WIN32`
- `LV_USE_FS_MEMFS`
- `LV_USE_FS_LITTLEFS`

- *LV_USE_FS_ARDUINO_ESP_LITTLEFS*
- *LV_USE_FS_ARDUINO_SD*

7.8.4 Limitazione dell'Accesso alle Directory

Se si utilizza uno dei seguenti driver del file system:

- *LV_USE_FS_STDIO*
- *LV_USE_FS_POSIX*
- *LV_USE_FS_WIN32*

sarà disponibile una macro *LV_FS_xxx_PATH* in *lv_conf.h* utilizzabile per fornire un path che viene aggiunto dinamicamente come prefisso alla porzione *path_to_file* delle stringhe di path fornite alle funzioni *lv_fs_...* () quando vengono aperti file e directory. Questo può essere utile per limitare l'accesso alle directory (ad esempio, quando una parte di un path può essere digitata da un utente finale) o semplicemente per ridurre la lunghezza delle stringhe dei path fornite alle funzioni *lv_fs_...* ().

Per farlo, si inserisce il path completo della directory a cui si desidera limitare l'accesso nella macro *LV_FS_xxx_PATH* in *lv_conf.h*. Non anteporre al path la lettera identificativa del driver e aggiungere un carattere separatore di directory alla fine.

Esempi per file system di tipo Unix:

```
#define LV_FS_WIN32_PATH "/home/users/me/"
```

Esempi per file system di tipo Windows/DOS:

```
#define LV_FS_WIN32_PATH "C:/Users/me/"
```

In entrambi i casi, le stringhe del path passate alle funzioni *lv_fs_...* () nell'applicazione vengono ridotte a:

- "Z:wip/proposal.txt"

7.8.5 Esempio di Utilizzo

L'esempio seguente mostra come leggere da un file:

```
lv_fs_file_t f;
lv_fs_res_t res;
res = lv_fs_open(&f, "S:folder/file.txt", LV_FS_MODE_RD);
if(res != LV_FS_RES_OK) my_error_handling();

uint32_t read_num;
uint8_t buf[8];
res = lv_fs_read(&f, buf, 8, &read_num);
if(res != LV_FS_RES_OK || read_num != 8) my_error_handling();

lv_fs_close(&f);
```

La modalità in *lv_fs_open()* può essere *LV_FS_MODE_WR* per aprire solo in scrittura, *LV_FS_MODE_RD* per solo in lettura o *LV_FS_MODE_RD* | *LV_FS_MODE_WR* per entrambe.

Questo esempio mostra come leggere il contenuto di una directory. Spetta al driver decidere come contrassegnare le directory nel risultato, ma può essere una buona pratica inserire '/' davanti al nome di ogni directory.

```
lv_fs_dir_t dir;
lv_fs_res_t res;
res = lv_fs_dir_open(&dir, "S:/folder");
if(res != LV_FS_RES_OK) my_error_handling();

char fn[256];
while(1) {
    res = lv_fs_dir_read(&dir, fn, sizeof(fn));
    if(res != LV_FS_RES_OK) {
        my_error_handling();
        break;
    }

    /* fn is empty if there are no more files to read. */
    if(strlen(fn) == 0) {
        break;
    }

    printf("%s\n", fn);
}

lv_fs_dir_close(&dir);
```

7.8.6 Utilizzare i Drive per le Immagini

I Widget *Image* possono essere aperti anche da file (oltre che dalle variabili memorizzate nel programma compilato).

Per utilizzare i file nei Widget Image sono necessarie le seguenti callback:

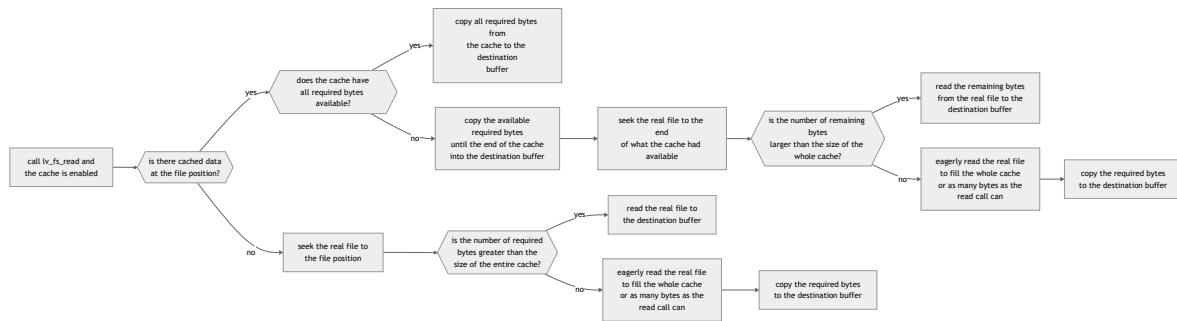
- open
- close
- read
- seek
- tell

7.8.7 Buffering/caching Opzionale dei File

I file memorizzeranno le letture nel buffer se l'opzione di configurazione `LV_FS_*_CACHE_SIZE` corrispondente è impostata su un valore maggiore di zero. Ogni file aperto memorizzerà nel buffer fino a quel numero di byte per ridurre il numero di chiamate al driver FS.

In generale, il buffering dei file può essere ottimizzato per diversi tipi di modelli di accesso. Quello implementato qui è ottimale per la lettura di file di grandi dimensioni in blocchi, che è ciò che fa il decoder di immagini. Ha il potenziale di chiamare `read` del driver meno volte di quante ne venga chiamata `lv_fs_read`. Nel caso migliore, quando la dimensione della cache è \geq la dimensione del file, `read` verrà chiamata una sola volta. Questa strategia è valida per la lettura lineare di file di grandi dimensioni, ma meno utile per letture "random" brevi su un file più grande del buffer, poiché verranno memorizzati nel buffer dati che verranno scartati dopo la successiva ricerca e lettura. In tal caso, la cache dovrebbe essere sufficientemente grande o disabilitata. Un altro caso in cui la cache dovrebbe essere disabilitata è quando si prevede che il contenuto del file cambi a causa di un fattore esterno, come nel caso di file speciali del sistema operativo.

L'implementazione è documentata di seguito. Si noti che le funzioni FS effettuano chiamate ad altre funzioni FS del driver quando la cache è abilitata. ad esempio, `lv_fs_read` potrebbe chiamare `seek` del driver, quindi il driver deve implementare più callback quando la cache è abilitata.

lv_fs_read (comportamento quando la cache è abilitata)

(Usare la rotellina del mouse per ingrandire e trascinare l'immagine per spostarla).

lv_fs_write (comportamento quando la cache è abilitata)

La parte della cache che coincide con il contenuto scritto verrà aggiornata per riflettere il contenuto scritto.

lv_fs_seek (comportamento quando la cache è abilitata)

La funzione seek del driver non verrà effettivamente chiamata a meno che whence non sia LV_FS_SEEK_END, nel qual caso verranno chiamati seek e tell per determinare dove si trova la fine del file.

lv_fs_tell (comportamento quando la cache è abilitata)

La funzione tell del driver non verrà effettivamente chiamata.

7.8.8 API

lv_fdrv.h

lv_fs.h

lv_fs_private.h

7.9 Observer

7.9.1 Come si Usa

Panoramica

Il modulo `lv_observer` è un'implementazione del Pattern Observer.

Questa implementazione è composta da:

Subject

(nella memoria globale o heap) sono "pacchetti logici", ciascuno contenente il valore "osservato" e il suo tipo (intero (`int32_t`), una stringa, un puntatore, un `lv_color_t`, un `float`, o un gruppo);

Observer

(zero o più per Subject, sempre allocati dinamicamente) sono sempre associati a un solo Subject e forniscono notifiche definite dall'utente ogni volta che il valore del Subject cambia.

Un Subject e i suoi Observer possono essere utilizzati in vari modi:

1. È sufficiente iscriversi a un Subject e ricevere una notifica quando il valore del Subject cambia.
2. Ci si iscrive a un gruppo Subject (connette un gruppo di Subject) per ricevere una notifica quando uno qualsiasi dei valori dei Subject nel gruppo cambia.
3. Esegue il bind dei Widget ai Subject per abbinare automaticamente il valore del Widget al Subject (ad esempio, il testo di un'Etichetta o il valore di un Arco).

Utilizzo

Per utilizzare l'Observer, è necessario che `LV_USE_OBSERVER` sia configurato a 1. (È 1 per default e può essere impostato a 0 per risparmiare spazio nel programma se non si utilizza l'Observer).

Un tipico caso d'uso si presenta così:

```
// Any typical global variable
lv_subject_t my_subject;

/*.....
 * main.c
 *.....*/
extern lv_subject_t my_subject;

void main(void)
{
    // Initialize Subject as integer with the default value of 10.
    lv_subject_init_int(&my_subject, 10);

    some_module_init();
}
```

(continues on next page)

(continua dalla pagina precedente)

```

}

/*-----
 * some_module.c
 *-----
 */

extern lv_subject_t my_subject;

// Will be called when Subject's value changes
static void some_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    int32_t v = lv_subject_get_int(subject);
    do_something(v);
}

void some_module_init(void)
{
    // Subscribe to Subject as an Observer.
    lv_subject_add_observer(&my_subject, some_observer_cb, NULL);
}

/*-----
 * some_system.c
 *-----
 */

extern lv_subject_t my_subject;

void some_event(void)
{
    // The below call sets Subject's value to 30 and notifies current Observers.
    lv_subject_set_int(&my_subject, 30);
}

```

Subject

Inizializzazione del Subject

I Subject devono essere variabili statiche o globali, oppure oggetti `lv_subject_t` allocati dinamicamente. Motivo: il loro contenuto deve rimanere valido per tutta la vita del Subject.

Per inizializzare un Subject, si usa `lv_subject_init_<type>(&subject, params, init_value)`. Esistono le seguenti funzioni di inizializzazione, una per ciascun tipo di Subject:

Intero

```
void lv_subject_init_int(subject, int_value)
```

Float

```
void lv_subject_init_float(subject, float_value)
```

Stringa

```
void lv_subject_init_string(subject, buf, prev_buf, buf_size, initial_string)
```

Puntatore

```
void lv_subject_init_pointer(subject, ptr)
```

Color

```
void lv_subject_init_color(subject, color)
```

Gruppo

```
void lv_subject_init_group(group_subject, subject_list[], count)
```

Impostazione del Valore di un Subject

Le seguenti funzioni vengono utilizzate per aggiornare il valore di un Subject:

Intero

```
void lv_subject_set_int(subject, int_value)
```

Stringa

```
void lv_subject_copy_string(subject, buf)
```

Puntatore

```
void lv_subject_set_pointer(subject, ptr)
```

Color

```
void lv_subject_set_color(subject, color)
```

Al termine di ciascuna di queste chiamate, se il nuovo valore differisce dal valore precedente, viene inviata una notifica a tutti gli Observer correnti.

Ottenere il valore di un Subject

Le seguenti funzioni vengono utilizzate per ottenere il valore corrente di un Subject:

Intero

```
int32_t lv_subject_get_int(subject)
```

Float

```
float lv_subject_get_float(subject)
```

Stringa

```
const char * lv_subject_get_string(subject)
```

Puntatore

```
const void * lv_subject_get_pointer(subject)
```

Color

```
lv_color_t lv_subject_get_color(subject)
```

Ottenere il Valore Precedente di un Subject

Le seguenti funzioni vengono utilizzate per ottenere il valore precedente di un Subject:

Intero

```
int32_t lv_subject_get_previous_int(subject)
```

Float

```
float lv_subject_get_previous_float(subject)
```

Stringa

```
const char * lv_subject_get_previous_string(subject)
```

Puntatore

```
const void * lv_subject_get_previous_pointer(subject)
```

Color

```
lv_color_t lv_subject_get_previous_color(subject)
```

Impostazione di un Intervallo per un Subject

Il "range" [intervallo] degli oggetti interi e float può essere limitato impostando valori minimi e massimi:

- `lv_subject_set_min_value_int(subject, value)`
- `lv_subject_set_max_value_int(subject, value)`
- `lv_subject_set_min_value_float(subject, value)`
- `lv_subject_set_max_value_float(subject, value)`

L'intervallo di default è il valore minimo e massimo possibile, quindi non ci sono limiti predefiniti.

Observer

Iscrizione a un Subject

L'azione di iscrizione a un Subject:

- alloca dinamicamente un oggetto Observer,
- lo associa al Subject,
- esegue una notifica iniziale all'Observer (consentendo all'Observer di aggiornarsi con il valore corrente del Subject), e
- restituisce un puntatore all'Observer appena creato.

Successivamente, l'Observer riceverà una notifica ogni volta che il valore del Subject cambia, finché l'Observer rimane associato (iscritto) a quel Subject.

Le notifiche vengono eseguite chiamando la funzione di callback fornita al momento dell'iscrizione al Subject.

Iscrizione Semplice

Per iscriversi a un Subject si utilizza una delle funzioni `lv_subject_add_observer...()`. Questo argomento è trattato di seguito.

In alternativa, se si desidera associare il valore di un Subject alla proprietà di un Widget, è possibile utilizzare una delle funzioni `lv_<widget_type>_bind_...()`. Vedere [Binding del Widget](#) per maggiori dettagli.

Utilizzando `lv_obj_add_subject_...()` è anche possibile modificare il valore di un Subject su un trigger. È trattato in [Stili delle Sezioni](#)

Per il caso d'uso più elementare, iscriversi a un Subject utilizzando la seguente funzione:

```
lv_observer_t * observer = lv_subject_add_observer(&some_subject, some_observer_cb,
user_data)
```

dove la callback di notifica dell'Observer dovrebbe apparire così:

```
static void some_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    ...
}
```

Questa funzione restituisce un puntatore all'Observer appena creato.

Quando si utilizza questo metodo di iscrizione, è responsabilità dell'utente chiamare `lv_observer_remove(observer)` quando l'Observer non è più necessario, la quale annulla l'iscrizione al Subject e lo elimina dall'heap LVGL.

Sottoscrizione Durante l'Associazione dell'Observer a un Oggetto Non-Widget

La funzione sottoscrive un Subject e associa inoltre l'Observer a un puntatore a qualsiasi tipo di oggetto, una copia del quale viene salvata nel campo `target`. Questa funzione dovrebbe essere utilizzata quando il puntatore *non* punta a un Widget.

```
lv_observer_t * observer = lv_subject_add_observer_with_target(&some_subject,
some_observer_cb, some_pointer, user_data)
```

Una copia del puntatore passato può essere recuperata chiamando `lv_observer_get_target(observer)`, ad esempio all'interno della funzione di callback.

Quando si utilizza questo metodo di iscrizione, è responsabilità dell'utente chiamare `lv_observer_remove(observer)` quando l'Observer non è più necessario, la quale annulla l'iscrizione al Subject e lo elimina dall'heap LVGL.

Sottoscrizione Durante l'Associazione dell'Observer a un Widget

La funzione seguente associa un Widget a un Observer durante l'iscrizione a un Subject. Una copia del puntatore a quel Widget viene salvata nel campo `target`. Funziona esattamente come il metodo precedente, tranne per il fatto che, quando il Widget viene eliminato, l'Observer così creato verrà automaticamente rimosso dall'iscrizione al Subject e cancellato dall'heap LVGL. Si noti che questo è diverso dal [Binding del Widget](#).

```
lv_observer_t * observer = lv_subject_add_observer_obj(&some_subject, some_observer_cb,
widget, user_data)
```

In questo modo è possibile creare un numero qualsiasi di Observer e associarli a un Widget.

Una copia del puntatore al Widget può essere recuperata chiamando `lv_observer_get_target_obj(observer)`, ad esempio all'interno della funzione di callback. Si noti che questa funzione restituisce il puntatore memorizzato come tipo `lv_obj_t *`, a differenza del tipo `void *` restituito da `lv_observer_get_target(observer)`. (`lv_observer_get_target(observer)` può comunque essere utilizzato se si necessita di quel puntatore come tipo `void *` per qualsiasi motivo, ma nella pratica, è raro).

Annullo dell'iscrizione a un Subject

Quando un Observer e gli eventi associati non sono più necessari, è possibile rimuoverli dalla memoria utilizzando uno dei metodi seguenti, a seconda dell'effetto desiderato:

- Se è necessario eliminare un Widget, è sufficiente farlo, il che rimuoverà automaticamente e correttamente l'Observer (e i suoi eventi) dal Subject, eliminando al contempo il Widget.
- Se il Widget NON deve essere eliminato:
 - `lv_obj_remove_from_subject(widget, subject)` elimina tutti gli Observer associati a `widget`. `subject` può essere NULL per annullare l'iscrizione del Widget da tutti i Subject associati.
 - `lv_subject_deinit(subject)` disconnette automaticamente `subject` da tutti gli Observer e gli eventi Widget associati. Ciò include le sottoscrizioni effettuate utilizzando una qualsiasi delle funzioni *Binding del Widget* descritte di seguito.
 - `lv_observer_remove(observer)` elimina quello specifico Observer e lo disconnette automaticamente dal suo `subject` e da tutti i Widget associati, dove `observer` è il valore restituito da una qualsiasi delle funzioni `lv_subject_add_observer_...()` sopra.

Gruppi di Subject

Quando un elemento del sistema si basa su più di un valore (ad esempio, deve essere avvisato quando cambia uno qualsiasi di un SET di due o più valori), può essere impostato come Observer di un Gruppo di Subject.

Consideriamo l'esempio di uno strumento che misura tensione o corrente. Per visualizzare il valore misurato su un'etichetta, sono necessari 3 elementi:

1. Cosa viene misurato (corrente o tensione)?
2. Qual è il valore misurato?
3. Qual è l'intervallo o l'unità ("mV", "V", "mA", "A")?

Quando uno qualsiasi di questi 3 valori di input cambia, l'etichetta deve essere aggiornata e deve conoscere tutti e 3 i valori per comporre il suo testo.

Per gestire questa situazione, è possibile creare un array con gli indirizzi di tutti i Subject su cui si basa e passare tale array come parametro quando si inizializza un Subject con tipo GROUP.

```
static lv_subject_t * subject_list[3] = {&subject_1, &subject_2, &subject_3};
lv_subject_init_group(&subject_all, subject_list, 3); /* Last argument is number of elements. */
```

Gli Observer vengono poi aggiunti ai Gruppi di Subject (ad esempio `subject_all`) nel modo consueto. Una volta fatto questo, una modifica al valore di uno qualsiasi dei Subject del gruppo attiva una notifica a tutti gli Observer iscritti al Gruppo Subject (ad esempio `subject_all`).

Ad esempio, lo scenario sopra descritto con la misurazione di Tensione/Corrente potrebbe apparire così:

```
lv_obj_t * label = lv_label_create(lv_screen_active());

lv_subject_t subject_mode; // Voltage or Current
lv_subject_t subject_value; // Measured value
lv_subject_t subject_unit; // The unit
lv_subject_t subject_all; // Subject group that connects the above 3 Subjects
lv_subject_t * subject_list[3] = {&subject_mode, &subject_value, &subject_unit}; // The elements of the group

lv_subject_init_int(&subject_mode, 0); // Let's say 0 is Voltage, 1 is Current
lv_subject_init_int(&subject_value, 0);
lv_subject_init_pointer(&subject_unit, "V");
lv_subject_init_group(&subject_all, subject_list, 3);

lv_subject_add_observer_obj(&subject_all, all_observer_cb, label, NULL);

...

static void all_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    lv_obj_t * label = lv_observer_get_target(observer);
    lv_subject_t * subject_mode = lv_subject_get_group_element(subject, 0);
    lv_subject_t * subject_value = lv_subject_get_group_element(subject, 1);
    lv_subject_t * subject_unit = lv_subject_get_group_element(subject, 2);

    int32_t mode = lv_subject_get_int(subject_mode);
    int32_t value = lv_subject_get_int(subject_value);
    const char * unit = lv_subject_get_pointer(subject_unit);

    lv_label_set_text_fmt(label, "%s: %d %s", mode ? "Current" : "Voltage", value, unit);
}
```

Binding del Widget

I seguenti metodi di sottoscrizione a un Subject intero associano l'Observer a UNA delle proprietà di un Widget, come se la proprietà stessa fosse l'Observer. Qualsiasi delle seguenti proprietà del Widget può essere quindi associata al valore intero di un Subject:

- flag (o combinazione di flag con OR) dai valori dell'enumerazione `LV_OBJ_FLAG_...`;
- stato (o combinazione di stati con OR) dai valori dell'enumerazione `LV_STATE_...`;
- **valori di testo e/o interi per**
 - Etichetta
 - Span;
 - Arco
 - Drop-Down
 - Roller
 - Slider
 - Valori Min/Max della Sezione "Scale"

È possibile creare un numero qualsiasi di Observer per un singolo Widget, ciascuno associato a UNA delle proprietà sopra indicate.

Tutte le funzioni `lv_..._bind_...()` descritte di seguito sono simili a `lv_subject_add_observer_obj(&some_subject, some_observer_cb, widget, user_data)` in quanto creano un Observer e vi associano il Widget. La differenza è che gli aggiornamenti alla proprietà del Widget così associato vengono gestiti internamente: l'utente *non fornisce funzioni di callback* per nessuno di questi metodi di sottoscrizione; i metodi di callback sono forniti dal sottosistema di Observer.

i Nota

Sebbene gli esempi seguenti mostrino il salvataggio di un riferimento agli oggetti Observer creati per le varie funzioni `lv_..._bind_...()`, non è necessario farlo a meno che non lo siano per qualche scopo, poiché gli oggetti Observer creati verranno eliminati automaticamente quando il Widget viene eliminato.

Qualsiasi Tipo di Widget

I Flag

I seguenti metodi di sottoscrizione a un Subject intero influenzano il flag di un Widget (o una combinazione di flag tramite OR). Quando si verifica la sottoscrizione, e ogni volta che il valore del Subject viene modificato successivamente, il valore del Subject viene confrontato con il valore di riferimento specificato e il/i flag specificato/i è/sono:

- SET quando il valore intero del Subject soddisfa la condizione indicata e
- CLEARED in caso contrario.

Ecco le funzioni che eseguono questo metodo di sottoscrizione a un Subject. L'argomento `flags` può contenere un singolo valore o una combinazione tramite OR bit a bit di uno qualsiasi dei valori dell'enumerazione `LV_OBJ_FLAG_`.

...

uguale

```
lv_obj_bind_flag_if_eq(widget, &subject, flags, ref_value)
```

diverso

```
lv_obj_bind_flag_if_not_eq(widget, &subject, flags, ref_value)
```

maggiore di

```
lv_obj_bind_flag_if_gt(widget, &subject, flags, ref_value)
```

maggior o uguale

```
lv_obj_bind_flag_if_ge(widget, &subject, flags, ref_value)
```

minore di

```
lv_obj_bind_flag_if_lt(widget, &subject, flags, ref_value)
```

minore o uguale di

```
lv_obj_bind_flag_if_le(widget, &subject, flags, ref_value)
```

Stati

I seguenti metodi di sottoscrizione a un Subject intero influenzano gli stati di un Widget (o una combinazione di stati con OR). Quando avviene la sottoscrizione, e ogni volta che il valore del Subject viene modificato successivamente, tale valore viene confrontato con quello di riferimento specificato e lo stato/gli stati specificati sono:

- SET quando il valore intero del Subject soddisfa la condizione indicata e
- CLEARED in caso contrario.

Ecco le funzioni che eseguono questo metodo di sottoscrizione a un Subject. L'argomento `states` può contenere un singolo valore o una combinazione con OR bit a bit di uno qualsiasi dei valori dell'enumerazione `LV_STATE_...`

uguale

```
lv_obj_bind_state_if_eq(widget, &subject, states, ref_value)
```

diverso

```
lv_obj_bind_state_if_not_eq(widget, &subject, states, ref_value)
```

maggior o uguale

```
lv_obj_bind_state_if_ge(widget, &subject, states, ref_value)
```

minore di

```
lv_obj_bind_state_if_lt(widget, &subject, states, ref_value)
```

minore o uguale di

```
lv_obj_bind_state_if_le(widget, &subject, states, ref_value)
```

Stato Checked

Il seguente metodo di sottoscrizione a un Subject intero influisce sullo stato `LV_STATE_CHECKED` di un Widget. Quando si verifica la sottoscrizione, e ogni volta che il valore del Subject viene modificato successivamente, il valore del Subject viene confrontato con un valore di riferimento di 0, e lo stato `LV_STATE_CHECKED` è:

- CLEARED quando il valore del Subject è 0 e
- SET quando il valore intero del Subject è diverso da zero.

Si noti che si tratta di un binding bidirezionale (Subject <====> Widget), quindi l'interazione diretta (o programmatica) con il Widget che determina il suo stato `LV_STATE_CHECKED` su SET o CLEARED determina anche l'impostazione del valore del Subject rispettivamente su 1 o 0.

- `lv_obj_bind_checked(widget, &subject)`

Tipi di Widget Specifici

Per informazioni su come associare gli oggetti ad Archi, Etichette, Slider, ecc., consultare la sezione "Data binding" della documentazione del widget in questione. Ad esempio: [Data binding per lv_label](#).

Modifica del Subject con l'Evento

È un requisito comune aggiornare un Subject in base a un'azione dell'utente (trigger). Per semplificare, le azioni *subject set* e *increment* possono essere associate direttamente a qualsiasi widget.

Internamente, queste sono implementate come callback di eventi speciali. Nota: queste callback **non** vengono rimosse automaticamente quando un soggetto viene deinizializzato.

Attiva/Disattiva

```
lv_obj_add_subject_toggle_event(obj, subject)
```

Scambia il valore dei "Subject" quando si verifica l'attivazione. Se il valore non era 0, sarà 0 Se era 0 sarà 1.

Ad esempio:

```
lv_obj_add_subject_toggle_event(button1, subject1, LV_EVENT_CLICKED)
```

Questo alterna *subject1* tra 0 e 1 ogni volta che si clicca su *button1*.

Incremento

lv_obj_add_subject_increment_event(obj, subject, trigger, step) incrementa un valore di un Subject intero di *step*.

Restituisce un puntatore *lv_subject_increment_dsc_t* per configurare ulteriormente l'evento:

- *lv_obj_set_subject_increment_event_min_value*(obj, dsc, min_value): Imposta un valore minimo per l'evento. Il default è INT32_MIN
- *lv_obj_set_subject_increment_event_max_value*(obj, dsc, max_value): Imposta un valore massimo per l'evento. Il default è INT32_MAX
- *lv_obj_set_subject_increment_event_rollover*(obj, dsc, rollover): Imposta cosa fare quando viene superato il valore min/max. false: si ferma al valore min/max; true: salta all'altro estremo. Il default è false.

Funziona sia con *subject* interi che float, ma il valore min/max è un intero in entrambi i casi.

Se anche il *subject* impone un valore min/max, verrà utilizzato l'intervallo più ristretto.

Utilizzando uno *step* negativo, il valore verrà invece diminuito.

Ad esempio:

```
lv_obj_add_subject_increment_event(button1, subject1, LV_EVENT_CLICKED, 5)
```

Questo incrementerà *subject1* di 5 quando si clicca su *button1*, fermandosi ai limiti impostati dai valori min/max dell'evento o da *lv_subject_set_min_value_int()* e *lv_subject_set_max_value_int()* (o stesso vale per i *subject* float).

Impostare su un Valore

- *lv_obj_add_subject_set_int_event*(obj, subject, trigger, value)
- *lv_obj_add_subject_set_string_event*(obj, subject, trigger, text)

Queste funzioni impostano l'oggetto specificato (intero o stringa) su un valore fisso quando si verifica l'evento trigger specificato.

API

lv_observer.h

lv_observer_private.h

7.9.2 Esempi

Associare il valore di uno slider a un'etichetta

```
#include "../../lv_examples.h"
#if LV_USE_OBSERVER && LV_USE_SLIDER && LV_USE_LABEL && LV_BUILD_EXAMPLES

static lv_subject_t temperature_subject;

/**
 * A slider sends a message on value change and a label display's that value
 */
void lv_example_observer_1(void)
{
    lv_subject_init_int(&temperature_subject, 28);

    /*Create a slider in the center of the display*/
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_obj_center(slider);
    lv_slider_bind_value(slider, &temperature_subject);

    /*Create a label below the slider*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_align(label, LV_ALIGN_CENTER, 0, 30);
    lv_label_bind_text(label, &temperature_subject, "%d °C");
}

#endif
```

Gestire l'accesso e i suoi stati

```
#include "../../lv_examples.h"
#if LV_USE_OBSERVER && LV_USE_SLIDER && LV_USE_LABEL && LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

/*This is the only interface between the UI and the application*/
static lv_subject_t engine_subject;

static void app_init(void);
static void ui_init(void);

/**
 * Simple PIN login screen to start an engine.
 * The only interface between the UI and the application is a single "subject".
 */
void lv_example_observer_2(void)
{
    lv_subject_init_int(&engine_subject, 0);
    app_init();
    ui_init();
}

/*-----
 * APPLICATION
 *
 * This part contains a demo application logic.
 * It doesn't know anything about the internals of the UI
 * and uses any the 'engine_subject' as an interface.
 * -----
*/
static void engine_state_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    LV_UNUSED(observer);

    int32_t v = lv_subject_get_int(subject);
    LV_UNUSED(v);
    /*In a real application set/clear a pin here*/
    LV_LOG_USER("Engine state: %d", v);
}

static void app_init(void)
{
    lv_subject_add_observer(&engine_subject, engine_state_observer_cb, NULL);
}

/*-----
 * USER INTERFACE
 *
 * This part contains only UI related code and data.
 * In a project it would a separate file and the
 * application couldn't see its internals
 * -----
*/

typedef enum {
    LOGGED_OUT,
    LOGGED_IN,
    AUTH_FAILED,
} auth_state_t;

static lv_subject_t auth_state_subject;

static void textarea_event_cb(lv_event_t * e)
{
    lv_obj_t * ta = lv_event_get_target_obj(e);
    if(lv_strcmp(lv_textarea_get_text(ta), "hello") == 0) {
        lv_subject_set_int(&auth_state_subject, LOGGED_IN);
    }
    else {
        lv_subject_set_int(&auth_state_subject, AUTH_FAILED);
    }
}

static void info_label_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    lv_obj_t * label = (lv_obj_t *) lv_observer_get_target(observer);
    switch(lv_subject_get_int(subject)) {
        case LOGGED_IN:
            lv_label_set_text(label, "Login successful");
            break;
        case LOGGED_OUT:
```

(continues on next page)

(continua dalla pagina precedente)

```

        lv_label_set_text(label, "Logged out");
        break;
    case AUTH_FAILED:
        lv_label_set_text(label, "Login failed");
        break;
    }
}

static void log_out_click_event_cb(lv_event_t * e)
{
    LV_UNUSED(e);
    lv_subject_set_int(&auth_state_subject, LOGGED_OUT);
}

static void ui_init(void)
{
    lv_subject_init_int(&auth_state_subject, LOGGED_OUT);

    /*Create a slider in the center of the display*/
    lv_obj_t * ta = lv_textarea_create(lv_screen_active());
    lv_obj_set_pos(ta, 10, 10);
    lv_obj_set_width(ta, 200);
    lv_textarea_set_one_line(ta, true);
    lv_textarea_set_password_mode(ta, true);
    lv_textarea_set_placeholder_text(ta, "The password is: hello");
    lv_obj_add_event_cb(ta, textarea_event_cb, LV_EVENT_READY, NULL);
    lv_obj_bind_state_if_eq(ta, &auth_state_subject, LV_STATE_DISABLED, LOGGED_IN);

    lv_obj_t * kb = lv_keyboard_create(lv_screen_active());
    lv_keyboard_set_textarea(kb, ta);

    lv_obj_t * btn;
    lv_obj_t * label;

    /*Create a log out button which will be active only when logged in*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_pos(btn, 220, 10);
    lv_obj_add_event_cb(btn, log_out_click_event_cb, LV_EVENT_CLICKED, NULL);
    lv_obj_bind_state_if_not_eq(btn, &auth_state_subject, LV_STATE_DISABLED, LOGGED_IN);

    label = lv_label_create(btn);
    lv_label_set_text(label, "LOG OUT");

    /*Create a label to show info*/
    label = lv_label_create(lv_screen_active());
    lv_obj_set_pos(label, 10, 60);
    lv_subject_add_observer_obj(&auth_state_subject, info_label_observer_cb, label, NULL);

    /*Create button which will be active only when logged in*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_pos(btn, 10, 80);
    lv_obj_add_flag(btn, LV_OBJ_FLAG_CHECKABLE);
    lv_obj_bind_state_if_not_eq(btn, &auth_state_subject, LV_STATE_DISABLED, LOGGED_IN);
    lv_obj_bind_checked(btn, &engine_subject);
    label = lv_label_create(btn);
    lv_label_set_text(label, "START ENGINE");
}

#endif

```

Impostare l'ora con modalità 12/24 e AM/PM

```

#include "../../lv_examples.h"
#if LV_USE_OBSERVER && LV_USE_SLIDER && LV_USE_LABEL && LV_USE_ROLLER && LV_USE_DROPDOWN && LV_FONT_MONTserrat_30 && LV_BUILD_EXAMPLES

static lv_subject_t hour_subject;
static lv_subject_t minute_subject;
static lv_subject_t format_subject;
static lv_subject_t am_pm_subject;
static lv_subject_t time_subject;
static lv_subject_t * time_group_array_subject[] = {&hour_subject, &minute_subject, &format_subject, &am_pm_subject};
const char * hour12_options = "01\n02\n03\n04\n05\n06\n07\n08\n09\n10\n11\n12\n13\n14\n15\n16\n17\n18\n19\n20\n21\n22\n23";
const char * hour24_options =
"00\n01\n02\n03\n04\n05\n06\n07\n08\n09\n10\n11\n12\n13\n14\n15\n16\n17\n18\n19\n20\n21\n22\n23\n24\n25\n26\n27\n28\n29\n30\n31\n32\n33\n34\n35\n36\n37\n38\n39\n40\n41\n42\n43\n44\n45\n46\n47\n48\n49\n50\n51\n52\n53\n54\n55\n56\n57\n58\n59";

static void set_btn_clicked_event_cb(lv_event_t * e);
static void close_clicked_event_cb(lv_event_t * e);
static void hour_roller_options_update(lv_observer_t * observer, lv_subject_t * subject);
static void time_observer_cb(lv_observer_t * observer, lv_subject_t * subject);

typedef enum {
    TIME_FORMAT_12,
    TIME_FORMAT_24,
} time_format_t;

typedef enum {
    TIME_AM,
    TIME_PM,
} time_am_pm_t;

/**
 * Show how to handle a complex time setting with hour, minute, 12/24 hour mode, and AM/PM switch
 * In a real application the time can be displayed on multiple screens and it's not trivial
 * how and where to store the current values and how to get them.
 * In this example the widgets to set the time are created/deleted dynamically,
 * yet they always know what the current values are by using subjects.
 */
void lv_example_observer_3(void)
{
    /*Initialize the subjects.
     *The UI will update these and read the current values from here,
     *however the application can update these values at any time and
     *the widgets will be updated automatically. */
    lv_subject_init_int(&hour_subject, 7);
    lv_subject_init_int(&minute_subject, 45);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_subject_init_int(&format_subject, TIME_FORMAT_12);
lv_subject_init_int(&am_pm_subject, TIME_AM);
lv_subject_init_group(&time_subject, time_group_array_subject, 4);

/*Create the UI*/
lv_obj_t * time_label = lv_label_create(lv_screen_active());
lv_obj_set_style_text_font(time_label, &lv_font_montserrat_30, 0);
lv_subject_add_observer_obj(&time_subject, time_observer_cb, time_label, NULL);
lv_obj_set_pos(time_label, 24, 24);

lv_obj_t * set_btn = lv_button_create(lv_screen_active());
lv_obj_set_pos(set_btn, 180, 24);
lv_obj_add_event_cb(set_btn, set_btn_clicked_event_cb, LV_EVENT_CLICKED, NULL);

lv_obj_t * set_label = lv_label_create(set_btn);
lv_label_set_text(set_label, "Set");

/*Update some subjects to see if the UI is updated as well*/
lv_subject_set_int(&hour_subject, 9);
lv_subject_set_int(&minute_subject, 30);
lv_subject_set_int(&am_pm_subject, TIME_PM);
}

static void set_btn_clicked_event_cb(lv_event_t * e)
{
    lv_obj_t * set_btn = lv_event_get_target_obj(e);
    lv_obj_add_state(set_btn, LV_STATE_DISABLED);

    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, lv_pct(100), LV_SIZE_CONTENT);
    lv_obj_align(cont, LV_ALIGN_BOTTOM_MID, 0, 0);

    lv_obj_t * hour_roller = lv_roller_create(cont);
    lv_roller_add_flag(hour_roller, LV_OBJ_FLAG_FLEX_IN_NEW_TRACK);
    lv_subject_add_observer_obj(&format_subject, hour_roller_options_update, hour_roller, NULL);
    lv_roller_bind_value(hour_roller, &hour_subject);
    lv_obj_set_pos(hour_roller, 0, 0);

    lv_obj_t * min_roller = lv_roller_create(cont);
    lv_roller_set_options(min_roller, minute_options, LV_ROLLER_MODE_NORMAL);
    lv_roller_bind_value(min_roller, &minute_subject);
    lv_obj_set_pos(min_roller, 64, 0);

    lv_obj_t * format_dropdown = lv_dropdown_create(cont);
    lv_dropdown_set_options(format_dropdown, "12\nn24");
    lv_dropdown_bind_value(format_dropdown, &format_subject);
    lv_obj_set_pos(format_dropdown, 128, 0);
    lv_obj_set_width(format_dropdown, 80);

    lv_obj_t * am_pm_dropdown = lv_dropdown_create(cont);
    lv_dropdown_set_options(am_pm_dropdown, "am\npm");
    lv_dropdown_bind_value(am_pm_dropdown, &am_pm_subject);
    lv_obj_bind_state_if_eq(am_pm_dropdown, &format_subject, LV_STATE_DISABLED, TIME_FORMAT_24);
    lv_obj_set_pos(am_pm_dropdown, 128, 48);
    lv_obj_set_width(am_pm_dropdown, 80);

    lv_obj_t * close_btn = lv_button_create(cont);
    lv_obj_align(close_btn, LV_ALIGN_TOP_RIGHT, 0, 0);
    /*Pass the set_btn as user_data to make it non-disabled on close*/
    lv_obj_add_event_cb(close_btn, close_clicked_event_cb, LV_EVENT_CLICKED, set_btn);

    lv_obj_t * close_label = lv_label_create(close_btn);
    lv_label_set_text(close_label, LV_SYMBOL_CLOSE);
}

static void close_clicked_event_cb(lv_event_t * e)
{
    lv_obj_t * set_btn = (lv_obj_t *) lv_event_get_user_data(e);
    lv_obj_t * close_btn = lv_event_get_target_obj(e);
    lv_obj_t * cont = lv_obj_get_parent(close_btn);
    lv_obj_remove_state(set_btn, LV_STATE_DISABLED);
    lv_obj_delete(cont);
}

/*Watch all related subject to display the current time correctly*/
static void time_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    int32_t hour = lv_subject_get_int(lv_subject_get_group_element(subject, 0));
    int32_t minute = lv_subject_get_int(lv_subject_get_group_element(subject, 1));
    int32_t format = lv_subject_get_int(lv_subject_get_group_element(subject, 2));
    int32_t am_pm = lv_subject_get_int(lv_subject_get_group_element(subject, 3));

    lv_obj_t * label = (lv_obj_t *) lv_observer_get_target(observer);

    if(format == TIME_FORMAT_24) {
        lv_label_set_text_fmt(label, "%" LV_PRId32 ":%02" LV_PRId32, hour, minute);
    }
    else {
        lv_label_set_text_fmt(label, "%" LV_PRId32":%02" LV_PRId32" %s", hour + 1, minute, am_pm == TIME_AM ? "am" : "pm");
    }
}

/*Change the hour options on format change*/
static void hour_roller_options_update(lv_observer_t * observer, lv_subject_t * subject)
{
    lv_obj_t * roller = (lv_obj_t *) lv_observer_get_target(observer);
    int32_t prev_selected = lv_roller_get_selected(roller);
    int32_t v = lv_subject_get_int(subject);
    if(v == TIME_FORMAT_12) {
        prev_selected--;
        if(prev_selected > 12) prev_selected -= 12;
        lv_roller_set_options(roller, hour12_options, LV_ROLLER_MODE_NORMAL);
    }
    else {
        prev_selected++;
        lv_roller_set_options(roller, hour24_options, LV_ROLLER_MODE_NORMAL);
    }

    lv_roller_set_selected(roller, prev_selected, LV_ANIM_OFF);
    lv_obj_send_event(roller, LV_EVENT_VALUE_CHANGED, NULL);
}

#endif

```

Visualizzazione a schede personalizzata con gestione dello stato

```
#include "../../lv_examples.h"
#if LV_USE_OBSERVER && LV_USE_SLIDER && LV_USE_LABEL && LV_USE_ROLLER && LV_USE_DROPDOWN && LV_FONT_MONTserrat_30 && LV_BUILD_EXAMPLES

static void cont_observer_cb(lv_observer_t * observer, lv_subject_t * subject);
static void btn_create(lv_obj_t * parent, const char * text);
static void btn_click_event_cb(lv_event_t * e);
static void btn_observer_cb(lv_observer_t * observer, lv_subject_t * subject);
static void indicator_observer_cb(lv_observer_t * observer, lv_subject_t * subject);

static lv_subject_t current_tab_subject;
static lv_subject_t slider_subject[4];
static lv_subject_t dropdown_subject[3];
static lv_subject_t roller_subject[2];

void lv_example_observer_4(void)
{
    lv_subject_init_int(&current_tab_subject, 0);
    lv_subject_init_int(&slider_subject[0], 0);
    lv_subject_init_int(&slider_subject[1], 0);
    lv_subject_init_int(&slider_subject[2], 0);
    lv_subject_init_int(&slider_subject[3], 0);
    lv_subject_init_int(&dropdown_subject[0], 0);
    lv_subject_init_int(&dropdown_subject[1], 0);
    lv_subject_init_int(&dropdown_subject[2], 0);
    lv_subject_init_int(&roller_subject[0], 0);
    lv_subject_init_int(&roller_subject[1], 0);

    lv_obj_t * main_cont = lv_obj_create(lv_screen_active());
    lv_obj_remove_style_all(main_cont);
    lv_obj_set_size(main_cont, lv_pct(100), lv_pct(100));
    lv_obj_set_style_pad_all(main_cont, 0, 0);
    lv_obj_set_flex_flow(main_cont, LV_FLEX_FLOW_COLUMN);

    lv_obj_t * cont = lv_obj_create(main_cont);
    lv_obj_remove_style_all(cont);
    lv_obj_set_flex_grow(cont, 1);
    lv_obj_set_style_pad_all(cont, 8, 0);
    lv_obj_set_width(cont, lv_pct(100));
    lv_subject_add_observer_obj(&current_tab_subject, cont_observer_cb, cont, NULL);
    lv_obj_set_scroll_dir(cont, LV_DIR_VER);

    lv_obj_t * footer = lv_obj_create(main_cont);
    lv_obj_remove_style_all(footer);
    lv_obj_set_style_pad_column(footer, 8, 0);
    lv_obj_set_style_pad_all(footer, 8, 0);
    lv_obj_set_flex_flow(footer, LV_FLEX_FLOW_ROW);
    lv_obj_set_size(footer, lv_pct(100), 60);
    lv_obj_align(footer, LV_ALIGN_BOTTOM_MID, 0, 0);

    btn_create(footer, "First");
    btn_create(footer, "Second");
    btn_create(footer, "Third");

    lv_obj_t * indicator = lv_obj_create(footer);
    lv_obj_remove_style(indicator, NULL, 0);
    lv_obj_set_style_bg_opa(indicator, LV_OPA_40, 0);
    lv_subject_add_observer_obj(&current_tab_subject, indicator_observer_cb, indicator, NULL);
    lv_obj_set_height(indicator, 10);
    lv_obj_align(indicator, LV_ALIGN_BOTTOM_LEFT, 0, 0);
    lv_obj_add_flag(indicator, LV_OBJ_FLAG_IGNORE_LAYOUT);

    /*Be sure the indicator has the correct size*/
    lv_obj_update_layout(indicator);
    lv_subject_notify(&current_tab_subject);
}

static int32_t anim_get_x_cb(lv_anim_t * a)
{
    return lv_obj_get_x_aligned((lv_obj_t *) a->var);
}

static void anim_set_x_cb(void * obj, int32_t v)
{
    lv_obj_set_x((lv_obj_t *) obj, v);
}

static void cont_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    int32_t prev_v = lv_subject_get_previous_int(subject);
    int32_t cur_v = lv_subject_get_int(subject);
    lv_obj_t * cont = (lv_obj_t *) lv_observer_get_target(observer);

    /*Animate out the previous content*/
    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_duration(&a, 300);
    lv_anim_set_path_cb(&a, lv_anim_path_ease_in_out);
    lv_anim_set_exec_cb(&a, anim_set_x_cb);
    lv_anim_set_value_cb(&a, anim_get_x_cb);
    lv_anim_set_completed_cb(&a, lv_obj_delete_anim_completed_cb);

    uint32_t i;
    uint32_t delay = 0;
    uint32_t child_cnt_prev = lv_obj_get_child_count(cont);
    for(i = 0; i < child_cnt_prev; i++) {
        lv_obj_t * child = lv_obj_get_child(cont, i);
        lv_anim_set_var(&a, child);
        if(prev_v < cur_v) {
            lv_anim_set_values(&a, 0, -20);
        }
        else {
            lv_anim_set_values(&a, 0, 20);
        }
        lv_anim_set_delay(&a, delay);
        lv_anim_start(&a);
        lv_obj_fade_out(child, 200, delay);
        delay += 50;
    }

    /*Create the widgets according to the current value*/
    if(cur_v == 0) {

```

(continues on next page)

(continua dalla pagina precedente)

```

    for(i = 0; i < 4; i++) {
        lv_obj_t * slider = lv_slider_create(cont);
        lv_slider_bind_value(slider, &slider_subject[i]);
        lv_obj_align(slider, LV_ALIGN_TOP_MID, 0, 10 + i * 30);
    }
}

if(cur_v == 1) {
    for(i = 0; i < 3; i++) {
        lv_obj_t * dropdown = lv_dropdown_create(cont);
        lv_dropdown_bind_value(dropdown, &dropdown_subject[i]);
        lv_obj_align(dropdown, LV_ALIGN_TOP_MID, 0, i * 50);
    }
}

if(cur_v == 2) {
    for(i = 0; i < 2; i++) {
        lv_obj_t * roller = lv_roller_create(cont);
        lv_roller_bind_value(roller, &roller_subject[i]);
        lv_obj_align(roller, LV_ALIGN_CENTER, -80 + i * 160, 0);
    }
}

/*Animate in the new widgets*/
lv_anim_set_completed_cb(&a, NULL);
for(i = child_cnt_prev; i < lv_obj_get_child_count(cont); i++) {
    lv_obj_t * child = lv_obj_get_child(cont, i);
    lv_anim_set_var(&a, child);
    if(prev_v < cur_v) {
        lv_anim_set_values(&a, 20, 0);
    }
    else {
        lv_anim_set_values(&a, -20, -0);
    }
    lv_anim_set_delay(&a, delay);
    lv_anim_start(&a);
    lv_obj_fade_in(child, 200, delay);
    delay += 50;
}

}

static void btn_create(lv_obj_t * parent, const char * text)
{
    lv_obj_t * btn = lv_button_create(parent);
    lv_obj_set.flex_grow(btn, 1);
    lv_obj.set.height(btn, lv_pct(100));
    lv_obj.set.style.radius(btn, 0, 0);
    lv_subject_add_observer_obj(&current_tab_subject, btn_observer_cb, btn, NULL);
    lv_obj.add.event_cb(btn, btn_click_event_cb, LV_EVENT_CLICKED, NULL);

    lv_obj_t * label = lv_label_create(btn);
    lv_label.set.text(label, text);
    lv_obj_center(label);
}

static void btn_click_event_cb(lv_event_t * e)
{
    lv_obj_t * btn = lv_event_get_target_obj(e);
    uint32_t idx = lv_obj_get_index(btn);
    lv_subject.set_int(&current_tab_subject, idx);
}

static void btn_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    int32_t prev_v = lv_subject.get.previous_int(subject);
    int32_t cur_v = lv_subject.get.int(subject);

    lv_obj_t * btn = (lv_obj_t *) lv_observer_get_target(observer);
    int32_t idx = (int32_t)lv_obj.get.index(btn);

    if(idx == prev_v) lv_obj.remove.state(btn, LV_STATE_CHECKED);
    if(idx == cur_v) lv_obj.add.state(btn, LV_STATE_CHECKED);
}

static void indicator_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    int32_t cur_v = lv_subject.get.int(subject);
    lv_obj_t * indicator = (lv_obj_t *) lv_observer_get_target(observer);

    lv_obj_t * footer = lv_obj.get.parent(indicator);
    lv_obj_t * btn_act = lv_obj.get.child(footer, cur_v);
    lv_obj.set.width(indicator, lv_obj.get.width(btn_act));

    lv_anim_t a;
    lv_anim.init(&a);
    lv_anim.set_exec_cb(&a, anim_set_x_cb);
    lv_anim.set.duration(&a, 300);
    lv_anim.set.path_cb(&a, lv_anim_path.ease_in_out);
    lv_anim.set.var(&a, indicator);
    lv_anim.set.values(&a, lv_obj.get.x(indicator), lv_obj.get.x(btn_act));
    lv_anim.start(&a);
}

#endif

```

Processo di aggiornamento del firmware

```

#include "../../lv_examples.h"
#ifndef LV_USE_OBSERVER && LV_USE_ARC && LV_USE_LABEL && LV_USE_BUTTON && LV_USE_SPINNER && LV_BUILD_EXAMPLES
typedef enum {
    FW_UPDATE_STATE_IDLE,
    FW_UPDATE_STATE_CONNECTING,
    FW_UPDATE_STATE_CONNECTED,
    FW_UPDATE_STATE_DOWNLOADING,
    FW_UPDATE_STATE_CANCEL,
    FW_UPDATE_STATE_READY,
} lv_fw_update_state_t;

```

(continues on next page)

(continua dalla pagina precedente)

```

static void fw_upload_manager_observer_cb(lv_observer_t * observer, lv_subject_t * subject);
static void fw_update_btn_clicked_event_cb(lv_event_t * e);
static void fw_update_close_event_cb(lv_event_t * e);
static void fw_update_win_observer_cb(lv_observer_t * observer, lv_subject_t * subject);

static lv_subject_t fw_download_percent_subject;
static lv_subject_t fw_update_status_subject;

/***
 * Show how to handle a complete firmware update process with observers.
 * Normally it's hard to implement a firmware update process because in some cases
 * - the App needs to was for the UI (wait for a button press)
 * - the UI needs to wait for the App (connecting or downloading)
 * With observers these complex mechanisms can be implemented a simple and clean way.
 */
void lv_example_observer_5(void)
{
    lv_subject_init_int(&fw_download_percent_subject, 0);
    lv_subject_init_int(&fw_update_status_subject, FW_UPDATE_STATE_IDLE);

    lv_subject_add_observer(&fw_update_status_subject, fw_upload_manager_observer_cb, NULL);

    /*Create start FW update button*/
    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_add_event_cb(btn, fw_update_btn_clicked_event_cb, LV_EVENT_CLICKED, NULL);
    lv_obj_center(btn);
    lv_obj_t * label = lv_label_create(btn);
    lv_label_set_text(label, "Firmware update");
}

static void fw_update_btn_clicked_event_cb(lv_event_t * e)
{
    LV_UNUSED(e);
    lv_obj_t * win = lv_win_create(lv_screen_active());
    lv_obj_set_size(win, lv_pct(90), lv_pct(90));
    lv_obj_set_height(lv_win_get_header(win), 40);
    lv_obj_set_style_radius(win, 8, 0);
    lv_obj_set_style_shadow_width(win, 24, 0);
    lv_obj_set_style_shadow_offset_x(win, 2, 0);
    lv_obj_set_style_shadow_offset_y(win, 3, 0);
    lv_obj_set_style_shadow_color(win, lv_color_hex3(0x888), 0);
    lv_win_add_title(win, "Firmware update");
    lv_obj_t * btn = lv_win_add_button(win, LV_SYMBOL_CLOSE, 40);
    lv_obj_add_event_cb(btn, fw_update_close_event_cb, LV_EVENT_CLICKED, NULL);
    lv_obj_center(win);

    lv_subject_set_int(&fw_update_status_subject, FW_UPDATE_STATE_IDLE);
    lv_subject_add_observer_obj(&fw_update_status_subject, fw_update_win_observer_cb, win, NULL);
}

static void fw_update_close_event_cb(lv_event_t * e)
{
    LV_UNUSED(e);
    lv_subject_set_int(&fw_update_status_subject, FW_UPDATE_STATE_CANCEL);
}

static void restart_btn_click_event_cb(lv_event_t * e)
{
    lv_obj_t * win = (lv_obj_t *) lv_event_get_user_data(e);
    lv_obj_delete(win);
    lv_subject_set_int(&fw_update_status_subject, FW_UPDATE_STATE_IDLE);
}

static void fw_update_win_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    lv_obj_t * win = (lv_obj_t *) lv_observer_get_target(observer);
    lv_obj_t * cont = lv_win_get_content(win);
    lv_fw_update_state_t status = (lv_fw_update_state_t) lv_subject_get_int(&fw_update_status_subject);
    if(status == FW_UPDATE_STATE_IDLE) {
        lv_obj_clean(cont);
        lv_obj_t * spinner = lv_spinner_create(cont);
        lv_obj_center(spinner);
        lv_obj_set_size(spinner, 130, 130);
        lv_obj_t * label = lv_label_create(cont);
        lv_label_set_text(label, "Connecting");
        lv_obj_center(label);

        lv_subject_set_int(subject, FW_UPDATE_STATE_CONNECTING);
    }
    else if(status == FW_UPDATE_STATE_DOWNLOADING) {
        lv_obj_clean(cont);
        lv_obj_t * arc = lv_arc_create(cont);
        lv_arc_bind_value(arc, &fw_download_percent_subject);
        lv_obj_center(arc);
        lv_obj_set_size(arc, 130, 130);
        lv_obj_remove_flag(arc, LV_OBJ_FLAG_CLICKABLE);

        lv_obj_t * label = lv_label_create(cont);
        lv_label_bind_text(label, &fw_download_percent_subject, "%d %");
        lv_obj_center(label);
    }
    else if(status == FW_UPDATE_STATE_READY) {
        lv_obj_clean(cont);
        lv_obj_t * label = lv_label_create(cont);
        lv_label_set_text(label, "Firmware update is ready");
        lv_obj_align(label, LV_ALIGN_CENTER, 0, -20);

        lv_obj_t * btn = lv_button_create(cont);
        lv_obj_align(btn, LV_ALIGN_CENTER, 0, 20);
        lv_obj_add_event_cb(btn, restart_btn_click_event_cb, LV_EVENT_CLICKED, win);

        label = lv_label_create(btn);
        lv_label_set_text(label, "Restart");
    }
    else if(status == FW_UPDATE_STATE_CANCEL) {
        lv_obj_delete(win);
    }
}

static void connect_timer_cb(lv_timer_t * t)

```

(continues on next page)

(continua dalla pagina precedente)

```

{
    if(lv_subject_get_int(&fw_update_status_subject) != FW_UPDATE_STATE_CANCEL) {
        lv_subject_set_int(&fw_update_status_subject, FW_UPDATE_STATE_CONNECTED);
    }
    lv_timer_delete(t);
}

static void download_timer_cb(lv_timer_t * t)
{
    if(lv_subject_get_int(&fw_update_status_subject) == FW_UPDATE_STATE_CANCEL) {
        lv_timer_delete(t);
        return;
    }

    int32_t v = lv_subject_get_int(&fw_download_percent_subject);
    if(v < 100) {
        lv_subject_set_int(&fw_download_percent_subject, v + 1);
    }
    else {
        lv_subject_set_int(&fw_update_status_subject, FW_UPDATE_STATE_READY);
        lv_timer_delete(t);
    }
}

/**
 * Emulate connection and FW downloading by timers
 */
static void fw_upload_manager_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    LV_UNUSED(subject);
    LV_UNUSED(observer);

    lv_fw_update_state_t state = (lv_fw_update_state_t) lv_subject_get_int(&fw_update_status_subject);
    if(state == FW_UPDATE_STATE_CONNECTING) {
        lv_timer_create(connect_timer_cb, 2000, NULL);
    }
    else if(state == FW_UPDATE_STATE_CONNECTED) {
        lv_subject_set_int(&fw_download_percent_subject, 0);
        lv_subject_set_int(&fw_update_status_subject, FW_UPDATE_STATE_DOWNLOADING);
        lv_timer_create(download_timer_cb, 50, NULL);
    }
}
#endif

```

Aggiornamento dello stile modulare al cambio del tema

```

#include "../../lv_examples.h"
#if LV_USE_OBSERVER && LV_USE_SLIDER && LV_USE_LABEL && LV_BUILD_EXAMPLES

typedef enum {
    THEME_MODE_LIGHT,
    THEME_MODE_DARK,
} lv_theme_mode_t;

static lv_obj_t * my_panel_create(lv_obj_t * parent);
static lv_obj_t * my_button_create(lv_obj_t * parent, const char * text, lv_event_cb_t event_cb);
static void switch_theme_event_cb(lv_event_t * e);

static lv_subject_t theme_subject;

/***
 * Change between light and dark mode
 */
void lv_example_observer_6(void)
{
    lv_subject_init_int(&theme_subject, THEME_MODE_DARK);

    lv_obj_t * panel1 = my_panel_create(lv_screen_active());
    lv_obj_set_flex_flow(panel1, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(panel1, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);
    lv_obj_set_size(panel1, lv_pct(90), lv_pct(90));
    lv_obj_center(panel1);

    my_button_create(panel1, "Button 1", switch_theme_event_cb);
    my_button_create(panel1, "Button 2", switch_theme_event_cb);
    my_button_create(panel1, "Button 3", switch_theme_event_cb);
    my_button_create(panel1, "Button 4", switch_theme_event_cb);
    my_button_create(panel1, "Button 5", switch_theme_event_cb);
    my_button_create(panel1, "Button 6", switch_theme_event_cb);
    my_button_create(panel1, "Button 7", switch_theme_event_cb);
    my_button_create(panel1, "Button 8", switch_theme_event_cb);
    my_button_create(panel1, "Button 9", switch_theme_event_cb);
    my_button_create(panel1, "Button 10", switch_theme_event_cb);
}

static void switch_theme_event_cb(lv_event_t * e)
{
    LV_UNUSED(e);
    if(lv_subject_get_int(&theme_subject) == THEME_MODE_LIGHT) lv_subject_set_int(&theme_subject, THEME_MODE_DARK);
    else lv_subject_set_int(&theme_subject, THEME_MODE_LIGHT);
}

/*-----
 * my_panel.c
 *
 * It would be a separate file with its own
 * local types, data, and functions
-----*/
typedef struct {
    lv_style_t style_main;
    lv_style_t style_scrollbar;
} lv_panel_styles_t;

static void my_panel_style_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    LV_UNUSED(subject);

```

(continues on next page)

(continua dalla pagina precedente)

```

LV_UNUSED(observer);

lv_theme_mode_t m = (lv_theme_mode_t) lv_subject_get_int(&theme_subject);
lv_panel_styles_t * styles = (lv_panel_styles_t *) lv_observer_get_target(observer);
if(m == THEME_MODE_LIGHT) {
    lv_style_set_bg_color(&styles->style_main, lv_color_hex3(0xffff));
    lv_style_set_shadow_color(&styles->style_main, lv_color_hex3(0x888));
    lv_style_set_text_color(&styles->style_main, lv_color_hex3(0x222));
    lv_style_set_bg_color(&styles->style_scrollbar, lv_color_hex3(0x888));
}
if(m == THEME_MODE_DARK) {
    lv_style_set_bg_color(&styles->style_main, lv_color_hex(0x040038));
    lv_style_set_shadow_color(&styles->style_main, lv_color_hex3(0xaa));
    lv_style_set_text_color(&styles->style_main, lv_color_hex3(0xeeee));
    lv_style_set_bg_color(&styles->style_scrollbar, lv_color_hex3(0xaa));
}

lv_obj_report_style_change(&styles->style_main);
lv_obj_report_style_change(&styles->style_scrollbar);
}

static lv_obj_t * my_panel_create(lv_obj_t * parent)
{
    static bool init = false;
    static lv_panel_styles_t styles;
    if(!init) {
        init = true;

        lv_style_init(&styles.style_main);
        lv_style_set_radius(&styles.style_main, 12);
        lv_style_set_bg_opa(&styles.style_main, LV_OPA_COVER);
        lv_style_set_shadow_width(&styles.style_main, 24);
        lv_style_set_shadow_offset_x(&styles.style_main, 4);
        lv_style_set_shadow_offset_y(&styles.style_main, 6);
        lv_style_set_pad_all(&styles.style_main, 12);
        lv_style_set_pad_gap(&styles.style_main, 16);

        lv_style_init(&styles.style_scrollbar);
        lv_style_set_width(&styles.style_scrollbar, 4);
        lv_style_set_radius(&styles.style_scrollbar, 2);
        lv_style_set_pad_right(&styles.style_scrollbar, 8);
        lv_style_set_pad_ver(&styles.style_scrollbar, 8);
        lv_style_set_bg_opa(&styles.style_scrollbar, LV_OPA_50);

        lv_subject_add_observer_with_target(&theme_subject, my_panel_style_observer_cb, &styles, NULL);
    }

    lv_obj_t * panel = lv_obj_create(parent);
    lv_obj_remove_style_all(panel);
    lv_obj_add_style(panel, &styles.style_main, 0);
    lv_obj_add_style(panel, &styles.style_scrollbar, LV_PART_SCROLLBAR);

    return panel;
}

/*-----
 * my_button.c
 *
 * It would be a separate file with its own
 * local types, data, and functions
 *-----*/
typedef struct {
    lv_style_t style_main;
    lv_style_t style_pressed;
} lv_button_styles_t;

static void my_button_style_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    LV_UNUSED(subject);
    LV_UNUSED(observer);

    lv_theme_mode_t m = (lv_theme_mode_t) lv_subject_get_int(&theme_subject);
    lv_button_styles_t * styles = (lv_button_styles_t *) lv_observer_get_target(observer);
    if(m == THEME_MODE_LIGHT) {
        lv_style_set_bg_color(&styles->style_main, lv_color_hex(0x3379de));
        lv_style_set_bg_grad_color(&styles->style_main, lv_color_hex(0xd249a5));
        lv_style_set_shadow_color(&styles->style_main, lv_color_hex(0x3379de));
        lv_style_set_text_color(&styles->style_main, lv_color_hex3(0xffff));

        lv_style_set_color_filter_opa(&styles->style_pressed, LV_OPA_70);
    }
    if(m == THEME_MODE_DARK) {
        lv_style_set_bg_color(&styles->style_main, lv_color_hex(0xde1382));
        lv_style_set_bg_grad_color(&styles->style_main, lv_color_hex(0x4b0c72));
        lv_style_set_shadow_color(&styles->style_main, lv_color_hex(0x4b0c72));
        lv_style_set_text_color(&styles->style_main, lv_color_hex3(0xffff));

        lv_style_set_color_filter_opa(&styles->style_pressed, LV_OPA_30);
    }

    lv_obj_report_style_change(&styles->style_main);
    lv_obj_report_style_change(&styles->style_pressed);
}

static lv_obj_t * my_button_create(lv_obj_t * parent, const char * text, lv_event_cb_t event_cb)
{
    static bool init = false;
    static lv_button_styles_t styles;
    if(!init) {
        init = true;

        lv_style_init(&styles.style_main);
        lv_style_set_radius(&styles.style_main, LV_RADIUS_CIRCLE);
        lv_style_set_bg_opa(&styles.style_main, LV_OPA_COVER);
        lv_style_set_bg_grad_dir(&styles.style_main, LV_GRAD_DIR_HOR);
        lv_style_set_shadow_width(&styles.style_main, 24);
        lv_style_set_shadow_offset_y(&styles.style_main, 6);
        lv_style_set_pad_hor(&styles.style_main, 32);
        lv_style_set_pad_ver(&styles.style_main, 12);

        lv_style_init(&styles.style_pressed);
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_set_color_filter_dsc(&styles.style_pressed, &lv_color_filter_shade);
lv_subject_add_observer_with_target(&theme_subject, my_button_style_observer_cb, &styles, NULL);
}

lv_obj_t * btn = lv_button_create(parent);
lv_obj_remove_style_all(btn);
lv_obj_add_style(btn, &styles.style_main, 0);
lv_obj_add_style(btn, &styles.style_pressed, LV_STATE_PRESSED);
lv_obj_add_event_cb(btn, event_cb, LV_EVENT_CLICKED, NULL);

lv_obj_t * label = lv_label_create(btn);
lv_label_set_text(label, text);

return btn;
}

#endif

```

7.10 Disegno

7.10.1 Pipeline di Disegno

Cos'è il Disegno?

Il "drawing" [disegno] (noto anche come *rendering*) consiste nello scrivere i colori dei pixel in un buffer, dove verranno visualizzati come pixel su un pannello di display. A volte questo viene fatto copiando i colori da elementi quali le proprietà dei colori di sfondo e di primo piano. Altre volte comporta il calcolo di quei colori prima che vengano scritti (ad esempio, combinandoli con altri colori quando un oggetto più in alto sull'asse Z ha un'opacità parziale).

Le sezioni seguenti illustrano la logica di disegno LVGL e come utilizzarla e, facoltativamente, adattarla al proprio progetto specifico (ad esempio, se si dispone di una GPU o di altre risorse da utilizzare).

Panoramica della Pipeline di Disegno

Sull'hardware di elaborazione moderno, progettato per essere utilizzato con schermi di grandi dimensioni, a volte sono disponibili diverse opzioni per eseguire il disegno. Ad esempio, alcuni MCU sono dotati di hardware molto valido (e veloce) per determinati tipi di attività di disegno. In alternativa, è possibile accedere a una libreria che esegue determinati tipi di attività di disegno con grande efficienza. Per consentire l'utilizzo di tali funzionalità nel modo più efficiente, LVGL v9 e versioni successive implementano una *Pipeline di Disegno*, simile a una catena di montaggio, in cui vengono prese decisioni su quali attività di disegno (*Draw Task*) assegnare a quali "entità logiche" (*Draw Unit*) per essere eseguite.

Questa Pipeline è progettata per essere flessibile ed estensibile. È possibile utilizzarla per eseguire rendering personalizzati con una GPU o sostituire parti della logica di rendering software integrata nella misura desiderata.

Utilizzando gli eventi, è anche possibile modificare le *Draw Task* o inserirne di nuove come Widget di rendering LVGL.

Le sezioni seguenti descrivono la terminologia e i concetti di base del rendering.

Draw Task

Un "Draw Task" (`lv_draw_task_t`) è un pacchetto di informazioni che viene creato all'inizio della Pipeline di Disegno quando viene effettuata una richiesta di disegno. Funzioni come `lv_draw_rect()` e `lv_draw_label()` creano una o più Draw Task e le passano lungo la Pipeline di Disegno. Ciascun Draw Task contiene tutte le informazioni necessarie per:

- calcolare quale *Draw Unit* deve ricevere questo task, e
- fornire alla Draw Unit tutte le informazioni necessarie per completare il task di disegno.

Un Draw Task contiene le seguenti informazioni:

type

definisce l'algoritmo di disegno utilizzato (ad esempio, linea, riempimento, bordo, immagine, etichetta, arco, triangolo, ecc.)

area

definisce il rettangolo in cui verrà eseguito il disegno

transformation matrix

[matrice di trasformazione] Se `LV_DRAW_TRANSFORM_USE_MATRIX` è configurato a '1'

state

in attesa, in coda, in corso, completato

drawing descriptor

[descrittore del disegno] contiene i dettagli del disegno da eseguire

preferred Draw Unit ID

identificatore della Draw Unit che deve eseguire questa attività

preference score

valore che descrive la velocità della Draw Unit specificata rispetto al rendering software (maggiori dettagli di seguito)

next

un link al Draw Task successivo nell'elenco.

I Draw Task vengono raccolti in un elenco e periodicamente inviati alle Draw Unit.

Draw Unit

Una "Draw Unit" (basata su `lv_draw_unit_t`) è qualsiasi "entità logica" in grado di generare l'output richiesto da un *Draw Task*. Può trattarsi di un core della CPU, una GPU, una libreria di rendering personalizzata per Draw Task specifici o qualsiasi entità in grado di eseguire il rendering.

Per un'implementazione di riferimento di una draw unit, vedere `lv_draw_sw.c`.

Durante l'inizializzazione di LVGL (`lv_init()`), viene creato un elenco di Draw Unit. Se `LV_USE_DRAW_SW` è impostato a 1 in `lv_conf.h` (il default), la Drawing Unit Software si inserisce in cima a tale elenco. Se la piattaforma dispone di altre unità di disegno, se sono configurate per essere utilizzate in `lv_conf.h`, vengono aggiunte a questo elenco durante l'inizializzazione di LVGL. Se si aggiungono le proprie Draw Unit, è possibile aggiungere ciascuna unità di disegno disponibile a tale elenco chiamando `lv_draw_create_unit(sizeof(your_draw_unit_t))`. A ogni chiamata a questa funzione, la draw unit appena creata viene aggiunta in cima a tale elenco, spostando le draw unit già esistenti più indietro nell'elenco, e le Draw Unit create in precedenza ancora più indietro. L'ordine di questo elenco (e quindi l'ordine in cui viene eseguita la *Valutazione delle Draw Task*) è regolato dall'ordine in cui viene creata ciascuna Draw Unit.

La creazione di questo elenco (e l'inizializzazione delle Draw Unit) viene normalmente gestita automaticamente configurando le Draw Unit disponibili in `lv_conf.h`, ad esempio impostando `LV_USE_DRAW_OPENGL` o `LV_USE_PXP` o `LV_USE_DRAW SDL` o `LV_USE_DRAW_VG_LITE` a 1. Tuttavia, se si introducono Draw Unit personalizzate, sarà necessario crearle e inizializzarle (dopo `lv_init()`) come sopra. Questo includerà diverse cose, ma l'impostazione delle callback `evaluate_cb` e `dispatch_cb` (menzionate più avanti) sono due di queste.

Per un esempio di come vengono eseguite la creazione e l'inizializzazione delle unità di disegno, vedere `lv_draw_sw_init()` in `lv_draw_sw.c` o le altre unità di disegno le cui funzioni `init` sono facoltativamente chiamate in `lv_init()`.

Priorità dei Thread

Tutte le draw unit operano con una priorità dei thread configurabile, impostabile utilizzando l'opzione di configurazione `LV_DRAW_THREAD_PRIO` in `lv_conf.h`. Ciò consente di ottimizzare il livello di priorità per tutte le drawing unit, il che è particolarmente utile per i sistemi con livelli di priorità limitati.

Per default, le draw unit utilizzano `LV_THREAD_PRIO_HIGH` come priorità dei thread. Questo approccio coerente garantisce che tutte le drawing unit (rendering software, acceleratori hardware come STM32 DMA2D, NXP VGLite, ecc.) utilizzino lo stesso livello di priorità, a meno che non sia configurato diversamente in modo esplicito.

Valutazione delle Draw Task

Quando viene creata ciascuna *Draw Task*, ogni Draw Unit viene "consultata" in merito alla sua "appropriatezza" per il task. Ciò avviene tramite un puntatore a funzione di "callback di valutazione" (nota anche come `evaluate_cb`), che ogni Draw Unit imposta (per sé stessa) durante la sua inizializzazione. Normalmente, tale valutazione:

- esamina facoltativamente il "punteggio di preferenza" esistente per il task sopra menzionato,
- se riesce a svolgere quel tipo di task (ad esempio, disegnare una linea) più velocemente di altre Draw Unit che hanno già segnalato, scrive il proprio "punteggio di preferenza" e il proprio "preferred Draw Unit ID" nei rispettivi campi nel task.

In questo modo, al termine della sequenza di valutazione, il task conterrà il punteggio e l'ID della Drawing Unit che verrà utilizzata per eseguire tale attività quando verrà *dispatched [invia]*.

Questa logica, ovviamente, può essere sovrascritta o ridefinita, a seconda della progettazione del sistema.

Come effetto collaterale, questo garantisce anche che la stessa Draw Unit venga selezionata in modo coerente, a seconda del tipo (e della natura) del task di disegno, evitando qualsiasi possibile "jitter" [tremolio] dello schermo nel caso in cui più di una Draw Unit sia in grado di eseguire un determinato tipo di attività.

La sequenza dell'elenco delle Draw Unit (con la Draw Unit Software alla fine) garantisce inoltre che la Draw Unit Software sia quella "responsabile": se nessun'altra Draw Unit ha segnalato di essere migliore in una determinata attività di disegno, allora la Draw Unit Software la gestirà.

Dispatching

Durante la raccolta delle Draw Task, LVGL le invia frequentemente alle Draw Unit assegnate. Questo viene gestito tramite `dispatch_cb` delle Draw Unit.

Se una Draw Unit è impegnata con un altro Draw Task, ritorna semplicemente. Tuttavia, se è disponibile, può accettare una Draw Task.

`lv_draw_get_next_available_task(layer, previous_task, draw_unit_id)` è un'utile funzione di supporto utilizzata da `dispatch_cb` per ottenere la successiva Draw Task su cui deve agire. Se ha gestito il task, si imposta il campo `state` del Draw Task su `LV_DRAW_TASK_STATE_FINISHED`. "Disponibile" in questo contesto significa che l'attività è stata messa in coda e assegnata a una determinata Draw Unit ed è pronta per essere eseguita. A questo scopo, vengono prese in considerazione le implicazioni derivanti dalla presenza di più thread di disegno.

Gerarchia degli Oggetti a Run-Time

Tutti gli elementi sopra elencati hanno questa relazione in fase di esecuzione:

- LVGL (globale)
 - elenco di *Draw Unit*
 - elenco di *Display*
 - * Layer: Ogni *oggetto Display* ha il proprio elenco di *Layer di Disegno*
 - Draw Task: Ogni layer ha il proprio elenco di *Draw Task*

API

[*lv_draw.h*](#)
[*lv_draw_label.h*](#)
[*lv_draw_private.h*](#)
[*lv_draw_rect.h*](#)
[*lv_draw_sw.h*](#)
[*lv_conf_internal.h*](#)

7.10.2 API di Disegno

Dove Utilizzare l'API di Disegno

Nella maggior parte dei casi si utilizza l'API di Disegno di LVGL tramite quella dei Widget: creando pulsanti, etichette, ecc. e impostandone stili, posizioni e altre proprietà. In questi casi, il rendering (disegno) viene gestito internamente e non viene visualizzata la [Pipeline di Disegno](#).

Tuttavia, ci sono tre punti in cui è possibile utilizzare direttamente l'API di Disegno di LVGL.

1. **Negli eventi di disegno dei Widget:** Ci sono codici evento che vengono inviati quando il Widget deve eseguire il rendering:

- `LV_EVENT_DRAW_MAIN_BEGIN`, `LV_EVENT_DRAW_MAIN`, `LV_EVENT_DRAW_MAIN_END`: Attivati rispettivamente prima, durante e dopo il disegno di un Widget. Il rendering dei Widget avviene in genere in `LV_EVENT_DRAW_MAIN`.
- `LV_EVENT_DRAW_POST_BEGIN`, `LV_EVENT_DRAW_POST`, `LV_EVENT_DRAW_POST_END`: Vengono attivati rispettivamente prima, durante e dopo il rendering di tutti i Widget figli. Questo può essere utile per disegni di tipo overlay, come le barre di scorrimento che devono essere visualizzate sopra gli elementi figli.

Questi sono rilevanti se viene implementato un nuovo Widget che utilizza un disegno personalizzato.

2. **Modifica dei task di disegno creati:** Quando viene creato un task di disegno per un Widget, viene inviato `LV_EVENT_DRAW_TASK_ADDED`. In questo caso, il task di disegno creato può essere modificato o è possibile aggiungerne di nuovi. Casi d'uso tipici sono la modifica di ogni barra di un grafico a barre o delle celle di una tabella.

Per motivi di prestazioni, questo evento è disabilitato per default. Abilitarlo impostando il flag `LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS` sul/sui Widget che debbano emettere questo evento.

3. **Disegna sul Widget Canvas:** Le funzioni di disegno possono essere utilizzate direttamente per disegnare su un Widget Canvas. In questo modo, il disegno personalizzato viene renderizzato in un buffer utilizzabile in seguito come immagine o maschera.

Per ulteriori informazioni, consultare [Canvas \(lv_canvas\)](#).

API di Disegno

I componenti principali dell'API di Disegno di LVGL sono le funzioni `lv_draw_rect()`, `lv_draw_label()`, `lv_draw_image()` e simili. Quando vengono chiamate, gli oggetti `lv_draw_task_t` vengono creati internamente.

Queste funzioni hanno i seguenti parametri:

- **Layer:** Questo è il target del disegno. Vedere i dettagli su [Layer di Disegno](#).
- **Draw Descriptor:** Questa è una struct di grandi dimensioni contenente tutte le informazioni sul disegno. Vedere i dettagli dei [Descrittori di Disegno](#).
- **Area** (in alcuni casi): Specifica dove disegnare.

Sistema di Coordinate

Alcune funzioni e descrittori di disegno richiedono parametri di area o punto. Queste sono sempre **coordinate assolute** sul display. Ad esempio, se il layer target si trova su un display 800x480 e l'area del layer è (100,100) (200,200), per visualizzare un oggetto 10x10 al centro, è necessario utilizzare le coordinate (145,145) (154,154) (non (40,40) (49,49)).

Eccezione: per il Widget Canvas si presume che il layer si trovi sempre alla coordinata (0,0), indipendentemente dalla posizione del Widget Canvas.

API

`lv_draw_arc.h`
`lv_draw_image.h`
`lv_draw_label.h`
`lv_draw_line.h`
`lv_draw_mask.h`
`lv_draw_triangle.h`

7.10.3 Layer di Disegno

Da non confondere con i *4 layer principali del display*, un *Draw Layer* è un buffer creato durante il rendering, necessario per determinate proprietà di stile, in modo che diversi set di pixel vengano combinati correttamente. I fattori che richiedono tali layer sono:

- opacità parziale
- applicazione di una maschera di bit
- modalità blend
- angoli ritagliati (applicazione di una maschera di bit)
- trasformazioni
 - scala
 - inclinazione
 - rotazione

Successivamente, quel layer verrà unito allo schermo o al suo layer padre nel punto corretto della sequenza di rendering.

Tipi di Layer

Layer Semplice

Le seguenti proprietà di stile attivano la creazione di un "Layer Semplice":

- `opa_layered`
- `bitmap_mask_src`
- `blend_mode`

In questo caso, il Widget verrà suddiviso in blocchi di dimensioni `LV_DRAW_SW_LAYER_SIMPLE_BUF_SIZE`.

Se non c'è memoria per un nuovo blocco, LVGL tenterà di allocare il layer dopo che un altro blocco sarà stato renderizzato e liberato.

Layer di Trasformazione

Le seguenti proprietà di stile attivano la creazione di un "Layer di Trasformazione":

- `transform_scale_x`
- `transform_scale_y`
- `transform_skew_x`
- `transform_skew_y`
- `transform_rotate`

Data la natura delle trasformazioni, il Widget da trasformare (e i suoi elementi secondari) deve essere renderizzato per primo, seguito dalla fase di trasformazione. Ciò richiede un'area di disegno temporanea (layer), spesso più grande del Widget vero e proprio, per fornire un'area di dimensioni adeguate alla trasformazione. LVGL cerca di renderizzare l'area più piccola possibile del widget, ma a causa della natura delle trasformazioni, in questo caso non è possibile effettuare alcuna suddivisione.

Clipping degli Angoli del Layer

La proprietà di stile `clip_corner` fa sì che gli angoli del Widget padre ritaglino gli angoli dei Widget figlio, i cui angoli altrimenti supererebbero un angolo arrotondato del genitore.

Esempio:

```
lv_obj_t * clipper_obj = lv_obj_create(lv_screen_active());
lv_obj_set_style_bg_color(clipper_obj, lv_palette_main(LV_PALETTE_GREY), 0);
lv_obj_center(clipper_obj);
lv_obj_set_style_pad_all(clipper_obj, 0, 0);
lv_obj_set_style_radius(clipper_obj, 40, 0);

lv_obj_t * clipped_obj = lv_obj_create(clipper_obj);
lv_obj_set_style_bg_color(clipped_obj, lv_palette_main(LV_PALETTE_RED), 0);
lv_obj_set_size(clipped_obj, 80, 80);
```

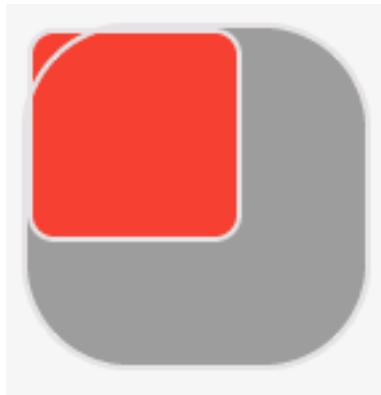


Fig. 1: Senza lo Stile "Clip Corner"

Si può notare che gli angoli del genitore non ritagliano i figli. Ma se si aggiunge questa riga, si otterrà:

```
lv_obj_set_style_clip_corner(clipper_obj, true, 0);
```

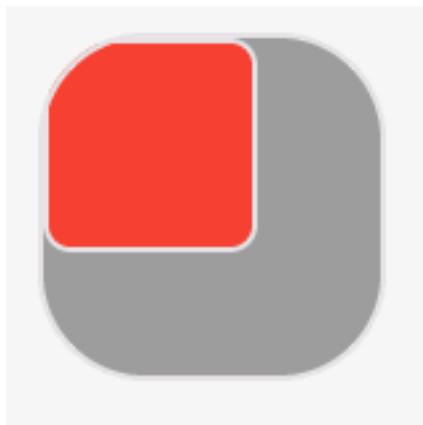


Fig. 2: Risultato dello Stile "Clip Corner"

Questa operazione comporta un consumo temporaneo di RAM. Con `LV_USE_LAYER_DEBUG` attivo...

...si può notare che per il clipping dei figli, è necessario allocare temporaneamente due aree di lavoro intermedie, ciascuna con un'altezza pari al valore di "radius-style" attualmente assegnato al genitore. Queste aree di lavoro vengono restituite all'heap al termine del rendering.

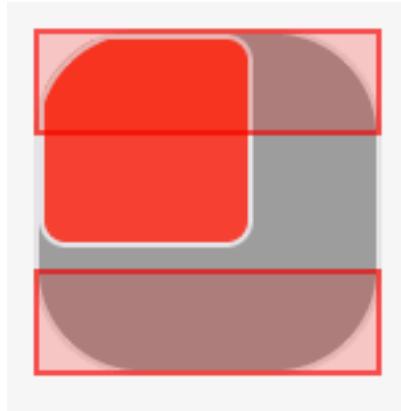


Fig. 3: Layer Creati per Implementare lo Stile "Clip Corner" Durante il Rendering

Ottenere il Layer Corrente

Il primo parametro delle funzioni `lv_draw_rect/label/etc` è un layer.

Nella maggior parte dei casi non viene creato un layer, ma ne viene utilizzato uno esistente per disegnarlo.

In questi casi è possibile utilizzare l'API di disegno e il layer corrente può essere utilizzato in modo diverso:

- Negli eventi di disegno:** Negli eventi `LV_EVENT_DRAW_MAIN/POST_BEGIN/...`, il widget viene renderizzato su un layer del display o su un altro layer temporaneo creato in precedenza durante il rendering. Il layer target corrente può essere recuperato utilizzando `lv_event_get_layer(e)`.

È anche possibile creare nuovi layer in questi eventi, ma è necessario anche quello precedente poiché sarà il layer padre in `lv_draw_layer()`.

- Modifica dei Task di Disegno creati:** In `LV_EVENT_DRAW_TASK_ADDED` i task di disegno creati da `lv_draw_rect/label/etc` possono essere modificati. Non è necessario conoscere il layer corrente per modificare un task di disegno. Tuttavia, se è necessario disegnare qualcosa di nuovo con `lv_draw_rect/label/etc`, è necessario anche il layer corrente.

Il layer corrente può essere letto dal descrittore del disegno base. Ad esempio:

```
/* In LV_EVENT_DRAW_TASK_ADDED */
lv_task_t * t = lv_event_get_draw_task(e);
lv_draw_base_dsc_t * draw_dsc = lv_draw_task_get_draw_dsc(t);

lv_layer_t * current_layer = draw_dsc.layer;
```

- Disegnare sul Widget Canvas:** Il canvas in sé non memorizza un layer, ma è possibile crearne e utilizzarne uno facilmente in questo modo:

```
/* Initialize a layer */
lv_layer_t layer;
lv_canvas_init_layer(canvas, &layer);

/* Draw something here */

/* Wait until the rendering is ready */
lv_canvas_finish_layer(canvas, &layer);
```

Creazione di un Nuovo Layer

Per creare un nuovo layer, si usa `lv_draw_layer_create()`:

```
lv_area_t layer_area = {10, 10, 80, 50}; /* Area of the new layer */
lv_layer_t * new_layer = lv_draw_layer_create(parent_layer, LV_COLOR_FORMAT_RGB565, &layer_area);
```

Una volta creato il layer, è possibile aggiungervi task di disegno utilizzando [API di Disegno](#) e [Descrittori di disegno](#). Nella maggior parte dei casi, ciò significa chiamare le funzioni `lv_draw_rect/label/etc`.

Infine, il layer deve essere renderizzato sul suo layer padre. Poiché un layer si comporta in modo simile a un'immagine, può essere renderizzato allo stesso modo delle immagini:

```
lv_draw_image_dsc_t image_draw_dsc;
lv_draw_image_dsc_init(&image_draw_dsc);
image_draw_dsc.src = new_layer; /* Source image is the new layer. */
/* Draw new layer to parent layer. */
lv_draw_layer(parent_layer, &image_draw_dsc, &layer_area);
```

Considerazioni sulla Memoria

Buffer dei Layer

Il buffer per un layer (dove avviene il rendering) non viene allocato al momento della creazione. Al contrario, viene allocato da [Draw Unit](#) quando viene eseguito il primo [Task di Disegno](#).

I buffer dei layer possono essere di grandi dimensioni, quindi assicurarsi che ci sia sufficiente memoria heap o aumentare [LV_MEM_SIZE](#) in `lv_conf.h`.

Requisiti di Memoria per i Tipi di Layer

Per risparmiare memoria, LVGL può eseguire il rendering di alcuni tipi di layer in blocchi più piccoli:

- Layer semplici:** I layer semplici possono essere renderizzati in blocchi. Ad esempio, con `opa_layered = 140`, è possibile eseguire il rendering di solo 10 righe del layer alla volta, poi delle 10 righe successive e così via. Questo evita di allocare un buffer di grandi dimensioni per l'intero layer. La dimensione del buffer per un blocco viene impostata tramite [LV_DRAW_LAYER_SIMPLE_BUF_SIZE](#) in `lv_conf.h`.
- Layer di Trasformazione:** I Widget trasformati non possono essere renderizzati in blocchi perché le trasformazioni spesso interessano pixel esterni all'area specificata. Per tali layer, LVGL alloca un buffer sufficientemente grande da renderizzare l'intera area trasformata senza limiti.

Limite di Memoria per i Layer

La memoria totale disponibile per i layer contemporaneamente è controllata da [LV_DRAW_LAYER_MAX_MEMORY](#) in `lv_conf.h`. Se impostato a 0, non c'è limite.

API

`lv_obj_event.h`

`lv_draw.h`

7.10.4 Descrittori di Disegno

Panoramica

Ogni tipo di [Task di Disegno](#) ha il proprio tipo di descrittore di disegno. Ad esempio, `lv_draw_label_dsc_t` viene utilizzato per il disegno di etichette, `lv_draw_image_dsc_t` viene utilizzato per il disegno di immagini.

Quando viene chiamata una funzione `lv_draw_...`, questa crea un Task di Disegno, copia il descrittore di disegno in un blocco di memoria da `malloc`, e lo libera automaticamente quando necessario. Pertanto, le variabili locali del descrittore di disegno possono essere utilizzate in modo sicuro.

Relazione con gli Stili

Nella maggior parte dei casi, le proprietà di stile vengono mappate 1 a 1 ai campi dei descrittori di disegno. Ad esempio:

- `label_dsc.color` corrisponde alla proprietà di stile `text_color`.
- `shadow_dsc.width`, `line_dsc.opa` e `arc_dsc.width` vengono mappati a `shadow_width`, `line_opa` e `arc_width` negli stili.

Consultare [Proprietà di Stile](#) per vedere l'elenco delle proprietà di stile e il loro significato.

Descrittore di Disegno di Base

In ogni descrittore di disegno è presente un "descrittore di base" generico con tipo `lv_draw_dsc_base_t` e con base nel nome. Ad esempio `label_dsc.base`. Questa struct memorizza informazioni utili su quale Widget e parte hanno creato il descrittore di disegno. Vedere tutti i campi in `lv_draw_dsc_base_t`.

In un evento `LV_EVENT_DRAW_TASK_ADDED`, gli elementi del descrittore di base sono molto utili per identificare il Task di Disegno. Ad esempio:

```
/* In LV_EVENT_DRAW_TASK_ADDED */
lv_draw_task_t * t = lv_event_get_draw_task(e);
lv_draw_base_dsc_t * draw_dsc = lv_draw_task_get_draw_dsc(t);
draw_dsc.obj; /* The Widget for which the draw descriptor was created */
draw_dsc.part; /* The Widget part for which the draw descriptor was created
E.g. LV_PART_INDICATOR */
draw_dsc.id1; /* A Widget type specific ID (e.g. table row index).
See the docs of the given Widget. */
draw_dsc.id2;

draw_dsc.layer; /* The target layer.
Required when a new Draw Tasks are also created */
```

Inizializzazione Semplice

Prima di utilizzare un descrittore di disegno, è necessario inizializzarlo con la funzione correlata. Ad esempio, `lv_draw_label_dsc_init(&my_label_draw_dsc)`.

Dopo l'inizializzazione, è possibile impostare ogni campo del descrittore di disegno. I valori di default sono abbastanza sensati e ragionevoli, quindi in genere solo pochi campi necessitano di modifiche. Ad esempio:

```
/* In LV_EVENT_DRAW_MAIN */
lv_draw_label_dsc_t my_label_draw_dsc;
lv_draw_label_dsc_init(&my_label_draw_dsc);
my_label_draw_dsc.font = &my_font;
my_label_draw_dsc.color = lv_color_hex(0xff0000);
my_label_draw_dsc.text = "Hello";

lv_area_t a = {10, 10, 200, 50}; /* Draw label here */

lv_draw_label(lv_event_get_layer(e), &my_label_draw_dsc, &a);
```

Inizializzazione per i Widget

Durante il rendering di una parte di un Widget, le funzioni di supporto possono inizializzare i descrittori di disegno in base agli stili e a una parte specifica del Widget nello stato corrente.

Ad esempio:

```
/* In LV_EVENT_DRAW_MAIN */
lv_draw_rect_dsc_t cur_dsc;
lv_draw_rect_dsc_init(&cur_dsc);
lv_obj_init_draw_rect_dsc(obj, LV_PART_CURSOR, &cur_dsc);
cur_dsc.fill_color = lv_color_hex(0xffff00); /* Modify if needed */
lv_draw_rect(layer, &cur_dsc, &area);
```

Le funzioni `lv_obj_init_draw_...` inizializzano automaticamente i campi del descrittore di base.

Modifica dei descrittori di disegno

In `LV_EVENT_DRAW_TASK_ADDED`, è possibile accedere al descrittore di disegno di `draw_task` (utilizzando `lv_draw_task_get_label_dsc()` e funzioni simili) e modificarlo (per cambiare colore, testo, font, ecc.). Ciò significa che in `LV_EVENT_DRAW_TASK_ADDED`, i `draw_task` e i descrittori di disegno sono già inizializzati ed è sufficiente modificare solo alcuni valori specifici.

Ad esempio:

```
/* In LV_EVENT_DRAW_TASK_ADDED */
lv_draw_task_t * t = lv_event_get_draw_task(e);
lv_draw_label_dsc_t * draw_dsc = lv_draw_task_get_label_dsc(t);

/* Check a few things in 'draw_dsc->base' */

/* Make the color lighter for longer texts */
draw_dsc->color = lv_color_lighten(draw_dsc->color,
LV_MN(lv_strlen(draw_dsc->text)) * 5, 255);

/* Create new Draw Tasks if needed by calling
* `lv_draw_...(draw_dsc->base.layer, ...)` functions */
```

Descrittore di Disegno del Rettangolo

`lv_draw_rect_dsc_t` è un descrittore di supporto che combina:

- Riempimento
- Border [bordatura]
- Contorno (un bordo con i propri stili)
- Shadow
- Immagine di sfondo (un'immagine con i propri stili)

in un'unica chiamata.

`lv_obj_init_draw_rect_dsc(obj, part, &dsc);` inizializza un descrittore di disegno da un Widget e `lv_draw_rect(layer, &dsc, area)` disegna il rettangolo in un'area specificata.

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw a rectangle to the canvas
 */
void lv_example_canvas_3(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);

    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_rect_dsc_t dsc;
    lv_draw_rect_dsc_init(&dsc);
    dsc.bg_color = lv_palette_main(LV_PALETTE_RED);
    dsc.border_color = lv_palette_main(LV_PALETTE_BLUE);
    dsc.border_width = 3;
    dsc.outline_color = lv_palette_main(LV_PALETTE_GREEN);
    dsc.outline_width = 2;
    dsc.outline_pad = 2;
    dsc.outline_opa = LV_OPA_50;
    dsc.radius = 5;
    dsc.border_width = 3;

    lv_area_t coords = {10, 10, 40, 30};

    lv_draw_rect(&layer, &dsc, &coords);
    lv_canvas_finish_layer(canvas, &layer);
}
#endif
```

Descrittore di Disegno del Riempimento

I campi principali di `lv_draw_fill_dsc_t` sono semplici. Ha raggio, opacità e colore per disegnare un rettangolo. Se l'opacità è 0, non verrà creata alcun task di disegno.

- `lv_draw_fill_dsc_init(&dsc)` inizializza un Task di Disegno di riempimento.
- `lv_draw_sw_fill(layer, &dsc, area)` crea un Task di Disegno per riempire un'area.
- `lv_draw_task_get_fill_dsc(draw_task)` recupera il descrittore di riempimento da un Task di Disegno.

Gradienti

Il campo `grad` del descrittore di riempimento (o `lv_grad_dsc_t` in generale) supporta i tipi:

- Orizzontale
- Verticale
- Inclinato
- Radiale
- Conico

di gradienti.

Di seguito sono riportati alcuni esempi di gradienti.

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the background style properties
 */
void lv_example_style_2(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_radius(&style, 5);

    /*Make a gradient*/
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    static lv_grad_dsc_t grad;
    grad.dir = LV_GRAD_DIR_VER;
    grad.stops_count = 2;
    grad.stops[0].color = lv_palette_lighten(LV_PALETTE_GREY, 1);
    grad.stops[0].opa = LV_OPA_COVER;
    grad.stops[1].color = lv_palette_main(LV_PALETTE_BLUE);
    grad.stops[1].opa = LV_OPA_COVER;

    /*Shift the gradient to the bottom*/
    grad.stops[0].frac = 128;
    grad.stops[1].frac = 192;

    lv_style_set_bg_grad(&style, &grad);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

/***
 * Simulate metallic knob using conical gradient
 * For best effect set LV_GRADIENT_MAX_STOPS to 8 or at least 3
 */
void lv_example_style_16(void)
{
#if LV_GRADIENT_MAX_STOPS >= 8
    static const lv_color_t grad_colors[8] = {
        LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
        LV_COLOR_MAKE(0xff, 0xff, 0xff),
        LV_COLOR_MAKE(0xfa, 0xfa, 0xfa),
        LV_COLOR_MAKE(0x79, 0x79, 0x79),
        LV_COLOR_MAKE(0x48, 0x48, 0x48),
        LV_COLOR_MAKE(0x4b, 0x4b, 0x4b),
        LV_COLOR_MAKE(0x70, 0x70, 0x70),
        LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
    };
#elif LV_GRADIENT_MAX_STOPS >= 3
    static const lv_color_t grad_colors[3] = {
        LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
        LV_COLOR_MAKE(0xff, 0xff, 0xff),
        LV_COLOR_MAKE(0x79, 0x79, 0x79),
    };
#else
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
        LV_COLOR_MAKE(0x79, 0x79, 0x79),
    };
#endif

/*Create a style with gradient background and shadow*/
static lv_style_t style;
lv_style_init(&style);
lv_style_set_radius(&style, 500);
lv_style_set_bg_opa(&style, LV_OPA_COVER);
lv_style_set_shadow_color(&style, lv_color_black());
lv_style_set_shadow_width(&style, 50);
lv_style_set_shadow_offset_x(&style, 20);
lv_style_set_shadow_offset_y(&style, 20);
lv_style_set_shadow_opa(&style, LV_OPA_50);

/*First define a color gradient. In this example we use a gray color map with random values.*/
static lv_grad_dsc_t grad;

lv_grad_init_stops(&grad, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));

/*Make a conical gradient with the center in the middle of the object*/
#if LV_GRADIENT_MAX_STOPS >= 8
    lv_grad_conical_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, 0, 120, LV_GRAD_EXTEND_REFLECT);
#elif LV_GRADIENT_MAX_STOPS >= 3
    lv_grad_conical_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, 45, 125, LV_GRAD_EXTEND_REFLECT);
#else
    lv_grad_conical_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, 45, 110, LV_GRAD_EXTEND_REFLECT);
#endif

/*Set gradient as background*/
lv_style_set_bg_grad(&style, &grad);

/*Create an object with the new style*/
lv_obj_t * obj = lv_obj_create(lv_screen_active());
lv_obj_add_style(obj, &style, 0);
lv_obj_set_size(obj, 200, 200);
lv_obj_center(obj);
}

#endif
```

(continues on next page)

(continua dalla pagina precedente)

```
void lv_example_style_16(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, LV_PCT(80));
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS is not enabled");
    lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/

#endif /*LV_BUILD_EXAMPLES*/
```

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

/**
 * Using radial gradient as background
 */
void lv_example_style_17(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0x9B, 0x18, 0x42),
        LV_COLOR_MAKE(0x00, 0x00, 0x00),
    };

    int32_t width = lv_display_get_horizontal_resolution(NULL);
    int32_t height = lv_display_get_vertical_resolution(NULL);

    static lv_style_t style;
    lv_style_init(&style);

    /*First define a color gradient. In this example we use a purple to black color map.*/
    static lv_grad_desc_t grad;

    lv_grad_init_stops(&grad, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));

    /*Make a radial gradient with the center in the middle of the object, extending to the farthest corner*/
    lv_grad_radial_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, LV_GRAD_RIGHT, LV_GRAD_BOTTOM, LV_GRAD_EXTEND_PAD);

    /*Set gradient as background*/
    lv_style_set_bg_grad(&style, &grad);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_set_size(obj, width, height);
    lv_obj_center(obj);
}

#else

void lv_example_style_17(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, LV_PCT(80));
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS is not enabled");
    lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/

#endif /*LV_BUILD_EXAMPLES*/
```

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

/**
 * Using various gradients for button background
 */
void lv_example_style_18(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0x26, 0xa0, 0xda),
        LV_COLOR_MAKE(0x31, 0x47, 0x55),
    };

    /*Create a linear gradient going from the top left corner to the bottom at an angle, with reflected color map*/
    static lv_style_t style_with_linear_gradient_bg;
    static lv_grad_desc_t linear_gradient_desc; /*NOTE: the gradient descriptor must be static or global variable!*/

    lv_style_init(&style_with_linear_gradient_bg);
    lv_grad_init_stops(&linear_gradient_desc, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));
    lv_grad_linear_init(&linear_gradient_desc, lv_pct(0), lv_pct(0), lv_pct(20), lv_pct(100), LV_GRAD_EXTEND_REFLECT);
    lv_style_set_bg_grad(&style_with_linear_gradient_bg, &linear_gradient_desc);
    lv_style_set_bg_opa(&style_with_linear_gradient_bg, LV_OPA_COVER);

    /*Create a radial gradient with the center in the top left 1/3rd of the object, extending to the bottom right corner, with reflected color map*/
    static lv_style_t style_with_radial_gradient_bg;
    static lv_grad_desc_t radial_gradient_desc; /*NOTE: the gradient descriptor must be static or global variable!*/

    lv_style_init(&style_with_radial_gradient_bg);
    lv_grad_init_stops(&radial_gradient_desc, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));
    lv_grad_radial_init(&radial_gradient_desc, lv_pct(30), lv_pct(30), lv_pct(100), lv_pct(100), LV_GRAD_EXTEND_REFLECT);
    lv_style_set_bg_grad(&style_with_radial_gradient_bg, &radial_gradient_desc);
    lv_style_set_bg_opa(&style_with_radial_gradient_bg, LV_OPA_COVER);

    /*Create buttons with different gradient styles*/

    lv_obj_t * btn;
    lv_obj_t * label;

    /*Simple horizontal gradient*/

```

(continues on next page)

(continua dalla pagina precedente)

```

btn = lv_button_create(lv_screen_active());
lv_obj_set_style_bg_color(btn, grad_colors[0], 0);
lv_obj_set_style_bg_grad_color(btn, grad_colors[1], 0);
lv_obj_set_style_bg_grad_dir(btn, LV_GRAD_DIR_HOR, 0);
lv_obj_set_size(btn, 150, 50);
lv_obj_align(btn, LV_ALIGN_CENTER, 0, -100);

label = lv_label_create(btn);
lv_label_set_text(label, "Horizontal");
lv_obj_center(label);

/*Simple vertical gradient*/
btn = lv_button_create(lv_screen_active());
lv_obj_set_style_bg_color(btn, grad_colors[0], 0);
lv_obj_set_style_bg_grad_color(btn, grad_colors[1], 0);
lv_obj_set_style_bg_grad_dir(btn, LV_GRAD_DIR_VER, 0);
lv_obj_set_size(btn, 150, 50);
lv_obj_align(btn, LV_ALIGN_CENTER, 0, -40);

label = lv_label_create(btn);
lv_label_set_text(label, "Vertical");
lv_obj_center(label);

/*Complex linear gradient*/
btn = lv_button_create(lv_screen_active());
lv_obj_add_style(btn, &style_with_linear_gradient_bg, 0);
lv_obj_set_size(btn, 150, 50);
lv_obj_align(btn, LV_ALIGN_CENTER, 0, 20);

label = lv_label_create(btn);
lv_label_set_text(label, "Linear");
lv_obj_center(label);

/*Complex radial gradient*/
btn = lv_button_create(lv_screen_active());
lv_obj_add_style(btn, &style_with_radial_gradient_bg, 0);
lv_obj_set_size(btn, 150, 50);
lv_obj_align(btn, LV_ALIGN_CENTER, 0, 80);

label = lv_label_create(btn);
lv_label_set_text(label, "Radial");
lv_obj_center(label);
}

#else

void lv_example_style_18(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, LV_PCT(80));
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS is not enabled");
    lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif /*LV_BUILD_EXAMPLES*/

```

Per ogni tipo di gradiente è possibile assegnare più valori di colore e opacità. Questi sono chiamati "stop". Il numero massimo di stop è limitato a [LV_GRADIENT_MAX_STOPS](#).

Un gradiente è fondamentalmente una transizione di colori e opacità tra gli stop.

Oltre a impostare il colore e l'opacità di ogni stop, è anche possibile impostare il punto di inizio rispetto all'intera area del gradiente.

Ad esempio con 3 stop può essere impostato così:

- 10% rosso: 0-10% completamente rosso
- 60% verde: transizione dal 10 al 60% da rosso a verde, il 60% è completamente verde
- 65% blu: transizione rapida dal verde al blu tra 60%--65%. Dopo il 65% completamente blu.

La posizione degli stop è chiamata frazione o offset e corrisponde a valori a 8 bit, dove 0 corrisponde allo 0% e 255 al 100% dell'intera area del gradiente.

`lv_grad_init_stops(grad_dsc, colors, opas, fracs, cnt)` inizializza un descrittore del gradiente con stop contenenti il colore, l'opacità e la frazione di ogni stop.

```

static const lv_color_t colors[2] = {
    LV_COLOR_MAKE(0xe8, 0x8, 0x8),
    LV_COLOR_MAKE(0x79, 0x79, 0x79),
};

static const lv_opa_t opas[2] = {
    170,
    255,
};

static const uint8_t fracs[2] = {
    170,
    255,
};

lv_grad_init_stops(&grad, colors, opas, fracs, sizeof(colors) / sizeof(lv_color_t));

```

Se l'array di opacità è `NULL`, verranno utilizzati 255 colori per ogni stop. Se l'array di frazioni è `NULL`, i colori saranno distribuiti uniformemente. Ad esempio, con 3 colori: 0%, 50%, 100%

Padding [riempimento]

I gradienti lineari, radiali e conici sono definiti tra due punti o angoli. È possibile definire come riempire le aree esterne ai punti o angoli di inizio e fine:

- `LV_GRAD_EXTEND_PAD`: Ripete lo stesso colore
- `LV_GRAD_EXTEND_REPEAT`: Ripete il pattern
- `LV_GRAD_EXTEND_REFLECT`: Ripete il pattern normale e speculare alternativamente

Gradienti Orizzontali e Verticali

I tipi di gradiente più semplici e solitamente più veloci sono i gradienti orizzontali e verticali.

Dopo aver inizializzato gli stop con `lv_grad_init_stops` si chiama `lv_grad_horizontal_init(&grad_dsc)` o `lv_grad_vertical_init(&grad_dsc)` per ottenere un descrittore di gradiente orizzontale o verticale.

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

static void position_bullet(lv_event_t * e, lv_point_t * p)
{
    lv_indev_t * indev = (lv_indev_t *) lv_event_get_param(e);
    lv_indev_get_point(indev, p);

    lv_obj_t * bullet = lv_event_get_target_obj(e);
    lv_obj_t * parent = lv_obj_get_parent(bullet);

    p->x -= lv_obj_get_x(parent);
    p->y -= lv_obj_get_y(parent);

    int32_t w = lv_obj_get_width(parent);
    int32_t h = lv_obj_get_height(parent);
    lv_obj_set_pos(bullet, LV_CLAMP(5, p->x, w - 20), LV_CLAMP(5, p->y, h - 20));
}

static void frac_1_event_cb(lv_event_t * e)
{
    lv_style_t * style = (lv_style_t *) lv_event_get_user_data(e);
    lv_style_value_t v;
    if(lv_style_get_prop(style, LV_STYLE_BG_GRAD, &v) != LV_STYLE_RES_FOUND) {
        LV_LOG_WARN("style prop not found");
    }
    else {
        lv_grad_dsc_t * dsc = (lv_grad_dsc_t *)v.ptr;

        lv_point_t p;
        position_bullet(e, &p);

        lv_obj_t * bullet = lv_event_get_target_obj(e);
        lv_obj_t * parent = lv_obj_get_parent(bullet);
        dsc->stops[0].frac = (uint8_t)LV_CLAMP(0, p.x * 255 / lv_obj_get_width(parent), 255);

        lv_obj_invalidate(parent);
    }
}

static void frac_2_event_cb(lv_event_t * e)
{
    lv_style_t * style = (lv_style_t *) lv_event_get_user_data(e);
    lv_style_value_t v;
    if(lv_style_get_prop(style, LV_STYLE_BG_GRAD, &v) != LV_STYLE_RES_FOUND) {
        LV_LOG_WARN("style prop not found");
    }
    else {
        lv_grad_dsc_t * dsc = (lv_grad_dsc_t *)v.ptr;

        lv_point_t p;
        position_bullet(e, &p);

        lv_obj_t * bullet = lv_event_get_target_obj(e);
        lv_obj_t * parent = lv_obj_get_parent(bullet);

        dsc->stops[1].frac = (uint8_t)LV_CLAMP(0, p.x * 255 / lv_obj_get_width(parent), 255);
        lv_obj_invalidate(parent);
    }
}

/**
 * Play with a simple horizontal gradient.
 * Adjust the stop positions of the gradient.
 */
void lv_example_grad_1(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0xff, 0x00, 0x00),
        LV_COLOR_MAKE(0x00, 0xff, 0x00),
    };

    static const lv_opa_t grad_opa[2] = {
        LV_OPA_100,
        LV_OPA_0,
    };
}
```

(continues on next page)

(continua dalla pagina precedente)

```

static const uint8_t frac[2] = {
    20 * 255 / 100, /*20%/
    80 * 255 / 100, /*80%/
};

static lv_style_t style;
lv_style_init(&style);

static lv_grad_dsc_t grad_dsc;
lv_grad_init_stops(&grad_dsc, grad_colors, grad_opa, frac, sizeof(grad_colors) / sizeof(lv_color_t));
lv_grad_horizontal_init(&grad_dsc);

/*Set gradient as background*/
lv_style_set_bg_grad(&style, &grad_dsc);
lv_style_set_border_width(&style, 2);
lv_style_set_pad_all(&style, 0);
lv_style_set_radius(&style, 12);

/*Create an object with the new style*/
lv_obj_t * obj = lv_obj_create(lv_screen_active());
lv_obj_add_style(obj, &style, 0);
lv_obj_set_size(obj, lv_pct(80), lv_pct(80));
lv_obj_center(obj);

lv_obj_t * frac_1 = lv_button_create(obj);
lv_obj_set_size(frac_1, 15, 15);
lv_obj_set_style_bg_color(frac_1, lv_color_hex(0xffff00), 0);
lv_obj_add_event_cb(frac_1, frac_1_event_cb, LV_EVENT_PRESSING, &style);
lv_obj_set_ext_click_area(frac_1, 5);
lv_obj_set_pos(frac_1, lv_pct(20), lv_pct(50));

lv_obj_t * frac_2 = lv_button_create(obj);
lv_obj_set_size(frac_2, 15, 15);
lv_obj_set_style_bg_color(frac_2, lv_color_hex(0xff00ff), 0);
lv_obj_add_event_cb(frac_2, frac_2_event_cb, LV_EVENT_PRESSING, &style);
lv_obj_set_ext_click_area(frac_2, 5);
lv_obj_set_pos(frac_2, lv_pct(80), lv_pct(50));
}

#endif

```

Gradienti Lineari

Il gradiente lineare (o skew) è simile al gradiente orizzontale o verticale, ma l'angolo del gradiente può essere controllato.

Il gradiente lineare verrà renderizzato lungo una linea definita da 2 punti.

Dopo aver inizializzato gli stop con `lv_grad_init_stops()` si chiama `lv_grad_linear_init(&grad_dsc, from_x, from_y, to_x, to_y, LV_GRAD_EXTEND_...)` con i valori dei punti e si estende la strategia del pattern per ottenere un descrittore di gradiente lineare.

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL
#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS
static void position_bullet(lv_event_t * e, lv_point_t * p)
{
    lv_indev_t * indev = (lv_indev_t *) lv_event_get_param(e);
    lv_indev_get_point(indev, p);

    lv_obj_t * bullet = lv_event_get_target_obj(e);
    lv_obj_t * parent = lv_obj_get_parent(bullet);

    p->x = lv_obj_get_x(parent);
    p->y = lv_obj_get_y(parent);

    int32_t w = lv_obj_get_width(parent);
    int32_t h = lv_obj_get_height(parent);
    lv_obj_set_pos(bullet, LV_CLAMP(5, p->x, w - 20), LV_CLAMP(5, p->y, h - 20));
}

static void start_event_cb(lv_event_t * e)
{
    lv_style_t * style = (lv_style_t *) lv_event_get_user_data(e);
    lv_style_value_t v;
    lv_style_get_prop(style, LV_STYLE_BG_GRAD, &v);
    lv_grad_dsc_t * dsc = (lv_grad_dsc_t *) v.ptr;

    lv_point_t p;
    position_bullet(e, &p);

    dsc->params.linear.start.x = p.x;
    dsc->params.linear.start.y = p.y;

    lv_obj_t * bullet = lv_event_get_target_obj(e);
    lv_obj_t * parent = lv_obj_get_parent(bullet);
    lv_obj_invalidate(parent);
}

static void end_event_cb(lv_event_t * e)
{
    lv_style_t * style = (lv_style_t *) lv_event_get_user_data(e);
    lv_style_value_t v;
    lv_style_get_prop(style, LV_STYLE_BG_GRAD, &v);
    lv_grad_dsc_t * dsc = (lv_grad_dsc_t *) v.ptr;

    lv_point_t p;
    position_bullet(e, &p);

    dsc->params.linear.end.x = p.x;
    dsc->params.linear.end.y = p.y;
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * bullet = lv_event_get_target_obj(e);
lv_obj_t * parent = lv_obj_get_parent(bullet);
lv_obj_invalidate(parent);
}

/**
 * Play with the linear gradient.
 * Adjust the 2 point in between the a linear gradient can be drawn (can be skew as well)
 */
void lv_example_grad_2(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0xff, 0x00, 0x00),
        LV_COLOR_MAKE(0x00, 0xff, 0x00),
    };

    static const lv_opa_t grad_opa[2] = {
        LV_OPA_100,
        LV_OPA_0,
    };

    static lv_style_t style;
    lv_style_init(&style);

    /*First define a color gradient. In this example we use a purple to black color map.*/
    static lv_grad_dsc_t grad;
    lv_grad_init_stops(&grad, grad_colors, grad_opa, NULL, sizeof(grad_colors) / sizeof(lv_color_t));

    lv_grad_linear_init(&grad, 100, 100, 200, 150, LV_GRAD_EXTEND_PAD);

    /*Set gradient as background*/
    lv_style_set_bg_grad(&style, &grad);
    lv_style_set_border_width(&style, 2);
    lv_style_set_pad_all(&style, 0);
    lv_style_set_radius(&style, 12);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_set_size(obj, lv_pct(80), lv_pct(80));
    lv_obj_center(obj);

    lv_obj_t * start = lv_button_create(obj);
    lv_obj_set_size(start, 15, 15);
    lv_obj_set_style_bg_color(start, lv_color_hex(0x0000ff), 0);
    lv_obj_add_event_cb(start, start_event_cb, LV_EVENT_PRESSING, &style);
    lv_obj_set_ext_click_area(start, 5);
    lv_obj_set_pos(start, 100, 100);

    lv_obj_t * end = lv_button_create(obj);
    lv_obj_set_size(end, 15, 15);
    lv_obj_set_style_bg_color(end, lv_color_hex(0x00ffff), 0);
    lv_obj_add_event_cb(end, end_event_cb, LV_EVENT_PRESSING, &style);
    lv_obj_set_ext_click_area(end, 5);
    lv_obj_set_pos(end, 200, 150);
}

#else

void lv_example_grad_2(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS needs to be enabled");
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif

```

Gradienti Radiali

Il gradiente radiale è descritto da due cerchi: un cerchio esterno e un cerchio interno (chiamato anche punto focale). Il gradiente verrà calcolato tra il cerchio del punto focale e il bordo del cerchio esterno.

Se il centro del punto focale e il centro del cerchio principale coincidono, il gradiente si distribuirà uniformemente in tutte le direzioni. Se i punti centrali non coincidono, il gradiente avrà una forma a uovo.

Il cerchio del punto focale dovrebbe trovarsi all'interno di quello principale.

Dopo aver inizializzato gli stop con `lv_grad_init_stops()`, il cerchio esterno può essere impostato con: `lv_grad_radial_init(&grad_dsc, center_x, center_y, edge_x, edge_y, LV_GRAD_EXTEND_...)`

Per le coordinate del centro e del bordo, è possibile utilizzare i valori px o `lv_pct()`.

Il cerchio interno (punto focale) può essere impostato con: `lv_grad_radial_set_focal(&grad_dsc, center_x, center_y, radius)`

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

static void position_bullet(lv_event_t * e, lv_point_t * p)
{
    lv_indev_t * indev = (lv_indev_t *) lv_event_get_param(e);

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_indev_get_point(indev, p);

lv_obj_t * bullet = lv_event_get_target_obj(e);
lv_obj_t * parent = lv_obj_get_parent(bullet);

p->x = lv_obj_get_x(parent);
p->y = lv_obj_get_y(parent);

int32_t w = lv_obj_get_width(parent);
int32_t h = lv_obj_get_height(parent);
lv_obj_set_pos(bullet, LV_CLAMP(5, p->x, w - 20), LV_CLAMP(5, p->y, h - 20));
}

static void focal_event_cb(lv_event_t * e)
{
    lv_style_t * style = (lv_style_t *) lv_event_get_user_data(e);
    lv_style_value_t v;
    lv_style_get_prop(style, LV_STYLE_BG_GRAD, &v);
    lv_grad_dsc_t * dsc = (lv_grad_dsc_t *)v.ptr;

    lv_point_t p;
    position_bullet(e, &p);

    dsc->params.radial.focal.x = p.x;
    dsc->params.radial.focal.y = p.y;
    dsc->params.radial.focal_extent.x = p.x + 10;
    dsc->params.radial.focal_extent.y = p.y;

    lv_obj_t * bullet = lv_event_get_target_obj(e);
    lv_obj_t * parent = lv_obj_get_parent(bullet);
    lv_obj_invalidate(parent);
}

static void end_event_cb(lv_event_t * e)
{
    lv_style_t * style = (lv_style_t *) lv_event_get_user_data(e);
    lv_style_value_t v;
    lv_style_get_prop(style, LV_STYLE_BG_GRAD, &v);
    lv_grad_dsc_t * dsc = (lv_grad_dsc_t *)v.ptr;

    lv_point_t p;
    position_bullet(e, &p);

    dsc->params.radial.end.x = p.x;
    dsc->params.radial.end.y = p.y;
    dsc->params.radial.end_extent.x = p.x + 100;
    dsc->params.radial.end_extent.y = p.y;
    lv_obj_t * bullet = lv_event_get_target_obj(e);
    lv_obj_t * parent = lv_obj_get_parent(bullet);
    lv_obj_invalidate(parent);
}

/**
 * Play with the radial gradient
 * Adjust the end circle and focal point position.
 * The radius of the end circle and an focal point are hardcoded in the example.
 */
void lv_example_grad_3(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0xff, 0x00, 0x00),
        LV_COLOR_MAKE(0x00, 0xff, 0x00),
    };

    static const lv_opa_t grad_opa[2] = {
        LV_OPA_100,
        LV_OPA_0,
    };

    static lv_style_t style;
    lv_style_init(&style);

    /*First define a color gradient. In this example we use a purple to black color map.*/
    static lv_grad_dsc_t grad;

    lv_grad_init_stops(&grad, grad_colors, grad_opa, NULL, sizeof(grad_colors) / sizeof(lv_color_t));

    /*Init a radial gradient where the center is at 100;100
     *and the edge of the circle is at 200;100.
     *Try LV_GRAD_EXTEND_REFLECT and LV_GRAD_EXTEND_REPEAT too. */
    lv_grad_radial_init(&grad, 100, 100, 200, 100, LV_GRAD_EXTEND_PAD);

    /*The gradient will be calculated between the focal point's circle and the
     *edge of the circle. If the center of the focal point and the
     *center of the main circle is the same, the gradient will spread
     *evenly in all directions. The focal point should be inside the
     *main circle.*/
    lv_grad_radial_set_focal(&grad, 50, 50, 10);

    /*Set the widget containing the gradient*/
    lv_style_set_bg_grad(&style, &grad);
    lv_style_set_border_width(&style, 2);
    lv_style_set_pad_all(&style, 0);
    lv_style_set_radius(&style, 12);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_set_size(obj, lv_pct(80), lv_pct(80));
    lv_obj_center(obj);

    lv_obj_t * focal = lv_button_create(obj);
    lv_obj_set_size(focal, 15, 15);
    lv_obj_set_style_bg_color(focal, lv_color_hex(0x0000ff), 0);
    lv_obj_add_event_cb(focal, focal_event_cb, LV_EVENT_PRESSING, &style);
    lv_obj_set_ext_click_area(focal, 5);
    lv_obj_set_pos(focal, 50, 50);

    lv_obj_t * end = lv_button_create(obj);
    lv_obj_set_size(end, 15, 15);
    lv_obj_set_style_bg_color(end, lv_color_hex(0x00ffff), 0);
    lv_obj_add_event_cb(end, end_event_cb, LV_EVENT_PRESSING, &style);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_ext_click_area(end, 5);
lv_obj_set_pos(end, 100, 100);
}

void lv_example_grad_3(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS needs to be enabled");
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif

```

Gradienti Conici

Il gradiente conico è definito tra gli angoli all'interno di un cerchio e i colori sono mappati a ciascun angolo.

Dopo aver inizializzato gli stop con `lv_grad_init_stops()`, il gradiente conico può essere impostato con: `lv_grad_conical_init(&grad, center_x, center_y, angle_start, angle_end, LV_GRAD_EXTEND_...)`

Per le coordinate del centro e del bordo, è possibile utilizzare i valori `px` o `lv_pct()`.

L'angolo zero si trova sul lato destro e quello di 90 gradi in basso.

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

static void position_bullet(lv_event_t * e, lv_point_t * p)
{
    lv_indev_t * indev = (lv_indev_t *) lv_event_get_param(e);
    lv_indev_get_point(indev, p);

    lv_obj_t * bullet = lv_event_get_target_obj(e);
    lv_obj_t * parent = lv_obj_get_parent(bullet);

    p->x -= lv_obj_get_x(parent);
    p->y -= lv_obj_get_y(parent);

    int32_t w = lv_obj_get_width(parent);
    int32_t h = lv_obj_get_height(parent);
    lv_obj_set_pos(bullet, LV_CLAMP(5, p->x, w - 20), LV_CLAMP(5, p->y, h - 20));
}

static void start_event_cb(lv_event_t * e)
{
    lv_style_t * style = (lv_style_t *) lv_event_get_user_data(e);
    lv_style_value_t v;
    lv_style_get_prop(style, LV_STYLE_BG_GRAD, &v);
    lv_grad_dsc_t * dsc = (lv_grad_dsc_t *)v.ptr;

    lv_point_t p;
    position_bullet(e, &p);

    lv_obj_t * bullet = lv_event_get_target_obj(e);
    lv_obj_t * parent = lv_obj_get_parent(bullet);
    p.x -= lv_obj_get_width(parent) / 2;
    p.y -= lv_obj_get_height(parent) / 2;

    dsc->params.conical.start_angle = lv_atan2(p.y, p.x);
    lv_obj_invalidate(parent);
}

static void end_event_cb(lv_event_t * e)
{
    lv_style_t * style = (lv_style_t *) lv_event_get_user_data(e);
    lv_style_value_t v;
    lv_style_get_prop(style, LV_STYLE_BG_GRAD, &v);
    lv_grad_dsc_t * dsc = (lv_grad_dsc_t *)v.ptr;

    lv_point_t p;
    position_bullet(e, &p);

    lv_obj_t * bullet = lv_event_get_target_obj(e);
    lv_obj_t * parent = lv_obj_get_parent(bullet);
    p.x -= lv_obj_get_width(parent) / 2;
    p.y -= lv_obj_get_height(parent) / 2;

    dsc->params.conical.end_angle = lv_atan2(p.y, p.x);
    lv_obj_invalidate(parent);
}

/***
 * Play with the conical gradient
 */
void lv_example_grad_4(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0xff, 0x00, 0x00),
        LV_COLOR_MAKE(0x00, 0xff, 0x00),
    };

    static const lv_opa_t grad_opa[2] = {
        LV_OPA_100,
        LV_OPA_0,
    };

    static lv_style_t style;
    lv_style_init(&style);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

/*First define a color gradient. In this example we use a purple to black color map.*/
static lv_grad_desc_t grad;

lv_grad_init_stops(&grad, grad_colors, grad_opa, NULL, sizeof(grad_colors) / sizeof(lv_color_t));

lv_grad_conical_init(&grad, lv_pct(50), lv_pct(50), 0, 180, LV_GRAD_EXTEND_PAD);

/*Set gradient as background*/
lv_style_set_bg_grad(&style, &grad);
lv_style_set_border_width(&style, 2);
lv_style_set_pad_all(&style, 0);
lv_style_set_radius(&style, 12);

/*Create an object with the new style*/
lv_obj_t * obj = lv_obj_create(lv_screen_active());
lv_obj_add_style(obj, &style, 0);
lv_obj_set_size(obj, lv_pct(80), lv_pct(80));
lv_obj_center(obj);

lv_obj_t * start = lv_button_create(obj);
lv_obj_set_size(start, 15, 15);
lv_obj_set_style_bg_color(start, lv_color_hex(0x0000ff), 0);
lv_obj_add_event_cb(start, start_event_cb, LV_EVENT_PRESSING, &style);
lv_obj_set_ext_click_area(start, 5);
lv_obj_set_pos(start, lv_pct(80), lv_pct(50));

lv_obj_t * end = lv_button_create(obj);
lv_obj_set_size(end, 15, 15);
lv_obj_set_style_bg_color(end, lv_color_hex(0x00ffff), 0);
lv_obj_add_event_cb(end, end_event_cb, LV_EVENT_PRESSING, &style);
lv_obj_set_ext_click_area(end, 5);
lv_obj_set_pos(end, lv_pct(20), lv_pct(50));
}

#else

void lv_example_grad_4(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS needs to be enabled");
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif

```

Descrittore di Disegno del Bordo

Il descrittore del bordo `lv_draw_border_desc_t` ha i campi raggio, opacità, larghezza, colore e lato. Se l'opacità o la larghezza sono pari a 0, non verrà creata alcun Task di Disegno.

`side` può contenere valori OR di `lv_border_side_t`, come `LV_BORDER_SIDE_BOTTOM`. `LV_BORDER_SIDE_ALL` si applica a tutti i lati, mentre `LV_BORDER_SIDE_INTERNAL` viene utilizzato dai layer superiori (ad esempio un Widget Table) per calcolare i lati del bordo. Tuttavia, la routine di disegno riceve solo valori più semplici.

Per disegnare i bordi vengono utilizzate le seguenti funzioni:

- `lv_draw_border_desc_init(&dsc)` inizializza un Task di Disegno dei bordi.
- `lv_draw_sw_border(layer, &dsc, area)` crea un Task per il Disegno di un bordo verso l'interno rispetto alla sua area.
- `lv_draw_task_get_border_desc(draw_task)` recupera il descrittore del bordo da un Task di Disegno.

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the border style properties
 */
void lv_example_style_3(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 10);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add border to the bottom+right*/
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_border_width(&style, 5);
    lv_style_set_border_opa(&style, LV_OPA_50);
    lv_style_set_border_side(&style, (lv_border_side_t)(LV_BORDER_SIDE_BOTTOM | LV_BORDER_SIDE_RIGHT));

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif

```

Contorni

Il contorno è simile al bordo, ma viene disegnato all'esterno dell'area di disegno dell'oggetto.

In pratica, non esiste un descrittore di contorno dedicato come `lv_draw_outline_dsc_t`, perché dal punto di vista del rendering, il contorno è semplicemente un altro bordo renderizzato all'esterno dei limiti dell'oggetto.

Il contorno viene utilizzato solo in `lv_draw_rect_dsc_t` per comodità. Le due differenze rispetto ai bordi in `lv_draw_rect_dsc_t` sono:

- Esiste una proprietà `outline_pad` per specificare lo spazio tra l'area target e il lato interno del contorno. Può essere negativa. Ad esempio, se `outline_pad = -width`, il contorno assomiglierà a un bordo.
- Non esiste una proprietà `border_side` per il contorno. Viene sempre renderizzato come un rettangolo pieno.

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the outline style properties
 */
void lv_example_style_4(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add outline*/
    lv_style_set_outline_width(&style, 2);
    lv_style_set_outline_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_outline_pad(&style, 8);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

Descrittore di Disegno del Riquadro dell'Ombra

Il descrittore di disegno dell'ombra a riquadro `lv_draw_box_shadow_dsc_t` descrive un'**ombra rettangolare arrotondata**. Non può generare ombre per forme, testo o immagini arbitrarie. Include i seguenti campi:

raggio

Raggio, `LV_RADIUS_CIRCLE`.

colore

Colore dell'ombra.

width

Aampiezza dell'ombra (raggio di sfocatura).

estensione

Espande il rettangolo in tutte le direzioni; può essere negativo.

ofs_x

Offset orizzontale.

ofs_y

Offset verticale.

opa

Opacità (intervallo 0-255). È possibile utilizzare anche valori come `LV_OPA_TRANSP`, `LV_OPA_10`, ecc.

bg_cover

Impostare a 1 se il background coprirà l'ombra (suggerimento per il motore di rendering di evitare il mascheramento).

Nota: Il rendering di ombre di grandi dimensioni può essere lento o richiedere molta memoria.

Le seguenti funzioni vengono utilizzate per disegnare l'ombra di un rettangolo:

- `lv_draw_box_shadow_dsc_init(&dsc)` Inizializza un Task di Disegno dell'ombra rettangolare.

- `lv_draw_sw_box_shadow(layer, &dsc, area)` crea un Task di Disegno per l'ombra rettangolare. Le dimensioni e la posizione dell'ombra dipendono da larghezza, estensione e offset.
- `lv_draw_task_get_box_shadow_dsc(draw_task)` recupera il descrittore dell'ombra rettangolare da un Task di Disegno.

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the Shadow style properties
 */
void lv_example_style_5(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add a shadow*/
    lv_style_set_shadow_width(&style, 55);
    lv_style_set_shadow_color(&style, lv_palette_main(LV_PALETTE_BLUE));

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

Descrittore di Disegno dell'Immagine

Il descrittore dell'immagine `lv_draw_image_dsc_t` definisce i parametri per il disegno dell'immagine. Si tratta di un descrittore complesso con le seguenti opzioni:

src

L'origine dell'immagine, un puntatore a `lv_image_dsc_t` o un path di un file.

opa

Opacità compresa tra 0 e 255. È possibile utilizzare anche opzioni come `LV_OPA_TRANSP`, `LV_OPA_10`, ecc.

clip_radius

Ritaglia gli angoli dell'immagine con questo raggio. Utilizzare `LV_RADIUS_CIRCLE` per il raggio massimo.

rotazione

Rotazione dell'immagine in unità di 0,1 gradi (ad esempio, 234 significa 23,4 gradi).

scale_x

Ridimensionamento orizzontale (zoom) dell'immagine. 256 (`LV_SCALE_NONE`) significa nessuno zoom, 512 raddoppia le dimensioni e 128 le dimezza.

scale_y

Come `scale_x` ma per il ridimensionamento verticale.

skew_x

Inclinazione orizzontale (trasformazione simile a un parallelogramma) in unità di 0,1 gradi (ad esempio, 456 significa 45,6 gradi).

skew_y

Inclinazione verticale, simile a `skew_x`.

pivot

Il punto di rotazione per le trasformazioni (ridimensionamento e rotazione). (0,0) è l'angolo in alto a sinistra dell'immagine e può essere impostato all'esterno dell'immagine.

bitmap_mask_src

Puntatore a un descrittore di immagine A8 o L8 utilizzato per mascherare l'immagine. La maschera è sempre allineata al centro.

recolor

Mescola questo colore con l'immagine. Per *LV_COLOR_FORMAT_A8*, questo sarà il colore dei pixel visibili.

recolor_opa

Intensità della ricolorazione (0 significa nessuna ricolorazione, 255 significa copertura completa).

blend_mode

Definisce come fondere i pixel dell'immagine con lo sfondo. Vedere *lv_blend_mode_t* per maggiori dettagli.

antialias

Impostare su 1 per abilitare l'anti-aliasing per le trasformazioni.

tile

Rielabora l'immagine (ripetendola sia orizzontalmente che verticalmente) se l'immagine è più piccola del campo *image_area* in *lv_draw_image_dsc_t*.

image_area

Indica l'area originale, non ritagliata, in cui viene disegnata l'immagine. Questo è essenziale per:

1. Rendering dei layer, in cui è possibile renderizzare solo una parte di un layer e *clip_radius* necessita delle dimensioni dell'immagine originale.
2. Tiling, in cui l'area di disegno è più grande dell'immagine.

sup

Campo interno per memorizzare informazioni sulla tavolozza o sul colore delle immagini A8.

Funzioni per il disegno di immagini:

- *lv_draw_image_dsc_init(&dsc)* inizializza un descrittore di disegno di immagini.
- *lv_draw_image(layer, &dsc, area)* crea un task per disegnare un'immagine in un'area specifica.
- *lv_draw_task_get_image_dsc(draw_task)* recupera il descrittore di immagine da un task.

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw an image to the canvas
 */
void lv_example_canvas_6(void)
{
    /*Create a buffer for the canvas*/
    static uint8_t cbuf[LV_CANVAS_BUF_SIZE(CANVAS_WIDTH, CANVAS_HEIGHT, 32, LV_DRAW_BUF_STRIDE_ALIGN)];

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_buffer(canvas, cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    LV_IMAGE_DECLARE(img_star);
    lv_draw_image_dsc_t dsc;
    lv_draw_image_dsc_init(&dsc);
    dsc.src = &img_star;

    lv_area_t coords = {10, 10, 10 + img_star.header.w - 1, 10 + img_star.header.h - 1};

    lv_draw_image(&layer, &dsc, &coords);
    lv_canvas_finish_layer(canvas, &layer);
}
#endif
```

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Using the Image style properties
 */
void lv_example_style_6(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 3));
    lv_style_set_border_width(&style, 2);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));

lv_style_set_image_recolor(&style, lv_palette_main(LV_PALETTE_BLUE));
lv_style_set_image_recolor_opa(&style, LV_OPA_50);
lv_style_set_transform_rotation(&style, 300);

/*Create an object with the new style*/
lv_obj_t * obj = lv_image_create(lv_screen_active());
lv_obj_add_style(obj, &style, 0);

LV_IMAGE_DECLARE(img_cogwheel_argb);
lv_image_set_src(obj, &img_cogwheel_argb);

lv_obj_center(obj);
}

#endif

```

Layer - Immagini Speciali

I layer sono trattati come immagini, quindi un `lv_draw_image_dsc_t` può descrivere come i layer vengono fusi con i layer genitori. Tutte le funzionalità dell'immagine si applicano anche ai layer.

`lv_draw_layer(layer, &dsc, area)` inizializza la fusione di un layer con il suo layer padre. Inoltre, per i layer possono essere utilizzate funzioni relative al disegno delle immagini.

Per maggiori dettagli, vedere [ILayer](#).

Descrittore di Disegno dell'Etichetta

Il descrittore dell'etichetta `lv_draw_label_dsc_t` offre ampie opzioni per il controllo del rendering del testo:

text

Testo da visualizzare.

font

Font da utilizzare, con supporto per font alternativi.

colore

Colore del testo.

opa

Opacità del testo.

line_space

Spazio aggiuntivo tra le righe.

letter_space

Spazio aggiuntivo tra i caratteri.

ofs_x

Offset orizzontale del testo.

ofs_y

Offset verticale del testo.

sel_start

Indice del primo carattere da selezionare (indice del carattere, non indice del byte, poiché alcuni caratteri possono essere caratteri multibyte). `LV_DRAW_LABEL_NO_TXT_SEL` significa nessuna selezione.

sel_end

Indice dell'ultimo carattere per la selezione.

sel_color

Colore dei caratteri selezionati.

sel_bg_color

Colore di sfondo per i caratteri selezionati.

align

Allineamento del testo. Vedere `lv_text_align_t`.

bidi_dir

Orientamento di base per la visualizzazione del testo da destra a sinistra (ad esempio, arabo). Vedere [lv_base_dir_t](#).

decor

Decorazione del testo, ad esempio, sottolineatura. Vedere [lv_text_decor_t](#).

flag

Flag per il rendering del testo. Vedere [lv_text_flag_t](#).

text_length

Numero di caratteri da visualizzare (0 indica il rendering fino a \0).

text_local

Impostare a 1 per allocare un buffer e copiare il testo.

text_static

Indica che `text` è costante e il suo puntatore può essere memorizzato nella cache.

hint

Puntatore a dati memorizzati esternamente per velocizzare il rendering. Vedere [lv_draw_label_hint_t](#).

Funzioni per il disegno di testo:

- [lv_draw_label_dsc_init](#)(`&dsc`) inizializza un descrittore di disegno dell'etichetta.
- [lv_draw_label](#)(`layer, &dsc, area`) crea un task per il rendering del testo in un'area.
- [lv_draw_character](#)(`layer, &dsc, point, unicode_letter`) crea un task per disegnare un carattere in un punto specifico.
- [lv_draw_task_get_label_dsc](#)(`draw_task`) recupera il descrittore di etichetta da un task.

Per il disegno di caratteri specifici in unità di disegno, si usa [lv_draw_label_iterate_characters](#)(`draw_unit, draw_dsc, area, callback`). Questa itera su tutti i caratteri, calcola le loro posizioni e richiama la callback per il rendering di ciascun carattere. Per i dettagli sulla callback, vedere [lv_draw_glyph_cb_t](#).

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_FONT_MONTserrat_18 && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw a text to the canvas
 */
void lv_example_canvas_4(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_label_dsc_t dsc;
    lv_draw_label_dsc_init(&dsc);
    dsc.color = lv_palette_main(LV_PALETTE_RED);
    dsc.font = &lv_font_montserrat_18;
    dsc.decor = LV_TEXT_DECOR_UNDERLINE;
    dsc.text = "Hello";

    lv_area_t coords = {10, 10, 30, 60};

    lv_draw_label(&layer, &dsc, &coords);

    lv_canvas_finish_layer(canvas, &layer);
}

#endif
```

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

/***
 * Using the text style properties
 */
void lv_example_style_B(void)
{
    static lv_style_t style;
    lv_style_init(&style);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_set_radius(&style, 5);
lv_style_set_bg_opa(&style, LV_OPA_COVER);
lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 2));
lv_style_set_border_width(&style, 2);
lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));
lv_style_set_pad_all(&style, 10);

lv_style_set_text_color(&style, lv_palette_main(LV_PALETTE_BLUE));
lv_style_set_text_letter_space(&style, 5);
lv_style_set_text_line_space(&style, 20);
lv_style_set_text_decor(&style, LV_TEXT_DECOR_UNDERLINE);

/*Create an object with the new style*/
lv_obj_t * obj = lv_label_create(lv_screen_active());
lv_obj_add_style(obj, &style, 0);
lv_label_set_text(obj, "Text of\n"
                     "a label");

lv_obj_center(obj);
}

#endif

```

Descrittore di Disegno dell'Arco

Il descrittore di disegno dell'arco `lv_draw_arc_desc_t` definisce il rendering dell'arco con questi campi:

colore

Colore dell'arco.

img_src

Sorgente dell'immagine per l'arco, o NULL se non utilizzato.

width

Spessore dell'arco.

start_angle

Angolo iniziale in gradi (ad esempio, 0° corrisponde alle ore 3, 90° alle ore 6).

end_angle

Angolo finale.

center

Punto centrale dell'arco.

raggio

Raggio dell'arco.

opa

Opacità dell'arco (0-255).

rounded

Arrotonda le estremità dell'arco.

Funzioni per il disegno dell'arco:

- `lv_draw_arc_desc_init(&dsc)` inizializza un descrittore di arco.
- `lv_draw_arc(layer, &dsc)` crea un task per il rendering di un arco.
- `lv_draw_task_get_arc_desc(draw_task)` recupera il descrittore dell'arco dal task.

```

#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw an arc to the canvas
 */
void lv_example_canvas_5(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;

```

(continues on next page)

(continua dalla pagina precedente)

```
lv_canvas_init_layer(canvas, &layer);

lv_draw_arc_dsc_t dsc;
lv_draw_arc_dsc_init(&dsc);
dsc.color = lv_palette_main(LV_PALETTE_RED);
dsc.center.x = 25;
dsc.center.y = 25;
dsc.width = 10;
dsc.radius = 15;
dsc.start_angle = 0;
dsc.end_angle = 220;

lv_draw_arc(&layer, &dsc);

lv_canvas_finish_layer(canvas, &layer);

}
#endif
```

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_ARC

/***
 * Using the Arc style properties
 */
void lv_example_style_7(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_arc_color(&style, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_arc_width(&style, 4);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_arc_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}
#endif
```

Descrittore di Disegno della Linea

Il descrittore della linea `lv_draw_line_dsc_t` definisce il rendering della linea con questi campi:

p1

Primo punto della linea (supporta coordinate in virgola mobile).

p2

Secondo punto della linea (supporta coordinate in virgola mobile).

colore

Colore della linea.

width

Spessore della linea.

opa

Opacità della linea (0-255).

dash_width

Lunghezza dei trattini (0 significa nessun tratto --- una linea continua).

dash_gap

Lunghezza degli spazi tra i trattini (0 significa nessuno spazio, ovvero una linea continua).

round_start

Arrotonda l'inizio della linea.

round_end

Arrotonda la fine della linea.

raw_end

Impostare su 1 per saltare i calcoli finali se non sono necessari.

Funzioni per il disegno di linee:

- `lv_draw_line_dsc_init(&dsc)` inizializza un descrittore di linea.
- `lv_draw_line(layer, &dsc)` crea un task per disegnare una linea.
- `lv_draw_task_get_line_dsc(draw_task)` recupera il descrittore di linea.

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS&& LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw a line to the canvas
 */
void lv_example_canvas_7(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_line_dsc_t dsc;
    lv_draw_line_dsc_init(&dsc);
    dsc.color = lv_palette_main(LV_PALETTE_RED);
    dsc.width = 4;
    dsc.round_end = 1;
    dsc.round_start = 1;
    dsc.p1.x = 15;
    dsc.p1.y = 15;
    dsc.p2.x = 35;
    dsc.p2.y = 10;
    lv_draw_line(&layer, &dsc);

    lv_canvas_finish_layer(canvas, &layer);
}

#endif

```

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LINE

/***
 * Using the line style properties
 */
void lv_example_style_9(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_line_color(&style, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_line_width(&style, 6);
    lv_style_set_line_rounded(&style, true);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_line_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);

    static lv_point_precise_t p[] = {{10, 30}, {30, 50}, {100, 0}};
    lv_line_set_points(obj, p, 3);

    lv_obj_center(obj);
}

#endif

```

Descrittore di Disegno del Triangolo

I triangoli sono definiti da `lv_draw_triangle_dsc_t`, che include:

p[3]

3 punti per i vertici del triangolo.

colore

Colore del triangolo.

opa

Opacità del triangolo.

grad

Opzioni del gradiente. Se `grad.dir` non è `LV_GRAD_DIR_NONE`, il campo `color` viene ignorato. Il campo `opa` regola l'opacità complessiva.

Funzioni per il disegno del triangolo:

- `lv_draw_triangle_dsc_init(&dsc)` inizializza un descrittore di triangolo.
- `lv_draw_triangle(layer, &dsc)` crea un task per disegnare un triangolo.
- `lv_draw_task_get_triangle_dsc(draw_task)` recupera il descrittore del triangolo.

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 150
#define CANVAS_HEIGHT 150

/***
 * Draw a triangle to the canvas
 */
void lv_example_canvas_9(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_triangle_dsc_t tri_dsc;
    lv_draw_triangle_dsc_init(&tri_dsc);
    tri_dsc.p[0].x = 10;
    tri_dsc.p[0].y = 10;
    tri_dsc.p[1].x = 100;
    tri_dsc.p[1].y = 30;
    tri_dsc.p[2].x = 50;
    tri_dsc.p[2].y = 100;

    tri_dsc.grad.stops_count = 2;
    tri_dsc.grad.dir = LV_GRAD_DIR_VER;
    tri_dsc.grad.stops[0].color = lv_color_hex(0xffff0000);
    tri_dsc.grad.stops[0].frac = 64; /*Start at 25%*/
    tri_dsc.grad.stops[0].opa = LV_OPA_COVER;
    tri_dsc.grad.stops[1].color = lv_color_hex(0x0000ff);
    tri_dsc.grad.stops[1].opa = LV_OPA_TRANS;
    tri_dsc.grad.stops[1].frac = 3 * 64; /*End at 75%*/

    tri_dsc.opa = 128; /*Set the overall opacity to 50%*/

    lv_draw_triangle(&layer, &tri_dsc);
    lv_canvas_finish_layer(canvas, &layer);
}

#endif
```

Descrittore di Disegno Vettoriale

TODO [da fare]

Operazione di Mascheramento

Esistono diverse opzioni per mascherare parti di un layer, Widget o disegno:

- Raggio dei rettangoli:** Imposta la proprietà di stile `radius` o il `radius` nei descrittori di disegno. Questo crea rettangoli, bordi, contorni, ecc. arrotondati. Tuttavia, il contenuto dei rendering successivi non verrà mascherato negli angoli.
 - Raggio di Ritaglio delle Immagini:** Analogamente ai rettangoli, anche le immagini possono essere renderrizzate con un raggio. Poiché il disegno dei layer e quello delle immagini vengono gestiti allo stesso modo, questo funziona anche per i layer.
- È possibile disegnare vari contenuti su un layer e quindi renderizzare il layer con un `clip_radius`, mascherando tutto il contenuto sugli angoli.
- Task di Disegno con Maschera Rettangolare:** Un Task di disegno speciale può mascherare un rettangolo da un layer impostando il canale alfa di determinati pixel a 0. Per ottenere questo risultato:

- Creare un descrittore `lv_draw_mask_rect_dsc_t`.
- Impostare i parametri `area`, `radius` e `keep_outside`. Se `keep_outside` è impostato su 1, le aree esterne ad `area` rimangono invariate. In caso contrario, vengono cancellate.
- Chiamare `lv_draw_mask_rect(layer, &dsc)`.

Nota: Il layer deve avere un formato colore con un canale alfa, in genere `LV_COLOR_FORMAT_ARGB8888`.

Nella maggior parte dei casi, il metodo del *'Ritagliare il Raggio Immagini'* è migliore perché fonde il layer con una maschera di raggio al volo, evitando una fase di mascheratura dedicata. Tuttavia, il *"Task di Disegno di Maschera Rettangolare"* è utile quando è necessario pulire più aree o quando l'area da mascherare è diversa dall'area del layer.

4. **Proprietà Stile dell'Angolo di Ritaglio:** Abilitando `..._style_clip_corner` in uno stile locale o globale, LVGL crea un layer per le aree degli angoli superiore e inferiore di un Widget. Esegue il rendering degli elementi figlio in queste aree e li fonde impostando `clip_radius` sul layer.
5. **Mascheramento Bitmap per Immagini:** Utilizzando `..._style_bitmap_mask` o `bitmap_mask` in `lv_draw_image_dsc_t` è possibile impostare un'immagine A8 o L8 come maschera per un'immagine/layer durante il blending.
 - Limitazione: La maschera si allinea sempre al centro e per un'immagine/layer è possibile utilizzare una sola maschera bitmap.
 - Quando si utilizza `..._style_bitmap_mask`, LVGL crea automaticamente un layer, esegue il rendering dei Widget al suo interno e applica la maschera bitmap durante il blending.
 - In alternativa, la proprietà `bitmap_mask` nel descrittore di disegno può essere utilizzata direttamente per il disegno dell'immagine.

Utilizzando il Widget Canvas con un buffer `LV_COLOR_FORMAT_L8`, è possibile eseguire il rendering dinamico delle maschere bitmap.

API

`lv_draw.h`
`lv_draw_arc.h`
`lv_draw_image.h`
`lv_draw_label.h`
`lv_draw_line.h`
`lv_draw_mask.h`
`lv_draw_rect.h`
`lv_draw_triangle.h`
`lv_grad.h`

7.10.5 Snapshot

Snapshot fornisce un'API per acquisire un'immagine snapshot [istantanea] per un Widget LVGL insieme ai suoi elementi figlio. L'immagine apparirà esattamente come il Widget sul display.

Utilizzo

È sufficiente chiamare la funzione `lv_snapshot_take(widget, color_format)` per generare il descrittore dell'immagine utilizzabile come sorgente dell'immagine di un Widget Image tramite `lv_image_set_src()`.

Nota: al momento sono supportati solo i seguenti formati colore:

- `LV_COLOR_FORMAT_RGB565`
- `LV_COLOR_FORMAT_ARGB8565`
- `LV_COLOR_FORMAT_RGB888`
- `LV_COLOR_FORMAT_XRGB8888`
- `LV_COLOR_FORMAT_ARGB8888`
- `LV_COLOR_FORMAT_A8`
- `LV_COLOR_FORMAT_L8`
- `LV_COLOR_FORMAT_I1`
- `LV_COLOR_FORMAT_ARGB2222`

- `LV_COLOR_FORMAT_ARGB4444`
- `LV_COLOR_FORMAT_ARGB1555`

Il supporto per i formati colore dipende dall'unità di disegno. Verificare l'unità di disegno.

Liberazione dell'Immagine

La memoria utilizzata da `lv_snapshot_take()` viene allocata dinamicamente tramite `lv_draw_buf_create()`. Si usa `lv_draw_buf_destroy()` per liberare la memoria allocata.

L'immagine snapshot, che è il buffer di disegno restituito da `lv_snapshot_take()` normalmente non verrà aggiunta alla cache perché può essere disegnata direttamente. Quindi non è necessario invalidare la cache chiamando `lv_image_cache_drop()` prima di distruggere il buffer di disegno.

Il frammento di codice seguente mostra l'uso corretto di `lv_snapshot_take()`:

```
void update_snapshot(lv_obj_t * widget, lv_obj_t * img_snapshot)
{
    lv_draw_buf_t* snapshot = (void*)lv_image_get_src(img_snapshot);
    if(snapshot) {
        lv_draw_buf_destroy(snapshot);
    }
    snapshot = lv_snapshot_take(widget, LV_COLOR_FORMAT_ARGB8888);
    lv_image_set_src(img_snapshot, snapshot);
}
```

Utilizzo di un Buffer Esistente

Se lo snapshot deve essere aggiornato ripetutamente o se il chiamante fornisce il buffer di disegno, si usa `lv_snapshot_take_to_draw_buf(widget, color_format, draw_buf)`. In questo caso, il chiamante è responsabile della creazione e della distruzione del buffer di disegno.

Se lo snapshot viene generato correttamente, il descrittore dell'immagine viene aggiornato, i dati dell'immagine vengono memorizzati nel `draw_buf` fornito e la funzione restituisce `LV_RESULT_OK`.

Si noti che lo snapshot potrebbe non funzionare se il buffer fornito non è sufficientemente grande, il che può accadere se le dimensioni del Widget cambiano. Si consiglia di utilizzare `lv_snapshot_reshape_draw_buf(widget, draw_buf)` per assicurarsi innanzitutto che il buffer sia sufficientemente grande e, in caso di errore, eliminare il buffer di disegno esistente e chiamare direttamente `lv_snapshot_take`.

Esempio

Esempio semplice di snapshot

```
#include "../../lv_examples.h"
#if LV_USE_SNAPSHOT && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * snapshot_obj = (lv_obj_t *) lv_event_get_user_data(e);
    lv_obj_t * img = lv_event_get_target_obj(e);

    if(snapshot_obj) {
        lv_draw_buf_t * snapshot = (lv_draw_buf_t *)lv_image_get_src(snapshot_obj);
        if(snapshot) {
            lv_draw_buf_destroy(snapshot);
        }

        /*Update the snapshot, we know parent of object is the container.*/
        snapshot = lv_snapshot_take(lv_obj_get_parent(img), LV_COLOR_FORMAT_ARGB8888);
        if(snapshot == NULL)
            return;
        lv_image_set_src(snapshot_obj, snapshot);
    }
}

void lv_example_snapshot_1(void)
{
    LV_IMAGE_DECLARE(img_star);
    lv_obj_t * root = lv_screen_active();
    lv_obj_set_style_bg_color(root, lv_palette_main(LV_PALETTE_LIGHT_BLUE), 0);

    /*Create an image object to show snapshot*/
    lv_obj_t * snapshot_obj = lv_image_create(root);
    lv_obj_set_style_bg_color(snapshot_obj, lv_palette_main(LV_PALETTE_PURPLE), 0);
    lv_obj_set_style_bg_opa(snapshot_obj, LV_OPA_100, 0);
    lv_image_set_scale(snapshot_obj, 128);
    lv_image_set_rotation(snapshot_obj, 300);

    /*Create the container and its children*/
    lv_obj_t * container = lv_obj_create(root);
```

(continues on next page)

(continua dalla pagina precedente)

```
lv_obj_center(container);
lv_obj_set_size(container, 180, 180);
lv_obj_set_flex_flow(container, LV_FLEX_FLOW_ROW_WRAP);
lv_obj_set_flex_align(container, LV_FLEX_ALIGN_SPACE_EVENLY, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);
lv_obj_set_style_radius(container, 50, 0);
lv_obj_t * img;
int i;
for(i = 0; i < 4; i++) {
    img = lv_image_create(container);
    lv_image_set_src(img, &img_star);
    lv_obj_set_style_bg_color(img, lv_color_black(), 0);
    lv_obj_set_style_bg_opa(img, LV_OPA_COVER, 0);
    // lv_obj_set_style_transform_scale(img, 400, LV_STATE_PRESSED);
    lv_obj_add_flag(img, LV_OBJ_FLAG_CLICKABLE);
    lv_obj_add_event_cb(img, event_cb, LV_EVENT_PRESSED, snapshot_obj);
    lv_obj_add_event_cb(img, event_cb, LV_EVENT_RELEASED, snapshot_obj);
}
#endif
```

API

lv_snapshot.h

CAPITOLO 8

LVGL Pro e XML

8.1 Introduzione

LVGL Pro è un toolkit completo per creare, testare, condividere e distribuire interfacce utente integrate in modo efficiente.

Si compone di 4 strumenti principali:

- **Un editor XML** con potenti funzionalità di auto-completamento per descrivere componenti dell'interfaccia utente, schermate, test, animazioni, data binding e molto altro. Un'anteprima pixel-perfect mostra l'interfaccia utente mentre si digitano i file XML. L'Editor è anche in grado di esportare codice C e persino di ricompilare la sua anteprima per eseguire codice C personalizzato. In queste pagine se ne può saperne di più su [XML](#) e su [Editor](#).
- **Online Share** è una versione online dell'Editor che può caricare i file XML da un repository GitHub per condividere facilmente le interfacce utente sviluppate senza dover configurare alcun ambiente di sviluppo. Se ne parla estesamente [qui](#).
- **CLI** (command-line interface) è uno strumento in grado di convalidare i file XML e generare codice C da essi, oppure di eseguire test dell'interfaccia utente in ambienti CI/CD. Scoprirne di più [qui](#).
- **Plugin Figma** che consente di estrarre le proprietà di stile dagli elementi Figma e di sincronizzarle nell'Editor o con la CLI. Scoprirne di più [qui](#).

Questi strumenti insieme consentono agli sviluppatori di creare interfacce utente efficaci, testarle in modo semplice e affidabile e collaborare con il resto del team e i clienti.

Questa sezione della documentazione illustra tutti i dettagli di ogni strumento.

8.2 Imparare dagli Esempi

Abbiamo preparato diversi tutorial su https://github.com/lvgl/lvgl_editor/tree/master/tutorials.

Con <https://viewer.lvgl.io/> si possono provare facilmente direttamente nel browser.

1. **Hello World** Come creare un pulsante con stile e un'etichetta in XML.
2. **New Component** Crea nuovi componenti e una schermata a partire da essi.
3. **Assets** Come usare immagini e font.
4. **Screens** Prova gli eventi di caricamento e creazione delle schermate.

5. **Layouts** I layout flex annidati in azione.
6. **Data Binding** La potenza dei data binding.
7. **Translations** Come tradurre i testi e visualizzarli in anteprima.
8. **Animations** Crea animazioni basate sulla timeline e le riproduce al clic del pulsante.

8.3 Editor

8.3.1 Panoramica

Introduzione

L'Editor di LVGL Pro è un editor XML per semplificare e velocizzare la creazione di interfacce utente embedded.

Il suo cuore è un potente editor XML con un'anteprima pixel-perfect in real-time. Oltre a supportare la creazione e l'anteprima di componenti e schermate, supporta anche la descrizione di test dell'interfaccia utente, animazioni, traduzioni e praticamente tutto ciò che è definito in XML.

Dai file XML, l'Editor può esportare codice C che può essere incorporato nel firmware proprio come qualsiasi codice C scritto a mano.

L'Editor può anche ricompilare la sua anteprima da LVGL e widget personalizzati scritti in C.

Questa sezione della documentazione descrive come installare e utilizzare l'Editor. Per saperne di più sul formato XML stesso, visitare [questa pagina](#).

Vantaggi

Utilizzare l'Editor di LVGL Pro invece di scrivere XML a mano offre diversi vantaggi:

- visualizzare i componenti all'istante nell'anteprima,
- utilizzare uno strumento di ispezione per visualizzare le dimensioni dei widget, i padding, ecc.,
- il completamento automatico velocizza la scrittura dell'XML,
- accelerare la reimplementazione dei design Figma con l'aiuto di un *plugin Figma*,
- supportare la scrittura e l'esecuzione di *test della UI* in pochi secondi,
- supportare l'adattamento del soggetto per testare i *Data binding*.

L'Editor può essere eseguito come:

- applicazione desktop autonoma su Windows, Linux e macOS;
- estensione VSCode; e
- online (tramite un browser web).

8.3.2 Installazione

Ecco le guide per scaricare e installare l'Editor LVGL Pro adatto piattaforma in uso:

Windows

Installare WSL

Prima di utilizzare l'Editor, è necessario installare WSL (Windows Subsystem for Linux).

Aprire un Terminale e digitare `wsl.exe --list --verbose` per verificare se è già installato.

In caso contrario, digitare `wsl --install` e premere "Enter".

Installare l'Editor

Accedere a <https://pro.lvgl.io/#download>, premere il pulsante [Download] e salvare l'immagine risultante sulla propria workstation. Quindi si esegue il programma di installazione scaricato per installare LVGL Pro Editor.

Linux

Installare Podman

L'Editor è in grado di ricompilare l'anteprima per utilizzarla ed eseguire anche il codice C. Gestiamo la compilazione in un container usando **Podman** (simile a Docker).

Il modo più semplice per installare Podman è tramite il Terminale:

- **Debian, Ubuntu, Linux Mint:** sudo apt-get install podman
- **Fedora:** sudo dnf -y install podman
- **Arch, Manjaro Linux:** sudo pacman -S podman

Per altre distribuzioni, consultare <https://podman.io/docs/installation#installing-on-linux>

Installare l'Editor

L'installazione dell'Editor LVGL Pro su Linux è semplice come scaricare l'AppImage, renderla eseguibile (chmod +x LVGL_Pro_Editor.AppImage) ed eseguirla.

Risoluzione dei problemi

Problema

Se si fa doppio clic sul file .AppImage e non succede nulla, provare ad eseguirlo dal terminale. Se si vede l'errore:

```
(process:3943): Gtk-ERROR **: 11:45:01.301: GTK 2/3 symbols detected.
Using GTK 2/3 and GTK 4 in the same process is not supported
Trace/breakpoint trap (core dumped)
```

Soluzione

Specificare la versione GTK per l'applicazione con --gtk-version=3.

```
./path/to/editor.AppImage --gtk-version=3
```

MacOS

Installare Podman

L'Editor è in grado di ricompilare l'anteprima per utilizzarla ed eseguire anche il codice C. Gestiamo la compilazione in un container usando **Podman** (simile a Docker). Il modo più semplice per installare Podman è tramite brew usando brew install podman.

Per maggiori informazioni visitare <https://podman.io/docs/installation#macos>

Installare l'Editor

Basta scaricare la versione ARM (per Apple Silicon) o AMD e trascinare l'applicazione nella cartella "Applications" del Finder.

8.3.3 Interfaccia Utente

Prima Esecuzione

Al primo avvio di LVGL Pro, verrà visualizzata una finestra con il messaggio

"Pulling Container image for LVGL, this may take a few minutes...".

Questo messaggio indica che l'editor sta scaricando un'immagine del container preconfigurata che include tutti gli strumenti e le dipendenze necessari per la creazione di progetti LVGL. LVGL Pro gestisce (e aggiorna) il container automaticamente in background. Il processo può richiedere alcuni minuti la prima volta, a seconda della velocità della connessione Internet, ma avviene una sola volta. Dopodiché, il container viene riutilizzato per le sessioni future.

Quando viene pubblicata una nuova versione del container (ad esempio con toolchain aggiornate, logica di compilazione, ecc.), l'editor ne verifica automaticamente la presenza e scarica la versione più recente. Non sarà necessario eseguire alcuna operazione manualmente, poiché l'aggiornamento avviene in background.

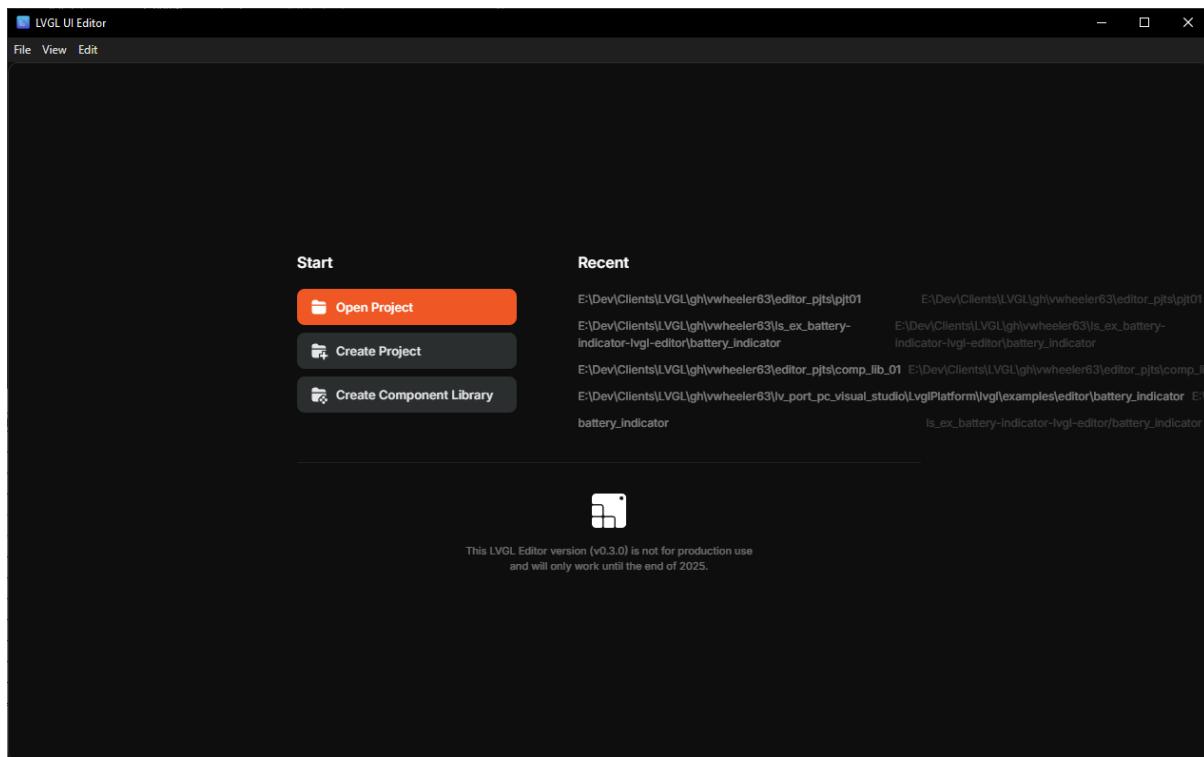
Login

1. inserire il proprio indirizzo email
2. si riceverà un'email con una OTP (One-Time Password)
3. verrà creato un account utente (non è necessaria alcuna password)

Selezione della Licenza

Dopo aver effettuato l'accesso, è necessario selezionare una licenza dall'elenco.

Screen di Avvio



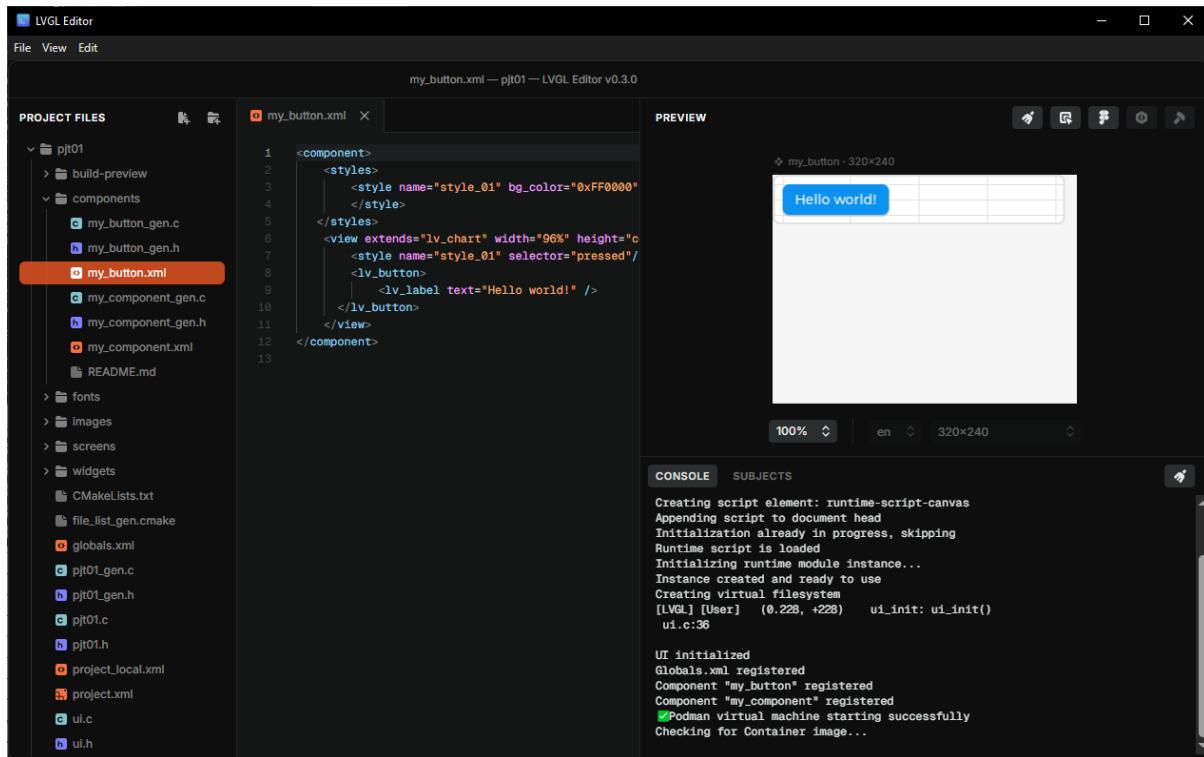
LVGL Pro Interfaccia Utente Iniziale

La "Launch Screen" [schermata di avvio] si apre ogni volta che si avvia l'Editor e si è già effettuato l'accesso.

Qui si può:

- Aprire un progetto esistente
- Creare un nuovo progetto
- Aprire un progetto recente

Lavorare su un Progetto



LVGL Pro Interfaccia Utente

Menù

LVGL Pro presenta il consueto menù File / View / Edit / ecc.

Pannello "Project Files"

Il pannello di sinistra è un tipico pannello di navigazione "project-tree". Consente di creare, aprire e rinominare file e gestire il progetto in modo efficiente e pratico.

Le sottodirectory visualizzate nel progetto sono per i rispettivi tipi di componenti dell'interfaccia utente (ad esempio Screen, Widget, Componenti, Font e Immagini).

Pannello "Editor"

Il Pannello Editor appare al centro della finestra. È qui che LVGL Pro svolge il suo prezioso lavoro, creando l'interfaccia utente o la Libreria di Componenti. È qui che si può osservare e modificare il contenuto di qualsiasi file nel progetto, ma la sua specialità è la modifica dell'XML che definisce i componenti dell'interfaccia utente ([I Widget](#), [Componenti](#) e [Screen](#)). Durante la modifica di XML, LVGL Pro dispone di una potente interfaccia utente con auto-completamento, che semplifica la definizione, la modifica e la convalida dei componenti dell'interfaccia utente. Simile all'assistenza contestuale disponibile in altri "editor intelligenti", il riquadro dell'editor assiste l'utente praticamente in ogni battitura, riducendo notevolmente la curva di apprendimento necessaria per produrre interfacce utente complete e di qualità professionale con LVGL Pro.

Pannello Destro

Pannello di "Preview" e "Inspector"

Quando il "Pannello Editor" modifica un file XML che definisce un Componente o un Widget, il "Pannello Preview" mostra un'anteprima in tempo reale della definizione dell'Interfaccia Utente XML in fase di modifica, sottoposta a *rendering LVGL reale*, in modo da poter visualizzare immediatamente i risultati del lavoro, che corrisponderanno al risultato finale prodotto dal firmware, con una precisione al pixel.

Allo stesso tempo, consente di interagire con l'interfaccia utente per vedere come apparirà e come funzionerà in fase di esecuzione.

Come Funziona:

- LVGL ha un parser XML integrato che può creare interfacce utente a runtime
- l'inspector utilizza LVGL così com'è
- l'editor invia i file XML a LVGL
- l'editor chiede a LVGL di creare un'istanza del file XML aperto
- quando l'XML cambia, viene caricato solo il file XML aperto, lo schermo viene cancellato e viene creata una nuova istanza
- l'editor XML analizza i file XML dei widget LVGL e i file XML dell'utente per convalidare gli XML
- il codice viene esportato anche analizzando i file XML

Trasforma il "Pannello Preview" in un "Inspector" tenendo premuto il tasto [Ctrl]. L'Inspector consente di:

- visualizzare il "bounding box", il padding, i margini e l'area di clic
- controllare le dimensioni e la posizione col trascinamento
- eseguire la misurazione
- fare doppio clic per passare al file XML
- fare un clic singolo per spostare il focus dell'editor sull'elemento XML corrispondente

Pannello "Output"

Il Pannello Output contiene diversi "tab" [schede] in cui è possibile visualizzare vari risultati e gestire determinati elenchi come parte del progetto.

Scheda Console

contiene il "log dell'output" delle azioni interne intraprese e dei relativi risultati.

Scheda Subjects

contiene l'elenco dei argomenti Subject Observer-Pattern implementati nei file sorgente XML. Ogni Subject rappresenta un singolo "dato" nell'applicazione, che gli elementi dell'Interfaccia Utente (Widget e Componenti) possono "osservare" e riflettere visivamente. Vedere [Data binding \(I Subject\)](#) per maggiori dettagli.

Scheda Test

TODO: Descrizione della Scheda Test

Schede in Arrivo: Test, Traduzioni, Animazioni

8.3.4 Hotkey

Attiva il Completamento Automatico

Mentre il cursore si trova in un punto qualsiasi del *Pannello "Editor"*:

Ctrl+Spazio

Pop-up della lista di auto-completamento

Modifica del Valore

Mentre il cursore si trova all'interno del valore di un attributo:

Ctrl+Su

Ctrl+Giù

Modifica il valore di 1 (con aggiornamento in tempo reale nell'anteprima)

Ctrl+Shift+Su**Ctrl+Shift+Giù**

Modifica il valore di 10 (con aggiornamento in tempo reale nell'anteprima)

Espora e Build**Ctrl+E**

Espora solo codice

Ctrl+B

Espora codice e ricompila

Ctrl+Shift+B

Pulisce e ricompila

Riquadro Anteprima**Ctrl+rotellina del mouse**

Zoom in e out

Ctrl++

Zoom in

Ctrl+-

Zoom out

Ctrl+0

Resetta lo zoom

Ctrl+Shift+0

Adatta a schermo

Scorrimento con due dita

Anteprima pan

Premendo il tasto Ctrl è inoltre possibile:

- visualizzare il "bounding box", la spaziatura interna, i margini e l'area di clic,
- controllare le dimensioni e la posizione trascinando,
- eseguire la misurazione,
- fare doppio clic per passare al file XML,
- fare un clic singolo per spostare il focus dell'editor sull'elemento XML corrispondente.

8.3.5 Licenza

LVGL Pro Editor è dotato di un modello di licenza semplice, scalabile e royalty-free, adatto a progetti open source, startup e aziende. [Confrontare tutti i piani](#).

8.4 Panoramica XML

8.4.1 Panoramica

Introduzione

Il modulo XML di LVGL implementa l'interfaccia utente dichiarativa di LVGL consentendo di descrivere le interfacce utente in XML.

Utilizzare XML invece della semplice modifica tramite trascinamento della selezione offre diversi vantaggi:

- **Sintassi familiare** funziona come HTML, facile da imparare e leggere
- **Controllo di versione intuitivo** testo semplice con differenze leggibili, senza file binari

- **Facile da condividere** copiare, incollare e inviare come testo
- **Pattern riutilizzabili** copiare e riutilizzare frammenti tra progetti
- **Automatizzabile** script e CI/CD possono elaborare XML senza sforzo
- **Compatibile con l'AI** gli strumenti di intelligenza artificiale possono leggere, generare e rifattorizzare XML
- **Nativamente modulare** creare componenti riutilizzabili per interfacce utente strutturate
- **Veloce da modificare** più veloce da digitare rispetto al trascinamento della selezione con il mouse
- **Caricamento a runtime** elaborare l'XML in fase di esecuzione senza ricompilazione
- **Multipiattaforma** lo stesso XML funziona su tutti i target LVGL

Il file XML può essere scritto a mano, ma si consiglia vivamente di utilizzare l'`Editor di LVGL Pro <editor>` per scrivere i file XML. L'Editor velocizza notevolmente lo sviluppo dell'interfaccia utente offrendo funzionalità quali:

- Anteprima istantanea dei file XML (componenti, screen)
- Modalità Inspector per visualizzare le dimensioni dei widget, i padding, ecc.
- Generazione di codice C da file XML
- Completamento automatico ed evidenziazione della sintassi
- Condivisione/anteprima online per collaborazione e test
- Integrazione con [Figma](#) per reimplementare facilmente i design Figma

Elementi dell'interfaccia utente

È importante distinguere tra *Widget*, *Componenti* e *Screen* [schermate].

I Widget

I *Widget* sono i componenti fondamentali dell'interfaccia utente e **non sono pensati per essere caricati in fase di esecuzione**, ma piuttosto compilati nell'applicazione come codice C. Le caratteristiche principali dei Widget sono:

- In XML, iniziano con un elemento radice `<widget>`.
- Sono simili ai widget nativi di LVGL.
- Possono contenere logica personalizzata e complessa.
- Non possono essere caricati da XML a runtime perché il codice custom non può essere caricato.
- Possono avere un'API ampia e personalizzata con funzioni `set/get/add`.
- È necessario un parser XML personalizzato per mappare le proprietà XML alle chiamate alle funzioni API.
- Possono essere composti, ovvero contenere altri widget (ad esempio, i tab delle Tabview, le liste delle Dropdown).
- Nell'editor dell'interfaccia utente, possono anche essere descritti in XML per creare elementi visivi più rapidamente ed esportare codice C.

Componenti

I *Components* sono elementi dell'interfaccia utente più semplici e **possono essere caricati a runtime**. Le caratteristiche principali dei componenti sono:

- In XML, iniziano con un elemento radice `<component>`.
- Sono costruiti a partire da Widget e/o altri Componenti.
- Sono definiti solo in XML e non possono avere codice C personalizzato (ma possono chiamare funzioni C come callback di eventi).
- Possono essere caricati da XML poiché contengono solo XML.

- Possono essere utilizzati per definire lo stile di Widget e altri Componenti, utilizzando associazioni di dati tramite argomenti, definendo animazioni, ecc.
- Come elementi figlio, possono contenere Widget e/o altri Componenti.
- Possono avere una semplice API per passare proprietà ai loro elementi figlio (ad esempio, `btn_text` al testo di una Label).

Che l'XML sia stato scritto manualmente o dall'UI Editor, i file che definiscono i Componenti possono essere registrati in LVGL e, successivamente, possono essere create istanze.

In altre parole, LVGL può leggere i file XML, "apprendere" i Componenti da essi e, successivamente, creare elementi figlio come parte di Screen e altri Componenti.

Screen

Gli *Screen* [schermate] sono simili ai Componenti:

- In XML, iniziano con un elemento radice `<screen>`.
- Sono costruiti a partire da Widget e/o altri Componenti per descrivere lo *Screen*.
- Possono essere caricati da XML a runtime poiché descrivono solo gli aspetti visivi dell'interfaccia utente.
- Non dispongono di un'API.
- È possibile farvi riferimento negli eventi di caricamento dello screen.

Dati globali

`globals.xml` è uno speciale file XML in cui sono disponibili

- stili
- costanti
- images
- font
- soggetti per i data binding.

da poter definire.

È possibile caricare più file `globals.xml`, ma ognuno verrà salvato nello stesso "scope" globale, il che significa che gli elementi duplicati verranno aggiunti una sola volta.

Utilizzo dei File XML

I file XML possono essere utilizzati in due modi:

1. Caricandoli direttamente in fase di esecuzione
2. Convertendoli in codice C utilizzando l'editor o la CLI

Entrambi i casi d'uso sono trattati in dettaglio nella sezione [Integrazione](#).

8.4.2 Sintassi

Convenzioni sui nomi

- Viene utilizzata una sintassi XML standard.
- Per i nomi degli attributi vengono utilizzate lettere minuscole separate da `_`.
- I nomi dei tag seguono le consuete regole per i nomi delle variabili: devono iniziare con una lettera o con '`_`' e il resto del nome può essere composto da lettere, '`_`' e cifre.
- L'API LVGL viene seguita il più possibile, ad esempio `align="center"`, `bg_color="0xff0000"`.

- Per i colori, sono supportate tutte queste sintassi (simili ai colori CSS): 0x112233, #112233, 112233, 0x123, #123, 123. Si noti che, come in CSS, 0x123, #123 e 123 significano tutti #112233.
- `params` può essere referenziato con \$.
- `consts` può essere referenziato con #.
- `styles` può essere associato a stati e/o parti in questo modo: `styles="style1 style2:pressed style3:focused:scrollbar"`.
- Gli stili locali (ovvero gli stili memorizzati all'interno del Componente e quindi non condivisi da altri Componenti) possono essere utilizzati in questo modo: `<lv_label style_text_color="0xff0000" style_text_color:checked="0x00ff00"/>`.

Tipi

Tutti i tipi possono essere utilizzati come tipi di proprietà API, ma solo un sottoinsieme di essi può essere utilizzato come costante e come tipi *Subject*.

Tipi semplici

Sono supportati i seguenti tipi semplici nativi:

bool

`true` o `false`.

int

Numero intero compreso tra circa -2B e 2B per default. (Uguale a `int32_t` in C.)

px

Semplici unità di pixel. L'unità px può essere omessa.

%

Percentuale. % deve essere aggiunto al valore come unità. (Ha lo stesso significato di `lv_pct()`.)

content

Significa `LV_SIZE_CONTENT`.

string

Semplice stringa terminata da \0. Quando si utilizzano più stringhe in una proprietà o in un array di stringhe, si dovrebbe usare ' [apice]. Es. `foo='a' 'b'`.

colore

Un colore memorizzato come RGB a 24 bit (`lv_color_t`).

opa

Valore di opacità compreso tra 0 e 255 o tra 0 e 100%. Come in CSS, i valori percentuali devono essere seguiti da '%'.

lv_obj

Puntatore a un Widget (`lv_obj_t*`).

screen

Puntatore a una schermata (anche `lv_obj_t*`).

time_ms

Indica un intervallo di tempo in millisecondi.

deg_0.1

Gradi con risoluzione 0,1.

scale_1/256

Scala/Zoom, dove 256 indica il 100%, 128 il 50%, 512 il 200% ecc.

Tipi basati sui nomi

Nei file XML, font, immagini, stili, ecc. non vengono utilizzati tramite puntatori, ma tramite nomi di stringa. Ad esempio, uno stile è definito come `<style name="red" bg_color="0xff0000"/>`. In seguito, è possibile fare riferimento a essi tramite i loro nomi.

Ciò significa che i valori effettivi devono essere associati ai nomi quando l'interfaccia utente viene caricata da XML, altrimenti LVGL non saprà cosa significa un nome.

La maggior parte di queste connessioni viene eseguita automaticamente (ad esempio, per stili, font, immagini, animazioni, gradienti, ecc.) quando vengono registrati i componenti e i `globals.xml`.

Tuttavia, altre devono essere collegate manualmente. Ad esempio, le callback degli eventi, in cui la callback stessa è disponibile solo nel codice, oppure font o immagini se fanno riferimento a dati memorizzati in memoria.

Per saperne di più su come registrare le coppie nome-dati, visitare [Caricamento di File XML a Runtime](#).

Array

Un array di qualsiasi tipo può essere definito in tre modi:

`int[count]`

Un array di interi. Il numero di elementi verrà passato come parametro separato.

`string[NULL]`

Un array che termina con un elemento NULL.

`string[]`

Nessuna terminazione NULL e nessun parametro di conteggio. Utilizzata quando il numero di elementi non è noto o è fornito tramite un altro meccanismo, ad esempio tramite un'altra impostazione.

Enum

`<enumdef>` può essere utilizzato nei tag `<api>` per creare enumerazioni personalizzate per i **Widgets**. Questa funzionalità non è supportata per i Componenti.

Ad esempio:

```
<api>
  <enumdef name="my_widget_mode" help="Possible modes" help-zh="Chinese help">
    <enum name="normal" help="Normal mode" help-zh="Normal mode in Chinese" value="0x10"/>
    <enum name="inverted" help="Inverted mode"/>
  </enumdef>

  <prop name="mode" help="help" type="enum:my_widget_mode" help="help"/>
</api>
```

Se utilizzato come tipo, il suffisso + consente di selezionare più valori e di combinarli con OR. Ad esempio: `type="axis+"`. In questo caso, le opzioni devono essere separate da |, per esempio: `axis=primary_x|secondary_y`.

Tipi composti

I tipi possono essere composti, ovvero sono possibili più opzioni/tipi. Ad esempio, per width: `type="px|%|content"`.

Limitazione dei valori accettati

Non ancora supportato; gli esempi seguenti illustrano la sintassi pianificata.

È anche possibile limitare le possibili opzioni che l'utente può selezionare da un enum. Ad esempio:

- Enum: `type="dir(top bottom)"`
- Colori: `type="color(0xff0000 0x00ff00 0x0000ff)"`
- Stringhe: `type="string('Ok' 'Cancel')"`

8.4.3 Licenza XML

Versione 1.0 – Giugno 2025

Copyright (c) 2025 LVGL LLC

1. Panoramica

La Specifica XML LVGL (la "Specifica") definisce un linguaggio basato su XML per la descrizione delle interfacce utente all'interno dell'ecosistema LVGL.

Questa Specifica è protetta da copyright. È fornita con diritti d'uso limitati come descritto di seguito.

Lo scopo di questa licenza è garantire che la Specifica rimanga liberamente utilizzabile dagli sviluppatori di applicazioni che lavorano con LVGL, limitandone al contempo l'utilizzo in editor di interfaccia utente o strumenti di progettazione di terze parti. Ciò protegge l'integrità e la sostenibilità a lungo termine dell'ecosistema LVGL impedendo a strumenti senza licenza di riutilizzare o replicare la Specifica in modi che competono con i prodotti LVGL ufficiali.

2. Utilizzo Consentito

È consentito:

- Utilizzare questa Specifica in **qualsiasi firmware o applicazione embedded** basata sulla libreria ufficiale **LVGL** (MIT-licensed).
- Utilizzare liberamente il loader XML LVGL in conformità con la sua licenza MIT.
- Utilizzare la Specifica con l'**editor UI LVGL** e gli **strumenti LVGL ufficiali come la CLI o il loader XML**.
- Scrivere, caricare, modificare o generare interfacce utente descritte nella Specifica XML LVGL solo per **uso interno**, inclusi gli strumenti di automazione o configurazione utilizzati esclusivamente all'interno della propria organizzazione.
- Creare **script o plugin interni** che aiutino i propri sviluppatori a utilizzare o gestire questa Specifica, a condizione che non vengano **condivisi all'esterno della propria organizzazione o resi pubblici**.
- Condividere file di interfaccia utente basati su XML all'interno del firmware del cliente o di progetti destinati a sistemi embedded basati su LVGL.

3. Restrizioni

Non è consentito:

- Creare, rilasciare o distribuire **editor di interfaccia utente, visual builder, layout designer, generatore di codice o strumento** che legga, scriva o interpreti file XML utilizzando questa Specifica, sia commerciale, open source o destinato all'**uso pubblico, all'uso da parte di clienti o partner, o per qualsiasi uso esterno alla propria organizzazione**, senza l'autorizzazione scritta di LVGL LLC.
- Implementare o estendere questa Specifica in qualsiasi **software o piattaforma che fornisca funzionalità di creazione, modifica o progettazione di interfacce utente simili a quelle dell'LVGL UI Editor**.
- Creare API, plugin, convertitori o SDK pubblici basati su questa Specifica.
- Condividere o pubblicare strumenti interni basati su questa Specifica.
- Utilizzare questa Specifica in strumenti o framework di progettazione generici al di fuori dell'ecosistema LVGL.

Queste restrizioni si applicano indipendentemente dal fatto che il software sia commerciale, gratuito o open source.

4. Chiarimento per gli utenti LVGL

È esplicitamente consentito:

- Utilizzare file XML con questa Specifica nei propri progetti.
- Caricare file XML in fase di esecuzione utilizzando il caricatore XML di LVGL.

- Generare file XML utilizzando l'LVGL UI editor.
- Personalizzare i file XML manualmente o con utilità interne.
- Condividere progetti basati su LVGL con file XML come parte di un prodotto commerciale o di un firmware.

Se si sviluppa software embedded utilizzando LVGL, **questa licenza non ha alcun effetto**. Questa licenza si applica solo agli **strumenti di sviluppo dell'interfaccia utente che leggono, generano o elaborano file XML utilizzando questa Specifica per scopi di sviluppo dell'interfaccia utente** in modo pubblico o commerciale.

È possibile creare **strumenti interni** per facilitare lo sviluppo utilizzando la Specifica, ma questi non devono essere pubblicati, venduti o condivisi esternamente senza una licenza.

5. Marchi

Questa licenza non concede alcun diritto di utilizzo del nome, del logo o dei marchi LVGL.

6. Licenze e Autorizzazioni

Per richiedere l'autorizzazione per un uso più ampio o una licenza commerciale (ad esempio, l'utilizzo di questa Specifica nel proprio strumento di interfaccia utente), contattare:

lvgl@lvgl.io

Tutti i diritti non espressamente concessi nel presente documento sono riservati a LVGL LLC.

8.5 Integrazione

8.5.1 Utilizzo del codice C esportato

Panoramica

Quando il codice C viene esportato dall'Editor dell'interfaccia utente di LVGL, non è necessario alcun codice XML (`LV_USE_XML` può essere 0), ed è molto simile al caso in cui il codice C viene scritto a mano.

Il codice C esportato può essere semplicemente inserito nel progetto e compilato nell'applicazione.

Inizializzazione

Il punto di ingresso principale è una funzione `<project_name>_init("asset_path")` situata in `<project_name>.c`. `<project_name>` è il nome di default della cartella padre, che può essere sovrascritto in `project.xml` come `<project name="my_project">`.

Il suo unico parametro è un path in cui si trovano le risorse basate su file (font e immagini). Ad esempio, se un file PNG e TTF si trova in "A:my_ui/v2.3/assets", deve essere passato a `ui_init()` in modo che tutti i percorsi utilizzati possano essere prefissati. Ciò significa che se `images/logo.png` è stato utilizzato nell'editor dell'interfaccia utente, il path completo sarà "A:my_ui/v2.3/assets/images/logo.png".

Si noti che la funzione "init" crea solo gli *screen permanenti* ma non carica alcuno screen. Tuttavia, questo può essere fatto facilmente chiamando `lv_screen_load(screen1)`.

Gli "Screen" possono anche essere creati facilmente chiamando la funzione "create" generata, ad esempio `main_screen_create()`.

Struttura del Codice Esportato

L'esportazione del codice segue alcune convenzioni.

I file generati terminano con `_gen.c` o `_gen.h`. Questi file non devono essere modificati poiché verranno sovrascritti alla successiva generazione di codice.

Per i file non generati, viene creato solo uno scheletro (se il file non esiste). Qui, l'utente può aggiungere codice personalizzato in base alle proprie esigenze.

Dai file XML di schermate e componenti, vengono generate funzioni "create" utilizzando il nome del file. Ad esempio:

- `lv_obj_t * main_screen_create(void)` da `main_screen.xml`
- `lv_obj_t * my_button_create(lv_obj_t * parent, const char * button_text)` da `my_button.xml`

Per i componenti, non vengono creati "setter", ma tutti i parametri vengono passati alla funzione "create".

Da `globals.xml`, vengono creati quattro file:

- File C e H generati contenenti tutte le inizializzazioni di oggetti, font, immagini, stili e altre.
- File C e H non generati che sono solo wrapper di quelli generati in cui è possibile inserire codice personalizzato.

Il nome di questi file è quello della cartella che contiene il file `globals.xml` per default. Può essere sovrascritto da `<globals name="my_lib">`.

Aggiunta di Codice Personalizzato

In alcuni casi, potrebbe essere necessario estendere il codice generato con codice scritto a mano. Vediamo come farlo in casi diversi.

Aggiunta di Codice Generico

`lv_timer`, un oggetto con un "observer" speciale, animazioni complesse o stili può essere aggiunto in `ui.c` o nel file C creato da `globals.xml`.

Ad esempio, per aggiungere alcune proprietà a uno stile definito in XML:

```
void ui_init(const char * asset_path)
{
    LV_LOG_USER("ui_init()\n");
    ui_lib_init(asset_path);

    lv_style_set_bg_color(&style_1, get_special_color());
}
```

Per creare un oggetto speciale che formatti un valore di temperatura intero come stringa con un punto decimale:

```
/* Assume that subject_temperature_int stores the value with 0.1 degree resolution */
/* E.g. 345 means 34.5 */

void temperature_to_string_observer_cb(lv_observer_t * observer,
                                       lv_subject_t * subject)
{
    int32_t upscaled_value = lv_subject_get_int(&subject_temperature_int);

    int32_t int_part = upscaled_value / 10;
    int32_t fract_part = upscaled_value % 10;

    char buf[64];
    lv_snprintf(buf, sizeof(buf), "%d.%d", int_part, fract_part);
    lv_subject_copy_string(&subject_temperature_string, buf);
}

lv_subject_add_observer(&subject_temperature_int,
                        temperature_to_string_observer_cb, NULL);
```

Wrapper del componente

Se un componente deve essere esteso con una funzionalità o un'API non supportata dall'editor, la soluzione più semplice è creare un componente wrapper.

Ad esempio, supponiamo che `my_button.xml` descriva alcune funzionalità di un pulsante, ma che sia necessario aggiungere alcuni setter, API speciali o funzionalità extra. In questo caso, basta creare manualmente nuovi file C e H per un `super_button` con una funzione come:

```
lv_obj_t * super_button_create(lv_obj_t * parent, ...some_arguments...)
{
    lv_obj_t * my_button = my_button_create(parent, ...);

    /* ...do something with the created 'my_button' */

    return my_button;
}
```

Integrazione con CMake

Viene generato un scheletro di file `CMakeLists.txt` facilmente integrabile in qualsiasi progetto.

Viene generato un file chiamato `file_list_gen.cmake` che viene utilizzato da `CMakeLists.txt`. Questo garantisce che vengano inclusi solo i file appropriati.

8.5.2 Caricamento di File XML a Runtime

Panoramica

LVGL è in grado di caricare file XML in fase di esecuzione senza ricompilare l'applicazione.

I file XML possono essere inviati al dispositivo tramite Bluetooth, Wi-Fi, porta seriale, caricati da una scheda SD, scaricati da un server o in qualsiasi altro modo.

I file XML devono essere registrati affinché LVGL possa analizzarne il contenuto. È come salvare il progetto dei componenti.

Dopodiché, è possibile creare istanze dei Componenti e degli Screen [schermate] registrati.

Si noti che i *Widgets in XML* non possono essere caricati in fase di esecuzione, poiché i widget vengono creati utilizzando codice C.

Registrazione di File XML

Registrazione Manuale

Si usa `lv_xml_register_component_from_data(name, xml_string)` and `lv_xml_register_component_from_file("A:path/to/my.xml")` per registrare i file XML che definiscono Componenti, Screen e `globals.xml` come stringhe o percorsi di file.

Di conseguenza, LVGL conoscerà tutte le `<view>`, gli `<style>`, le `<const>`, ecc., ovvero il contenuto completo di questi file XML necessario per creare poi delle istanze.

Allo stesso modo, le traduzioni possono essere registrate tramite `lv_xml_register_translation_from_data()` e `lv_xml_register_translation_from_file()`.

Font e immagini vengono registrati automaticamente quando `globals.xml` viene registrato.

Registrazione Batch da una Cartella

Invece di chiamare le funzioni di registrazione una alla volta, è anche possibile caricare più file XML e risorse contemporaneamente.

`lv_xml_load_all_from_path("A:path/to/dir")` attraverserà una directory e registrerà tutti i componenti XML, le schermate, le variabili globali e le traduzioni.

Registrazione da un BLOB

Sono disponibili alcune funzioni aggiuntive di caricamento XML quando si utilizza [FrogFS](#).

Con l'aiuto di FrogFS, è possibile creare un "blob" (file binario) da una cartella contenente tutti i file XML, le immagini e i font. Avere un singolo blob è più facile da gestire rispetto a molti file più piccoli.

Se il blob viene salvato in una memoria mappata su una memoria indirizzabile (flash, RAM, ecc.), si usa `lv_xml_load_all_from_data`:

```
extern const unsigned char my_blob[];
extern unsigned int my_blob_len;
lv_xml_load_t * handle = lv_xml_load_all_from_data(my_blob, my_blob_len);
if(handle == NULL) {
    LV_LOG_USER("error occurred while loading XML content from data");
}
/* 'handle' can optionally be passed to `lv_xml_unload` later */
```

Se il blob viene salvato come file (ad esempio, su una scheda SD), si usa `lv_xml_load_all_from_file`:

```
lv_xml_load_t * handle = lv_xml_load_all_from_file("A:/path/to/frogfs.bin");
if(handle == NULL) {
    LV_LOG_USER("An error occurred while loading XML content from a file");
}
/* 'handle' can optionally be passed to 'lv_xml_unload' later */
```

Registrazione di Dati Esterni

L'unica cosa che non può essere appresa dai file XML sono i dati memorizzati nella flash dell'applicazione. Tra questi rientrano immagini e font memorizzati in memoria, argomenti e costanti forniti dall'applicazione e, soprattutto, le callback degli eventi.

Per collegare questi dati al mondo XML, LVGL fornisce numerose funzioni `lv_xml_register_...` per registrare:

- eventi
- costanti
- timeline delle animazioni
- subject
- images
- font

Ad esempio, si usa `lv_xml_register_event_cb(scope, "event_cb_name", the_callback)` per collegare una callback a un nome. Dopodiché, i dati registrati saranno disponibili per nome nell'applicazione.

`scope` è solitamente `NULL` per registrare le risorse a livello globale. `lv_xml_component_get_scope(component_name)` restituisce un puntatore a uno "scope descriptor" utilizzabile per registrare i dati disponibili solo per un determinato componente.

Oltre agli eventi, è comune registrare immagini e font memorizzati nel codice dell'applicazione tramite `lv_xml_register_image(scope, "image_name", path_or_pointer)` e `lv_xml_register_font(scope, "font_name", path_or_pointer)`.

Registrare Widget

Sebbene i widget siano compilati nell'applicazione, devono essere registrati affinché il parser XML sappia come creare un elemento come `<lv_slider>` quando appare in XML.

Si usa `lv_xml_register_widget("widget_name", create_cb, apply_cb)` a tale scopo.

Per saperne di più sui widget in XML e sulle callback, fare riferimento a *I Widget*.

Si noti che i widget nativi di LVGL vengono registrati automaticamente.

Creazione di Istanze

Creazione di Screen

Per default, non viene creato alcun Widget o Screen. L'utente deve creare gli *Screen Permanenti* e qualsiasi altro Screen richiesto per iniziare.

Si usa `lv_xml_create_screen("name")`, dove "name" è il nome del file XML o quello utilizzato al momento della registrazione dei dati XML.

Restituisce un `lv_obj_t *` caricabile come qualsiasi *Screen* normale utilizzando `lv_screen_load()`.

Creazione di Componenti

Si usa `lv_xml_create(parent, "name", attributes)` per creare qualsiasi Widget, Componente o Screen in fase di esecuzione a partire dagli XML registrati.

`attributes` è un array di coppie nome-valore di proprietà, terminato da `NULL, NULL`.

Ad esempio, per creare un widget:

```
const char * attrs[] = {
    "width", "100",
    "value", "35",
    NULL, NULL
};

lv_obj_t * slider_1 = lv_xml_create(lv_screen_active(), "lv_slider", attrs);
```

Oppure un componente "custom":

```
const char * attrs[] = {
    "width", "100",
    "button_label", "Hello!",
    "color", "#00ff0000",
    NULL, NULL
};

lv_obj_t * my_button_1 = lv_xml_create(lv_screen_active(), "my_button", attrs);
```

Oppure per creare elementi figlio non-lv_obj_t:

```
const char * attrs[] = {
    "color", "#00ff0000",
    "axis", "primary_y",
    NULL, NULL
};

lv_chart_series_t * ser_1 = lv_xml_create(chart1, "lv_chart-series", attrs);
```

Oppure per aggiungere stili:

```
const char * attrs[] = {
    "name", "style1",
    "selector", "knob|pressed",
    NULL, NULL
};

lv_xml_create(button1, "style", attrs);
```

L'Intero flusso

Per caricare correttamente tutto in fase di esecuzione, i passaggi devono essere eseguiti in questo ordine:

- 1. Registrare i widget personalizzati.** Questi sono indipendenti dall'XML, ma i componenti XML si basano su di essi. I widget nativi vengono registrati automaticamente.
2. Registrare eventi, font, immagini, ecc., che vengono compilati nel codice in modo che siano disponibili per i Componenti e gli Screen registrati in seguito.
3. Registrare i file `globals.xml` e quelli che definiscono Componenti e Screen.
4. Creare gli *Screen Permanenti* e altre schermate richieste.
5. Caricare la schermata iniziale.

8.6 Elementi dell'interfaccia utente

8.6.1 Componenti

Panoramica

I Componenti sono uno degli elementi fondamentali per la creazione di nuovi elementi dell'interfaccia utente.

I `<component>` supportano i seguenti tag XML figli:

- `<consts>`
- `<api>`
- `<styles>`
- `<view>`
- `<previews>`

Sebbene non possano contenere codice C, sono molto potenti:

- Possono estendere un altro componente o Widget (è possibile definire la base)

- I Componenti possono essere creati a partire da Widget e altri Componenti
- È possibile definire un'API personalizzata
- È possibile definire stili locali e utilizzare stili globali.
- È possibile definire costanti locali e utilizzare costanti globali.
- È possibile aggiungere chiamate di funzione, modifiche dell'oggetto o eventi di caricamento/creazione dello schermo. Vedere gli [Eventi XML](#)
- È possibile definire le anteprime per visualizzare i componenti in diverse impostazioni nell'UI Editor

Quando un Componente viene creato, può estendere un altro Componente o Widget, ad esempio `<view extends="lv_slider">`. Se un Widget viene esteso, il componente può utilizzare l'API del Widget, ad esempio `<view extends="lv_slider" min_value="20">`. Se un Componente viene esteso, è possibile utilizzare le proprietà `<api>`: `<view extends="my_button" button_text="Apply">`.

A differenza dei widget (che vengono sempre compilati nell'applicazione), i Componenti possono:

1. essere caricati a runtime da XML oppure
2. essere esportati in codice C dall'UI Editor di LVGL e compilati con l'applicazione.

Utilizzo dei file XML

Nei file XML

L'utilizzo dei Componenti in XML è molto intuitivo. Il nome del Componente può essere utilizzato come tag XML nella `<view>` di altri Componenti, Schermate e Widget.

Le proprietà dei Componenti sono semplicemente attributi XML.

Per caricare i Componenti da file, si presuppone che i file XML vengano salvati sul dispositivo come dati (array di byte) o come file. Una volta salvati i dati, ogni Componente può essere registrato e successivamente possono essere create istanze.

```
<!-- my_button.xml -->
<component>
  <view extends="lv_button" flex_flow="row">
    <lv_image src="logo"/>
    <my_h3 text="Title"/>
  </view>
</component>
```

Stili, Costanti e API custom possono essere descritti anche nei file XML.

Registrazione

Una volta creato un Componente (ad esempio, `my_button`), può essere registrato chiamando:

- `lv_xml_register_component_from_file("A:lvgl/examples/others/xml/my_button.xml")`
- `lv_xml_register_component_from_data("my_button", xml_data_of_my_button)`

Queste funzioni di registrazione elaborano i dati XML e memorizzano internamente le informazioni rilevanti. Ciò è necessario affinché LVGL riconosca il Componente per nome.

Si noti che la "A:" nel path sopra riportato è una "lettera identificativa del driver" del file system, da [File System \(lv_fs_drv\)](#) e viene utilizzata di conseguenza. Consultare la documentazione per i dettagli.

Quando caricato da un file, il nome del file viene utilizzato come nome del Componente.

Durante la registrazione, la `<view>` del Componente viene salvata nella RAM.

Istanziazione

Dopo la registrazione, è possibile creare una nuova istanza di qualsiasi Componente registrato con:

```
lv_obj_t * obj = lv_xml_create(lv_screen_active(), "my_button", NULL);
```

Il Widget creato è un normale Widget LVGL che può essere utilizzato come qualsiasi altro Widget creato manualmente.

L'ultimo parametro può essere NULL o un elenco di attributi, come questo:

```
/* Can be local */
char * my_button_attrs[] = {
    "x", "10",
    "y", "-10",
    "align", "bottom_left",
    NULL, NULL,
};

lv_obj_t * btn1 = lv_xml_create(lv_screen_active(), "my_button", my_button_attrs);
```

Utilizzo del Codice Esportato

Se non è necessario caricare XML a runtime, è possibile utilizzare l'editor dell'interfaccia utente di LVGL per esportare file C e H dai file XML dei componenti. Il codice risultante è completamente autosufficiente e i file XML non sono più necessari. Il codice risultante è simile a quello che si potrebbe scrivere anche a mano.

Il codice esportato si presenta così:

```
lv_obj_t * component_name_create(lv_obj_t * parent, ...api properties...);
```

dove `component_name` viene sostituito dal nome del file XML del Componente.

Quando un Componente viene utilizzato nel codice XML di un altro Componente e il codice viene esportato, verrà chiamata questa funzione `create`. Ciò significa che i Componenti non dispongono di un'API set/get dettagliata, ma vengono creati con un set fisso di parametri.

Se l'utente ha bisogno di accedere o modificare i valori in modo dinamico, si consiglia di un *Data binding tramite Subject*.

L'utente può anche richiamare queste funzioni `..._create()` in qualsiasi momento dal codice dell'applicazione per creare nuovi Componenti su richiesta.

Proprietà Custom

Le proprietà degli elementi figlio possono essere modificate, ad esempio:

```
<my_button x="10" width="200" />
```

Tuttavia, è anche possibile definire proprietà personalizzate nel tag `<api>`. Queste proprietà possono quindi essere passate a qualsiasi proprietà dei figli facendo riferimento ad esse tramite `$`. Ad esempio:

```
<!-- my_button.xml -->
<component>
  <api>
    <prop name="btn_text" type="string"/>
  </api>

  <view extends="lv_button">
    <lv_label text="$btn_text"/>
  </view>
</component>
```

Può essere utilizzato in questo modo:

```
<!-- my_list.xml -->
<component>
  <view>
    <my_button btn_text="First"/>
    <my_button btn_text="Second"/>
    <my_button btn_text="Third"/>
  </view>
</component>
```

In questa configurazione, la proprietà `btn_text` è obbligatoria. Tuttavia, può essere resa facoltativa impostando un valore di default:

```
<prop name="btn_text" type="string" default="Title"/>
```

Vedere `<api>` per maggiori dettagli e *Sintassi XML* per tutti i tipi supportati.

Esempi

L'esempio seguente illustra il passaggio di parametri e l'utilizzo della proprietà dell'etichetta `text` su un Componente. Vengono mostrati anche Stili e Costanti.

```
<!-- h3.xml -->
<component>
  <view extends="lv_label" style_text_color="0xffff00"/>
</component>
```

```
<!-- red_button.xml -->
<component>
  <api>
    <prop type="string" name="btn_text" default="None"/>
  </api>

  <consts>
    <int name="thin" value="2"/>
  </consts>

  <styles>
    <style name="pressed_style" border_width="#thin" border_color="0xff0000"/>
  </styles>

  <view extends="lv_button" style_radius="0" style_bg_color="0xff0000">
    <style name="pressed_style" selector="pressed"/>

    <h3 text="Some text"/>
    <h3 text="$btn_text" y="40"/>
  </view>
</component>
```

```
lv_xml_register_component_from_file("A:path/to/h3.xml");
lv_xml_register_component_from_file("A:path/to/red_button.xml");

/* Creates a button with "None" text */
lv_xml_create(lv_screen_active(), "red_button", NULL);

/* Use attributes to set the button text */
const char * attrs[] = {
  "btn_text", "Click here",
  NULL, NULL,
};
lv_xml_create(lv_screen_active(), "red_button", attrs);
```

API

8.6.2 | Widget

Panoramica

Oltre a Componenti e Screen, i Widget sono gli altri elementi costitutivi principali delle interfacce utente.

Il file XML di un Widget è racchiuso in un elemento radice XML `<widget>`.

I tag `<widget>` supportano i seguenti tag XML figlio:

- `<consts>`
- `<api>`
- `<styles>`
- `<view>`
- `<prevIEWS>`

Proprio come i Componenti, anche i Widget possono essere figli di altri Widget e Componenti.

Si noti che i Widget non possono essere caricati direttamente da XML, ma è possibile scrivere e registrare semplici parser XML per i Widget. Quando un Widget viene referenziato nell'XML di un Componente o di uno Screen, il parser XML verrà chiamato per mappare le proprietà alle chiamate di funzione C.

LVGL fornisce già tutte le funzioni di supporto e le librerie necessarie. Sono disponibili anche [molti esempi di parser XML qui](#) per i Widget LVGL nativi.

Widgets Nativi

I widget LVGL nativi (ad esempio, *Slider* (`lv_slider`), *Label* (`lv_label`), *Grafici* (`lv_chart`), ecc.) dispongono già di parser XML e sono quindi disponibili in XML.

Ad esempio:

```
<component>
  <view>
    <lv_label x="10" text="Hello"/>
  </view>
</component>
```

I widget nativi sono costituiti da:

- Codice C puro (ad esempio, lv_slider.c)
- Un file XML per definire solo l'API (ad esempio, lv_slider.xml), utilizzato solo nell'UI Editor per convalidare e completare automaticamente le proprietà
- Un file C parser XML per mappare gli attributi XML alle funzioni C (ad esempio, lv_xml_slider_parser.c)

Parser XML

Scrivere un parser

Per rendere i widget accessibili da XML, è necessario creare e registrare un parser XML per ogni widget. Il parser XML per il Widget Label si presenta così:

```
void * lv_xml_label_create(lv_xml_parser_state_t * state, const char ** attrs)
{
  /* Create the label */
  void * obj = lv_label_create(lv_xml_state_get_parent(state));
  return obj;
}

void lv_xml_label_apply(lv_xml_parser_state_t * state, const char ** attrs)
{
  void * obj = lv_xml_state_get_item(state);

  /* Apply the common properties, e.g., width, height, styles, flags, etc. */
  lv_xml_obj_apply(state, attrs);

  /* Process the label-specific attributes */
  for(int i = 0; attrs[i]; i += 2) {
    const char * name = attrs[i];
    const char * value = attrs[i + 1];

    if(lv_streq("text", name)) lv_label_set_text(obj, value);
    if(lv_streq("long_mode", name)) lv_label_set_long_mode(obj, long_mode_text_to_enum(value));
    /* Process more props here ... */
  }
}

/* Helper to convert strings to enum values */
static lv_label_long_mode_t long_mode_text_to_enum(const char * txt)
{
  if(lv_streq("wrap", txt)) return LV_LABEL_LONG_MODE_WRAP;
  if(lv_streq("scroll", txt)) return LV_LABEL_LONG_MODE_SCROLL;

  LV_LOG_WARN("%s is an unknown value for label's long_mode", txt);
  return 0; /* Return 0 in the absence of a better option. */
}
```

Utilizzando righe come `if(lv_streq("text", name)) lv_label_set_text(obj, value);`, qualsiasi funzione set può essere mappata alle proprietà XML.

Registrare un widget

Un processore XML per Widget può essere registrato come segue:

```
lv_xml_register_widget("lv_label", lv_xml_label_create, lv_xml_label_apply);
```

Dopo la registrazione, un Widget può essere creato in questo modo partendo da codice C:

```
const char * attrs[] = {
  "text", "Click here",
  "align", "center",
  NULL, NULL,
};

lv_obj_t * label = lv_xml_create(lv_screen_active(), "lv_label", attrs);
```

E in XML può essere utilizzato in questo modo:

```
<view>
  <lv_label width="100" text="I'm a label!" wrap="scroll"/>
</view>
```

Utilizzo nell'UI Editor di LVGL

Nuovo widget

È possibile creare nuovi widget scrivendo manualmente codice C (allo stesso modo in cui vengono creati i widget LVGL nativi), tuttavia, utilizzare l'editor dell'interfaccia utente è molto più veloce e semplice.

Quando viene creato un file XML e viene utilizzato l'elemento radice <widget>, vengono generati automaticamente i seguenti file .C/.H:

<widget_name>.gen.h

Contiene l'implementazione API generata del Widget (sovrascritta a ogni esportazione di codice)

<widget_name>_private_gen.h

Contiene API private e dati per il Widget (sovrascritti a ogni esportazione di codice)

<widget_name>.gen.c

Contiene le componenti interne del Widget, ad esempio costruttore con elementi figlio, distruttore, gestore eventi, ecc. (sovrascritti a ogni esportazione di codice)

<widget_name>.h

Include <widget_name>.gen.h e consente all'utente di definire API personalizzate. Lo scheletro viene esportato una sola volta

<widget_name>.c

Contiene hook da <widget_name>.gen.c e consente all'utente di scrivere codice personalizzato. Lo scheletro viene esportato una sola volta

<widget_name>.xml_parser.c

Elabora le stringhe XML e richiama le funzioni richieste in base agli attributi impostati. Lo scheletro viene esportato una sola volta

Aggiunta di Codice Personalizzato

<widget_name>.c contiene tre hook:

- Hook del costruttore:** Viene richiamato quando vengono creati il Widget e tutti i suoi elementi figlio. Qui è possibile apportare modifiche agli elementi figlio.
- Hook del distruttore:** Viene richiamato quando il Widget viene eliminato. Qui è necessario liberare tutta la memoria allocata manualmente.
- Hook dell'evento:** Viene richiamato all'inizio della callback dell'evento del Widget per eseguire qualsiasi azione personalizzata.

In questo file C, è necessario implementare anche le funzioni set per ciascuna API <prop>. La dichiarazione di queste funzioni viene esportata automaticamente in <widget_name>.gen.h.

Oltre a queste, in questo file è possibile implementare liberamente codice e funzioni personalizzati.

Elementi

Gli elementi sono parti interne del widget a cui è possibile accedere e/o creare dinamicamente. Ad esempio: tabs di una tabview, lista di una dropdown, serie di un grafico, ecc.

Come qualsiasi altra proprietà dell'API del Widget, anche gli Elementi possono essere definiti nel tag <api> dell'XML Widget.

Per saperne di più sugli Elementi, consultare la pagina di documentazione per <api>.

8.6.3 Screen

Panoramica

Gli "Screen" [schermate] funzionano in modo molto simile ai Componenti. Entrambi possono essere:

- Caricati da XML
- Contenere Widget e Componenti come elementi figli

Le schermate sono racchiuse nell'elemento radice XML `<screen>` e utilizzate per organizzare il contenuto dell'interfaccia utente.

Le schermate possono avere solo i seguenti tag XML figli:

- `<consts>`
- `<styles>`
- `<view>`

Ciò significa che gli Screen **non possono** avere:

- `<api>`: Gli Screen vengono sempre creati così come sono, senza parametri.
- `<preview>`: Nell'UI Editor di LVGL', `<preview>` viene utilizzato per impostare lo stile, le dimensioni e altri parametri della "schermata" di anteprima. Per gli Screen, le opzioni di anteprima sono definite in `project.xml`. Per saperne di più, consultare [Anteprima di Screen](#)

Utilizzo

Ogni file XML descrive una *Schermata*. Il nome del file XML sarà anche il nome della Schermata.

Questo esempio illustra una schermata in XML. Nell'esempio, vengono utilizzati i componenti `<my_header>` e `<my_main_cont>` per semplificare la schermata.

```
<screen>
  <consts>
    <string name="title" value="Main menu"/>
  </consts>

  <styles>
    <style name="dark" bg_color="#333333"/>
  </styles>

  <view>
    <my_header label="#title"/>
    <my_main_cont>
      <style name="dark"/>

      <button text="Weather" icon="cloudy"/>
      <button text="Messages" icon="envelope"/>
      <button text="Settings" icon="cogwheel"/>
      <button text="About" icon="questionmark"/>
    </my_main_cont>
  </view>
</screen>
```

Esportazione del codice

L'UI Editor di LVGL può esportare anche il codice C per gli Screen. Genera file `screen_name_gen.c/h`, contenenti una sola funzione di creazione `lv_obj_t * screen_name_create(void)`.

Utilizzando questa funzione, è possibile creare e caricare un numero qualsiasi di istanze di schermata, a seconda delle necessità.

Screen Permanent

L'unica proprietà possibile del tag `<screen>` è `permanent`, che può essere `true` o `false` (il default).

Questa proprietà influenza gli *eventi di caricamento e creazione delle schermate*. Se una schermata è permanente, quando ne viene caricata una nuova, quella permanente non viene eliminata, mantenendo così il suo stato. Le schermate non permanenti verranno eliminate e create automaticamente all'apertura e alla chiusura.

Si presume inoltre che le schermate permanenti vengano create all'inizializzazione dell'interfaccia utente. Quando il codice viene esportato dall'Editor dell'Interfaccia Utente, le schermate permanenti vengono effettivamente create e memorizzate in puntatori globali.

Alla luce di ciò, le schermate permanenti devono essere **caricate** solo da `<screen_load_event screen="my_permanent_screen"/>` e quelle non-permanent devono essere **create** da `<screen_create_event screen="my_non_permanent_screen"/>`.

Anteprima

Le schermate non supportano il tag `<preview>` perché non ha senso visualizzare in anteprima ogni schermata con risoluzioni diverse.

Poiché le schermate sono correlate all'hardware target nel file `project.xml`, è possibile definire più elementi `<display>`. Nell'Editor dell'Interfaccia Utente, quando si sviluppa una schermata, l'utente può selezionare tra tutti i display definiti nell'Anteprima e la Schermata verrà visualizzata con la risoluzione e la profondità di colore specificate.

Questo è utile per verificare i design "responsive".

Eventi

È molto comune caricare o creare Schermate al clic di un pulsante o in seguito ad altri eventi.

Entrambi sono supportati aggiungendo tag XML speciali come elementi figlio di Componenti o Widget:

```
<view>
    <lv_button>
        <lv_label text="Click or Long press me"/>

        <!-- Create an instance of "screen2" and load it. -->
        <screen_load_event screen="first" trigger="clicked" anim_type="fade"/>

        <!-- Load an already created instance of screen1 -->
        <screen_create_event screen="about" trigger="long_pressed"/>
    </lv_button>
</view>
```

Scoprire di più nella pagina degli *Eventi XML*.

8.6.4 Animazioni

Panoramica

Le animazioni XML sono basate sulle *Timeline delle animazioni*.

Le timeline sono composte da animazioni semplici. Ad esempio: "cambia il "bg_opa" di "my_button_2" da 0 a 255 in 500 ms."

Ogni componente può definire le proprie animazioni della sequenza temporale, che possono poi essere riprodotte dal componente stesso o da qualsiasi componente padre.

Definizione delle Timeline

Le timeline possono essere definite all'interno degli `<screen>` e dei `<component>`. Per i `<widget>`, le timeline sono supportate solo nell'Editor dell'Interfaccia utente di LVGL, da cui è possibile esportare anche il codice C.

Esempio:

```
<component>
    <animations>

        <!-- Show Component and its children -->
        <timeline name="load">
            <animation prop="translate_x" target="self" start="-30" end="0" duration="500"/>
            <animation prop="opa" target="text" start="0" end="255" duration="500" delay="200"/>

            <include_timeline target="icon" timeline="show_up" delay="300"/>
        </timeline>

        <!-- Shake horizontally -->
        <timeline name="shake">
            <animation prop="translate_x" target="self" start="0" end="-30" duration="150"/>
            <animation prop="translate_x" target="self" start="-30" end="30" duration="300" delay="150"/>
            <animation prop="translate_x" target="self" start="30" end="0" duration="150" delay="450"/>
        </timeline>
    </animations>

    <view>
        <lv_button width="200">
            <my_icon name="icon" src="image1"/>
            <lv_label name="text" text="Click me"/>
        </lv_button>
    </view>
</component>
```

In sintesi: all'interno di `<animations>`, si possono definire le `<timeline>`, ciascuna con un nome univoco a cui si potrà fare riferimento in seguito.

All'interno di `<timeline>`, si aggiungono le `<animation>` per descrivere ogni passaggio. Le proprietà supportate di `<animation>` sono:

- **prop:** Proprietà di stile da animare. Sono supportate tutte le proprietà di stile intere (i colori no).
- **selector:** Selettore di stile, ad esempio `knob|pressed`. Il default: `main|default`.
- **target:** Nome dell'elemento dell'interfaccia utente da animare. `self` si riferisce all'elemento radice del Componente (`<view>`).
- **start:** Valore iniziale (solo intero).
- **end:** Valore finale (solo intero).
- **duration:** Durata dell'animazione in millisecondi.
- **delay:** Ritardo prima dell'inizio in millisecondi. Il default è 0.
- **early_apply:** Se `true`, il valore iniziale viene applicato immediatamente, anche durante il delay. Il default è `false`.

`<include_timeline>` è utilizzabile anche nei `<timeline>` per il "merge" delle animazioni di un'altra timeline. Si immagini che nell'esempio sopra `my_icon` definisca una timeline "show_up" che sfuma e ingrandisce l'icona. Tutti questi elementi sono descritti in `my_icon.xml` in modo incapsulato, ma possono essere referenziati in altre timeline.

Per includere una sequenza temporale, utilizzare le seguenti proprietà:

- **target:** nome dell'elemento dell'interfaccia utente di destinazione la cui timeline deve essere inclusa. `self` si riferisce all'elemento radice del Componente (`<view>`).
- **timeline:** nome della timeline da includere. Deve essere definito nel file XML del target.
- **delay:** Ritardo prima dell'inizio in millisecondi. Il default è 0.

Riproduzione di Timeline

Le Timeline possono essere attivate da eventi (ad esempio un clic) utilizzando `<play_timeline_event>` come elemento figlio di qualsiasi widget.

Esempio:

```
<view>
    <lv_label name="title" text="Hello world!"/>
    <custom_button name="button" y="20">
        <play_timeline_event trigger="clicked" target="button" timeline="bounce"/>
        <lv_label text="Click me!"/>
    </custom_button>
</view>
```

Si imposta un elemento UI `target` e si seleziona una delle sue `timeline` da riprodurre. Se `target="self"`, la timeline viene cercata nel Componente/widget/Screen corrente (ovvero nel file XML corrente).

Si può anche impostare `delay` e `reverse="true"` durante la riproduzione di una timeline.

Dietro le Quinte

Capire il funzionamento interno delle timeline aiuta a utilizzarle in modo efficace.

Quando un file XML viene registrato, il contenuto della sezione `<animations>` viene analizzato e i dati della `<timeline>` vengono memorizzati come "blueprint" [progetto]. I descrittori memorizzano i nomi dei target come stringhe.

Quando viene creata un'istanza di un Componente o di uno Screen, come ultimo passaggio vengono creati e inizializzati `lv_anim_timeline` dai "blueprint" salvati. Se vengono utilizzati anche `<include_timeline>`, la timeline richiesta viene inclusa anche nella timeline del Componente. Poiché anche tutti i figli vengono creati a questo punto, i nomi dei target di animazione salvati vengono risolti in puntatori utilizzando `lv_obj_find_by_name`.

Le istanze della timeline create e i relativi nomi vengono salvati nell'istanza del Componente.

Poiché ogni istanza ha la propria timeline, è possibile avere più Componenti (ad esempio 10 `<list_item>`) e riprodurre le rispettive timeline `load` in modo indipendente con ritardi diversi.

Quando un `<play_timeline_event>` viene aggiunto a un elemento dell'interfaccia utente, i nomi del target e della timeline vengono salvati come stringhe. Non può utilizzare puntatori poiché l'evento può fare riferimento a elementi dell'interfaccia utente che verranno creati solo in seguito nella `<view>`.

Infine, quando viene attivato l'evento di riproduzione della timeline, la timeline selezionata viene recuperata tramite il suo nome dalla destinazione e avviata in base agli altri parametri (inversione, ritardo, ecc.).

8.6.5 API

Il tag `<api>` può essere figlio dei tag `<widget>` e `<component>`. (Gli `<screen>` non supportano API personalizzate).

L'unico punto in comune è che sia i Widget che i Componenti supportano `<prop>` (proprietà) nel tag `<api>` per descrivere la loro interfaccia.

Tuttavia, poiché Widget e Componenti funzionano in modo molto diverso (i Widget hanno codice C, mentre i Componenti sono XML puro), anche le proprietà vengono interpretate in modo diverso.

Componenti

Panoramica

Mentre i widget possono avere API set/get complesse, i Componenti sono molto semplici.

Quando l'XML di un componente viene convertito in file C, viene generata solo una funzione `create`, in cui tutti i `<prop>` sono argomenti. Ad esempio:

```
<api>
  <prop name="prop1" type="int"/>
  <prop name="prop2" type="string"/>
</api>
```

Riferimento alle proprietà

I `<prop>` vengono semplicemente inoltrati alle API dei Widget o dei Componenti. Ad esempio, se un componente ha `<prop name="button_label" type="string"/>`, può essere utilizzato in un'etichetta figlia come `<lv_label text="$button_label"/>`.

Nel codice generato dall'editor dell'interfaccia utente di LVGL, questi vengono passati come argomenti nelle funzioni `create`/`set`.

Valori di default

Poiché ogni proprietà viene passata come argomento alla funzione "create", ciascuna deve avere un valore. Ciò può essere garantito:

- Semplicemente impostandoli nell'istanza XML
- Specificando un valore di default in `<api>`, ad esempio `<prop name="title" type="string" default="my_default_text"/>`

Generazione del codice

L'editor dell'interfaccia utente di LVGL può anche generare codice dall'XML del componente. A seconda di `<api>`, viene generata una funzione come questa:

```
lv_obj_t * my_component_create(lv_obj_t * parent, int32_t prop1, const char * prop2);
```

Queste proprietà vengono impostate una sola volta (al momento della creazione) e non ci sono funzioni `set` specifiche per modificarle in seguito. L'API generale di LVGL può comunque essere utilizzata per modificare qualsiasi widget nel componente, ma non vengono generate funzioni API dedicate.

Limitazioni

Si noti che nessuna delle funzionalità dell'API Widget come `<param>`, `<enumdef>` o `<element>` può essere utilizzata per i Componenti. Sono supportate solo le proprietà semplici inoltrate.

Esempio

```
<!-- my_button.xml -->
<component>
  <api>
    <prop name="button_icon" type="image" default="NULL"/>
    <prop name="button_label" type="string" default="Label"/>
  </api>

  <view extends="lv_button" flex_flow="row" width="100%">
    <lv_image src="$button_icon" inner_align="stretch" width="16" height="16"/>
    <lv_label text="$button_label"/>
  </view>
</component>

<!-- my_list.xml -->
<component>
  <view flex_flow="column">
    <my_button button_label="First"/>
    <my_button button_label="Wifi" button_icon="img_wifi"/>
    <my_button button_label="Third"/>
  </view>
</component>
```

I Widget

Proprietà

Le proprietà sono la parte fondamentale della descrizione dell'API di un Widget.

```
<api>
  <prop name="text" type="string" help="Text of the label."/>
</api>
```

Parametri

Alcune proprietà accettano più parametri. Ad esempio: `lv_label_set_bind_text(label, subject, "%d °C")`

Viene descritta come:

```
<api>
  <prop name="bind_text" help="Bind a subject's value to a label.">
    <param name="bind_text" type="subject" help="Integer or string subject."/>
    <param name="fmt" type="string" help="Format string, e.g. %d °C" />
  </prop>
</api>
```

e utilizzata come:

```
<lv_label bind_text="subject" bind_text-fmt="%d °C"/>
```

I parametri con lo stesso nome della proprietà possono essere referenziati direttamente. Altri parametri utilizzano la notazione `property-param`.

I parametri non impostati vengono sostituiti da:

- Il loro valore di default (se definito)
- Valori di default specifici del tipo (ad esempio, 0, false, NULL)

Mappatura

Ogni `<prop>` viene mappata a una funzione `set`. Questa mappatura è implementata nel parser XML del Widget. Vedere [i parser XML LVGL](#).

Se vengono utilizzati `<param>`, vengono passati alla stessa funzione `set`. Se una proprietà non è impostata su un'istanza del Widget, viene ignorata e viene utilizzato il valore di default nativo nel Widget.

<enumdef>

Utilizzato solo con i Widget, questo tag definisce gli enum per i valori dei parametri.

```
<api>
  <enumdef name="my_widget_mode" help="Possible modes">
    <enum name="normal" help="Normal mode" value="0x10"/>
    <enum name="inverted" help="Inverted mode"/>
  </enumdef>
  <prop name="mode" help="Set Widget mode">
    <param name="mode" type="enum:my_widget_mode"/>
  </prop>
</api>
```

(continues on next page)

(continua dalla pagina precedente)

```
</prop>
</api>
```

I valori effettivi dei campi enum non sono importanti durante l'esportazione del codice, poiché solo i nomi vengono utilizzati e risolti dal compilatore. I parser XML devono gestire la mappatura dei nomi delle stringhe enum (ad esempio "normal") agli enum C (ad esempio MY_WIDGET_MODE_NORMAL).

<element>

Anche esclusivi dei Widget, gli elementi definiscono sotto-widget o strutture interne (ad esempio, serie di grafici, liste di dropdown, tab di view).

Gli elementi sono anche molto utili per creare "slot" (simili ai tab di una tabview), come un'area di contenuto e intestazione per un widget window, dove è possibile creare direttamente elementi figlio.

Gli elementi sono descritti all'interno del tag `<api>` e possono avere elementi figlio `<arg>` e `<prop>`:

- Gli `<arg>` sono obbligatori e utilizzati durante la creazione/acquisizione dell'elemento.
- I `<prop>` sono opzionali e mappati ai setter.

Ecco un esempio per definire un "indicatore" che può essere aggiunto dinamicamente a un widget. Creerà elementi `my_indicator_t *`, allo stesso modo di `lv_chart_add_series`, per esempio.

```
<api>
  <element name="indicator" type="my_indicator_t" help="The indicator of my_widget" access="add">
    <!-- args are passed when the element is created -->
    <arg name="color" type="color"/>
    <arg name="max_value" type="int"/>

    <!-- props can be set by setters at any time -->
    <prop name="value" type="int"/>
  </element>
</api>
```

I tipi di elemento access possono essere:

- add: Crea più elementi dinamicamente. Ad esempio tab di tabview.
- get: Accede agli elementi creati implicitamente. Ad esempio, elenchi a discesa.
- set: Accesso a parti indicizzate. Ad esempio, celle di tabella.
- custom: Mappa funzioni C personalizzate in XML, ad esempio `bind_state_is_eq`

Poiché gli elementi add e get restituiscono un oggetto, hanno anche un tipo. Questo tipo può essere qualsiasi tipo custom, ad esempio, `type="my_data"`. Nel codice esportato, il valore restituito verrà salvato in una variabile `my_data_t *`.

Se il tipo è `type="lv_obj"`, l'elemento può avere widget o componenti figli.

Gli elementi sono referenziati come `<widget-element>` in `<view>`. Le parti del nome sono separate da -. Poiché - non è consentito all'interno dei nomi, è sicuro utilizzarlo come separatore.

Si noti che solo l'API può essere definita in XML per gli elementi; le implementazioni devono essere in C.

access="add"

Gli elementi vengono creati tramite una funzione add:

```
<api>
  <element name="indicator" type="obj" help="The indicator of my_widget" access="add">
    <arg name="color" type="color"/>
    <arg name="max_value" type="int"/>
    <prop name="value">
      <param name="value" type="int"/>
    </prop>
  </element>
</api>
```

Utilizzata in una vista:

```
<my_widget width="100px">
  <my_widget-indicator name="ind1" color="0xffff00" max_value="120" value="30"/>
</my_widget>
```

L'UI Editor genera questo:

```
lv_obj_t * my_widget_add_indicator(lv_obj_t * parent, lv_color_t color, int32_t max_value);
void my_widget_set_indicator_value(lv_obj_t * obj, int32_t value);
```

access="get"

Utilizzata per elementi interni/impliciti:

```
<api>
<element name="control_button" type="obj" help="A control button of my_widget" access="get">
<arg name="index" type="int"/>
<prop name="title" type="string"/>
</element>
</api>
```

Utilizzata in una vista:

```
<my_widget width="100px">
<my_widget-control_button name="btn1" index="3" title="Hello"/>
</my_widget>
```

L'UI Editor genera questo:

```
lv_obj_t * my_widget_get_control_button(lv_obj_t * parent, int32_t index);
void my_widget_set_control_button_title(lv_obj_t * obj, const char * text);
```

access="set"

Utilizzata per l'accesso indicizzato, come l'impostazione di valori in una tabella:

```
<api>
<element name="item" type="obj" access="set">
<arg name="index" type="int"/>
<prop name="icon" type="img_src"/>
<prop name="color" type="color"/>
</element>
</api>
```

Utilizzata in una vista:

```
<my_widget width="100px">
<my_widget-item index="3" icon_src="image1" color="0xffff0000"/>
</my_widget>
```

L'UI Editor genera questo:

```
void my_widget_set_item_icon(lv_obj_t * parent, int32_t index, const void * icon_src);
void my_widget_set_item_color(lv_obj_t * parent, int32_t index, lv_color_t color);
```

access="custom"

Utilizzata per descrivere qualsiasi funzione API personalizzata con un nome personalizzato. Gli elementi "custom" possono avere solo argomenti e nessun tipo, quindi sono puri setter.

```
<element name="bind_color" access="custom">
<arg name="subject" type="subject"/>
<arg name="new_color" type="color"/>
<arg name="ref_value" type="int"/>
</element>
```

Utilizzata in una vista:

```
<my_widget width="100px">
<my_widget-bind_color subject="subject_1" color="0xffff0000" ref_value="15"/>
</my_widget>
```

L'UI Editor genera questo:

```
void my_widget_bind_color(lv_obj_t * parent, lv_subject_t * subject, lv_color_t color, int32_t ref_value);
```

8.6.6 Costanti

Panoramica

Le costanti possono essere definite per sostituire qualsiasi valore con un tipo selezionato o per essere utilizzate come valori speciali.

I tipi supportati sono: - color - px - percentage - string - opa - bool

Utilizzo

```
<consts>
  <color name="color1" value="#0xffff00" help="Primary color"/>
  <px name="pad_xs" value="8" help="Small padding"/>
</consts>
```

Le costanti possono essere utilizzate in:

- Proprietà di stile
- Proprietà di Widget e Componenti

I valori costanti possono essere referenziati tramite #constant_name. Ad esempio:

```
<styles>
  <style name="style1" bg_color="#color1"/>
</styles>
```

8.6.7 Eventi in XML

Panoramica

Esistono diversi modi per definire gli eventi per le interazioni utente. Questi eventi possono essere aggiunti come elementi figli di qualsiasi widget.

Trigger

In tutti i tipi di evento, l'attributo trigger definisce il tipo di azione dell'utente che deve attivare l'evento. Tutti i tipi di evento LVGL sono supportati con una mappatura semplice:

- *LV_EVENT_ALL*: "tutti"
- *LV_EVENT_CLICKED*: "cliccato"
- *LV_EVENT_PRESSED*: "premuto"
- ecc.

Chiamata di Funzione

Le funzioni definite dall'utente possono essere chiamate in questo modo:

```
<view>
  <lv_button width="200" height="100">
    <event_cb callback="my_callback_1" trigger="clicked" user_data="some_text"/>
    <lv_label text="Hello"/>
  </lv_button>
</view>
```

Quando l'XML viene caricato in fase di esecuzione, il nome della callback deve essere mappato a una funzione utilizzando `lv_xml_register_event_cb("my_callback_1", an_event_handler)`.

La callback deve seguire la firma standard LVGL per le callback di eventi: `void an_event_handler(lv_event_t * e);`

Nel codice C esportato, si presume che esista una funzione con lo stesso nome. Ad esempio, `callback="my_callback_1"` verrà esportato come:

```
void my_callback_1(lv_event_t * e); /* At the beginning of the exported file */
lv_obj_add_event_cb(obj, my_callback_1, LV_EVENT_CLICKED, "some_text");
```

`user_data` è facoltativo. Se omesso, verrà passato NULL.

Eventi di Caricamento e Creazione delle Schermate

Utilizzando i tag `<screen_load_event>` e `<screen_create_event>` come elementi figli di un widget o di un componente, le schermate possono essere caricate o create in base a un trigger (ad esempio, un clic).

La differenza tra load e create è che:

- **load:** Carica semplicemente una schermata già esistente. Dopo aver abbandonato la schermata, questa rimane in memoria, quindi tutti gli stati vengono preservati.
- **create:** La schermata viene creata dinamicamente e, quando si abbandona la schermata, viene eliminata, quindi, tutte le modifiche vengono perse (a meno che non vengano salvate in `subjects`).

Entrambi i tag supportano i seguenti attributi opzionali:

- **trigger:** Codice evento che attiva l'azione (ad esempio, "clicked", "long_pressed", ecc.). Il default: "clicked".
- **anim_type:** Descrive come viene caricata la schermata (ad esempio, "move_right", "fade_in"). Il default: "none".
- **duration:** Durata dell'animazione in millisecondi. Il default: 0. Utilizzato solo se `anim_type` non è "none".
- **delay:** Tempo di attesa prima del caricamento della schermata in millisecondi. Il default: 0.

Questo è un semplice esempio di load e create:

```
<!-- screen1.xml -->
<screen>
  <view style_bg_color="0xffff88">
    <lv_button>
      <lv_label text="Create"/>
      <!-- Create an instance of "screen2" and load it. -->
      <screen_create_event screen="screen2" anim_type="over_right" duration="500" delay="1000"/>
    </lv_button>
  </view>
</screen>

<!-- screen2.xml -->
<screen>
  <view style_bg_color="0x77ff88">
    <lv_button>
      <lv_label text="Load"/>
      <!-- Load an already created instance of screen1.-->
      <screen_load_event screen="screen1"/>
    </lv_button>
  </view>
</screen>
```

```
lv_xml_register_component_from_data("screen1", screen1_xml);
lv_xml_register_component_from_data("screen2", screen2_xml);

/* Create an instance of screen1 so that it can be loaded from screen2. */
lv_obj_t * screen1 = lv_xml_create(NULL, "screen1", NULL);
lv_screen_load(screen1);
```

Impostare il Valore di Subject

È possibile impostare un valore `Subject` durante l'interazione dell'utente aggiungendo un elemento figlio speciale a qualsiasi widget:

```
<view>
  <lv_button width="200" height="100">
    <subject_set_int_event trigger="clicked" subject="subject_int" value="10"/>
    <subject_set_float_event trigger="clicked" subject="subject_float" value="12.34"/>
    <subject_set_string_event trigger="clicked" subject="subject_string" value="Hello"/>
    <lv_label text="Set the values"/>
  </lv_button>
</view>
```

L'utilizzo è semplice: il `subject` specificato verrà impostato sul `value` specificato quando si verifica il `trigger`.

Incrementare il Valore di Subject

L'incremento o il decremento di un valore `Subject` può essere definito come segue:

```
<view>
  <lv_button width="200" height="100">
    <subject_increment_event trigger="clicked" subject="subject_int1" step="10"/>
    <subject_increment_event trigger="clicked" subject="subject_int2" step="-10" min_value="0" max_value="50"/>
    <subject_increment_event trigger="clicked" subject="subject_float1" step="2"/>
  </lv_button>
</view>
```

L'elemento `<subject_increment_event>` definisce uno step da aggiungere al valore corrente del soggetto quando si verifica il trigger.

`subject` deve essere un soggetto `int` o `float`.

Se `step` è **negativo**, il valore del soggetto verrà decrementato. Ora sono supportati solo valori `step` interi.

Facoltativamente, è possibile impostare `min_value` e/o `max_value` per limitare il valore del `subject`. I valori min/max di default sono rispettivamente `INT32_MIN` (-2B) e `INT32_MAX` (+2B).

Anche `rollover` è una proprietà facoltativa. Se è `false` (il default) si ferma al valore min/max, se `true` salta all'altro estremo.

8.6.8 Anteprima

Panoramica

In `<component>` e `<widget>`, è possibile definire i tag `<preview>`. Questi **non** vengono esportati nel codice e **non** vengono caricati da XML.

Vengono utilizzati solo dall'Editor dell'Interfaccia Utente per descrivere il contesto del Componente. Per esempio, si potrebbe voler:

- cambiare il background dell'anteprima dell'editor in scuro,
- centrare il Componente,
- impostare i margini,
- modificare le dimensioni dell'anteprima.

È possibile utilizzare `style_radius` per arrotondare l'anteprima.

Utilizzo

Si può pensare a un tag `<preview>` come a un `lv_obj` in cui possono essere utilizzate le seguenti proprietà:

- `width`, `height`
- Qualsiasi proprietà di stile locale, ad esempio, `style_bg_color="0x333"`
- `flex` e `flex_flow`

È anche possibile definire più anteprime e nell'Editor dell'Interfaccia Utente è possibile selezionarne una.

Per gli Screen

Le schermate non supportano il tag `<preview>` perché non ha senso visualizzare in anteprima ogni schermata con risoluzioni diverse.

Poiché le schermate sono correlate all'hardware target nel file `project.xml`, è possibile definire più elementi `<display>`. Nell'Editor dell'Interfaccia Utente, quando si sviluppa una schermata, l'utente può selezionare tra tutti i display definiti nell'Anteprima e la Schermata verrà visualizzata con la risoluzione e la profondità di colore specificate.

Questo è utile per verificare i design "responsive".

Esempio

```
<component>
<previews>

<preview name="small_dark_round"
        width="240" height="240"
        style_bg_color="0x333" style_pad_all="32"
        style_radius="32"/>

<preview name="large_light"
        width="1980" height="1080"
        style_bg_color="0xeeeee"/>

</previews>
<view>
  ...
</view>
```

(continues on next page)

(continua dalla pagina precedente)

```
</view>
</component>
```

8.6.9 Stili

Panoramica

Nei file XML, è possibile utilizzare sia fogli di stile (`lv_style_t`) che stili locali.

Fogli di Stile

Nella sezione `<styles>`, gli stili e le loro proprietà possono essere definiti in questo modo:

```
<style name="red"
      help="What is this style about?"
      border_width="2px"
      border_color="#0xffff0000"/>
```

Gli stili possono essere referenziati in questo modo in `<view>`:

```
<view>
  <lv_slider value="20">
    <style name="main"/>
    <style name="red" selector="knob"/>
    <style name="blue" selector="knob|focused"/>
  </lv_slider>
</view>
```

Come mostrato nell'esempio, parti e stati possono essere impostati utilizzando `selector`.

Binding dello Stile

Invece di aggiungere stili direttamente agli elementi dell'interfaccia utente, è anche possibile aggiungerli in modo condizionale quando il valore di un `Subject` è uguale a un valore di riferimento.

Funziona a runtime ed è un ottimo modo per controllare l'aspetto o l'evento del layout in base a una condizione.

Un tipico caso d'uso è il passaggio da tema chiaro a tema scuro. Richiede

- un subject come `dark_theme_on`
- alcuni stili predefiniti che vengono aggiunti normalmente con il tag `<style>`
- alcuni stili scuri per controllare i colori richiesti per scurire

Ecco un esempio:

```
<component>
  <styles>
    <style name="style_base" bg_color="#0xeeee" text_color="#0x1111" radius="20" />
    <style name="style_dark" bg_color="#0x3333" text_color="#0xeeee" radius="20" />
  </styles>

  <view extends="lv_button">
    <style name="style_base" />
    <bind_style name="style_dark" subject="dark_theme_on" ref_value="1"/>
    <lv_label text="Apply"/>
  </view>
</component>
```

Stili Locali

Gli stili locali possono essere utilizzati direttamente, ad esempio:

```
<lv_label style_bg_opa="200" style_bg_color="#0x123456"/>
```

I selettori non sono ancora supportati per le proprietà di stile locale.

Gradienti

Prima del tag `<styles>`, il tag `<gradients>` può essere utilizzato per descrivere diversi gradienti, a cui è possibile fare riferimento in seguito negli stili.

Quando si crea un gradiente, è possibile farvi riferimento tramite il suo nome, ad esempio:

```
<style bg_grad="grad1"/>
```

o da

```
<lv_button style_bg_grad="grad1"/>
```

Notare che i gradienti non sono ancora supportati nell'editor dell'interfaccia utente di LVGL.

Gradiente Orizzontale o Verticale

Per definire un semplice gradiente <horizontal> o <vertical>:

```
<gradients>
  <horizontal name="grad1">
    <stop color="#ff0000" offset="20%" opa="40%"/>
    <stop color="#00ff00" offset="128" opa="100%"/>
  </horizontal>
</gradients>
```

Gradiente Lineare

Per definire un gradiente inclinato da due punti:

```
<gradients>
  <linear name="grad1" start="50 50" end="100 80">
    <stop color="#ff0000" offset="20%" opa="100%"/>
    <stop color="#00ff00" offset="240" opa="100%"/>
  </linear>
</gradients>
```

Gradiente Radiale

Per definire un gradiente radiale:

```
<gradients>
  <radial name="grad1" center="100 50%" edge="200 50%" focal_center="50 80%" focal_edge="55 80%">
    <stop color="#ff0000" opa="100%"/>
    <stop color="#00ff00" opa="100%"/>
  </radial>
</gradients>
```

Gradiente Conico

Per definire un gradiente conico:

```
<gradients>
  <conical name="grad1" center="80 50%" angle="45 270">
    <stop color="#ff0000" opa="100%"/>
    <stop color="#00ff00" opa="100%"/>
  </conical>
</gradients>
```

8.6.10 View

Panoramica

Il tag <view> può essere utilizzato:

- nei <component>
- nei <widget>
- negli <screen>
- nei <text>

per descrivere l'aspetto di questi elementi. All'interno di <view>, è possibile aggiungere elementi figlio in modo nidificato utilizzando i widget e i component. Ad esempio:

```
<view>
  <lv_button width="200">
    <my_icon src="image1"/>
    <lv_label text="Click me"/>
  </lv_button>
</view>
```

Extends

Anche `<view>` è un widget o un componente, che diventerà il genitore dei figli. Per definire il tipo di `<view>`, si usa l'attributo `extends`. Ad esempio, `extends="lv_slider"`. In questo caso, verrà creato prima un `lv_slider` e a esso verranno aggiunti i figli.

Aggiungendo proprietà a `<view>`, è possibile personalizzare il tipo esteso. Ad esempio:

```
<view extends="lv_slider" width="100%" style_bg_color="0xffff800" flex_flow="row">
  <style name="my_style" selector="pressed|knob"/>
  <lv_label text="Current value: />
  <lv_label bind_text="subject_1"/>
</view>
```

Regole per i valori consentiti di `extends`:

- `<component>`: può estendere sia i `<widget>` che i `<component>`
- `<widget>`: può estendere solo i `<widget>`
- `<screen>`: non può estendere nulla
- `<test>`: può estendere `<widget>`, `<component>` o `<screen>`

8.7 Risorse

8.7.1 Immagini

Panoramica

In XML, le immagini sono considerate risorse esterne che devono avere un nome per poter essere referenziate nei file XML. Di seguito è illustrato come mappare le immagini con i nomi.

Registrazione delle Immagini

Da File

Se le immagini vengono utilizzate come file (ad esempio, immagini PNG caricate da un file system), possono essere semplicemente aggiunte a `globals.xml`:

```
<images>
  <file name="avatar" src_path="images/avatar1.png"/>
  <file name="logo" src_path="images/path/to/my_logo.png"/>
</images>
```

Quando si registra `globals.xml` con `lv_xml_register_component_from_file("A:path/to/globals.xml")`, i nomi vengono automaticamente mappati al path.

Le immagini possono avere path relativi in `globals.xml`. Prima di registrare `globals.xml`, è possibile richiamare `lv_xml_set_default_asset_path("path/prefix/")` per impostare la cartella padre. Il path dei file delle immagini verrà aggiunto al path impostato qui.

Dai Dati

Se le immagini vengono convertite in array e compilate nel firmware, devono essere registrate esplicitamente utilizzando:

```
lv_xml_register_image(NULL, "image_name", &my_image)
```

dove `my_image` è un `lv_image_desc_t`.

Dopodiché, può essere utilizzato in modo identico ai file immagine:

```
<lv_image src="image_name" align="center"/>
```

Utilizzo in XML

Dopo la registrazione, è possibile fare riferimento alle immagini tramite il loro nome:

```
<lv_image src="avatar" align="center"/>
```

Note per l'Editor dell'Interfaccia Utente

Quando si utilizza l'editor dell'interfaccia utente di LVGL, le immagini possono essere aggiunte a `globals.xml` utilizzando il tag `<data>` invece di `<file>`.

In questo caso, l'editor genererà l'array C (`lv_image_desc_t`) e genererà anche il codice per registrare le immagini.

È anche possibile impostare il `color_format="..."` target delle immagini. Sono supportati tutti i formati di colore tipici, come i1...i8, a1...a8, rgb565, rgb888, argb8888, ecc.

8.7.2 Font

Panoramica

In XML, i font sono considerati risorse esterne. È necessario selezionare e denominare un motore di font di destinazione per potervi fare riferimento in seguito nei file XML.

Registrazione dei Font

Da File

I font utilizzati direttamente da un file TTF possono essere elencati nella sezione `<fonts>` di `globals.xml`.

Ogni elemento figlio di `<fonts>` inizia col nome del motore del font desiderato, una proprietà `name` e altre proprietà a seconda del motore selezionato.

`as_file` deve essere impostato a `true` per indicare che il font viene trattato come un file.

Ad esempio:

```
<globals>
  <font>
    <bin as_file="false" name="medium" src="path/to/file.ttf" range="0x20-0x7f" symbols="" size="24"/>
    <tiny_ttf as_file="true" name="big" src_path="path/to/file.ttf" range="0x20-0x7f" size="48"/>
  </font>
</globals>
```

Quando si registra `globals.xml` con `lv_xml_register_component_from_file("A:path/to/globals.xml")`, i font vengono creati automaticamente con il motore del font selezionato e mappati ai nomi specificati.

I font possono avere path relativi in `globals.xml`. Prima di registrare `globals.xml`, è possibile richiamare `lv_xml_set_default_asset_path("path/prefix/")` per impostare la cartella padre. I path dei font verranno quindi aggiunti a questo percorso di base.

Dai Dati

Se un font viene utilizzato da dati C (array) compilati nel firmware, il puntatore del font può essere registrato in XML in questo modo:

```
lv_xml_register_font(scope, "font_name", &my_font);
```

I font devono essere registrati prima di qualsiasi file XML che vi faccia riferimento, in modo che i dati del font siano già noti.

Nell'editor dell'interfaccia utente di LVGL, quando si utilizzano i font come dati, è anche necessario aggiungerli a `globals.xml` con `as_file=false`, in modo che l'editor possa generare i file C richiesti ed eseguire anche la validazione quando si fa riferimento ai font di dati.

Utilizzo in XML

Una volta registrato un font, è possibile farvi riferimento tramite il suo nome negli stili o anche nelle proprietà <api>:

```
<component>
  <api>
    <prop name="title_font" type="font" default="my_font_32"/>
  </api>

  <styles>
    <style name="subtitle" text_font="my_font_24"/>
  </styles>

  <view flex_flow="column">
    <lv_label style_text_font="$title_font" text="Title"/>
    <lv_label text="I'm the subtitle">
      <style name="subtitle"/>
    </lv_label>
  </view>
```

Motori dei Font

<bin>

I font binari possono fare riferimento a:

1. Un file di font ".bin" creato dal convertitore di font di LVGL da un file TTF (quando `as_file="true"`). Il font verrà caricato tramite `lv_binfont_create`.
2. Un puntatore `lv_font_t` compilato nel firmware (`as_file="false"`). Non è necessario alcun caricamento poiché `lv_font_t` può essere utilizzato direttamente.

I font <bin> richiedono queste proprietà:

name
Il nome per fare riferimento al font in seguito

src_path
Path del file del font

size
Dimensione del font in px (ad esempio, "12")

bpp
Bit-per-pixel: 1, 2, 4, o 8

as_file
`true` se il font è un file ".bin", `false` se è un `lv_font_t` in C

L'editor dell'interfaccia utente di LVGL genera sempre una chiamata a `lv_xml_register_font` utilizzando il set name. Se `as_file` è:

- `false`: Genera un file C con le strutture `lv_font_t`.
- `true`: Genera un file ".bin".

I font binari supportano anche la selezione di un sottoinsieme di caratteri:

range
Ad esempio, "0x30-0x39 0x100-0x200" per specificare gli intervalli Unicode da includere. Il default è "0x20-0x7F" per coprire l'intervallo ASCII.

symbols
Elenco di ulteriori caratteri da aggiungere, ad esempio "°ÁÓÚ". Può essere utilizzato insieme a range. Il default è vuoto (nessun carattere aggiuntivo).

<tiny_ttf>

I font TinyTTF utilizzano il motore *Motore del Font Tiny TTF* per caricare i file TTF direttamente o dai dati.

Proprietà richieste:

name
Il nome per fare riferimento al font in seguito

src_path

Path del file del font

size

Dimensione del font in px (ad esempio, "12")

as_file

true o false

L'editor dell'interfaccia utente di LVGL genera una chiamata a `lv_xml_register_font` utilizzando il set name. Se `as_file` è:

- `false`: Viene generato anche un file C con dati raw.
- `true`: Non viene generato altro, poiché il file TTF verrà utilizzato direttamente.

<freetype>

I font FreeType utilizzano il motore *Motore del Font FreeType* per caricare direttamente i file TTF. Il caricamento dai dati non è supportato, quindi `as_file` è sempre considerato `true`.

Proprietà richieste:

name

Il nome per fare riferimento al font in seguito

src_path

Path del file del font

size

Dimensione del font in px (ad esempio, "12")

L'editor dell'interfaccia utente di LVGL genera una chiamata a `lv_xml_register_font` utilizzando il set name.

8.8 Funzionalità

8.8.1 Data binding (I Subject)

Panoramica

Con l'aiuto dei *Soggetti*, è possibile creare l'interfaccia utente.

Un soggetto è un elemento di dati globale il cui valore può essere impostato dall'applicazione o dall'interfaccia utente e può essere associato alle proprietà del widget.

Ad esempio, il valore di un soggetto `room1_temperature` può essere impostato nell'applicazione quando viene misurata la temperatura e può essere associato a un'etichetta come questa:

```
<lv_label bind_text="room1_temperature"/>
```

Definizione dei Subject

I soggetti possono essere creati in `globals.xml` in questo modo:

```
<globals>
  <subjects>
    <int name="battery_power" value="32"/>
    <string name="user_name" value="John"/>
  </subjects>
</globals>
```

Come mostra l'esempio, un soggetto è costituito da un tipo, un nome e un valore iniziale. Attualmente, sono supportati solo i tipi intero e stringa.

Binding Semplice

Alcuni widget (ad esempio, label, slider) supportano il binding del valore del soggetto direttamente al widget. Questi "binding" [collegamenti] utilizzano attributi che iniziano con `bind_*` e fanno riferimento a un soggetto.

```
<lv_slider bind_value="some_subject"/>
<lv_label bind_text="some_subject"/>
```

Una volta creato un binding, se il valore del soggetto cambia (ad esempio, regolando lo slider), tutti i widget associati verranno aggiornati automaticamente.

Binding Complessi

Nei casi più complessi, quando un binding richiede più parametri, il binding può essere aggiunto come elemento figlio di un widget. Questo consente di associare più soggetti con parametri diversi. Ad esempio:

```
<lv_label text="Hello world">
  <lv_obj-bind_flag_if_eq subject="subject1" flag="hidden" ref_value="10"/>
  <lv_obj-bind_flag_if_gt subject="subject1" flag="clickable" ref_value="20"/>
</lv_label>
```

Spiegazione dei binding complessi:

- `<lv_obj-bind_flag_if_eq>` — Imposta un flag se il valore del soggetto **è uguale** al valore di riferimento.
- `<lv_obj-bind_flag_if_not_eq>` — Imposta un flag se il valore del soggetto **non è uguale** al valore di riferimento.
- `<lv_obj-bind_flag_if_gt>` — Imposta un flag se il valore del soggetto **è maggiore del** valore di riferimento.
- `<lv_obj-bind_flag_if_ge>` — Imposta un flag se il valore del soggetto **è maggiore o uguale al** valore di riferimento.
- `<lv_obj-bind_flag_if_lt>` — Imposta un flag se il valore del soggetto **è inferiore al** valore di riferimento.
- `<lv_obj-bind_flag_if_le>` — Imposta un flag se il valore del soggetto **è inferiore o uguale al** valore di riferimento.
- `<lv_obj-bind_state_if_eq>` — Imposta uno stato se il valore del soggetto **è uguale al** il valore di riferimento.
- `<lv_obj-bind_state_if_not_eq>` — Imposta uno stato se il valore del soggetto **è non uguale al** valore di riferimento.
- `<lv_obj-bind_state_if_gt>` — Imposta uno stato se il valore del soggetto **è maggiore del** valore di riferimento.
- `<lv_obj-bind_state_if_ge>` — Imposta uno stato se il valore del soggetto **è maggiore o uguale al** valore di riferimento.
- `<lv_obj-bind_state_if_lt>` — Imposta uno stato se il valore del soggetto **è minore del** valore di riferimento.
- `<lv_obj-bind_state_if_le>` — Imposta uno stato se il valore del soggetto **è minore o uguale al** valore di riferimento.

Nota: Il prefisso `lv_obj-` può essere omesso. Ad esempio, si può semplicemente scrivere `<bind_state_if_gt>`.

Eventi Relativi al Subject

Oltre a collegare le proprietà ai subject, è anche possibile aggiungere eventi che modificano il valore di un subject quando viene premuto, rilasciato, ecc.

Per saperne di più, consultare [Impostare il Valore di Subject](#) and [Incrementare il Valore di Subject](#).

8.8.2 Test dell'interfaccia utente

Panoramica

Il modulo di test XML è un modo potente e flessibile per definire test funzionali dell'interfaccia utente.

I file XML di test sono simili ai componenti, ma sono racchiusi nel tag `<test>` e sono composti da due parti principali:

- **Definizione dell'Interfaccia Utente:** Usa `<styles>`, `<consts>` e `<view>` per definire l'aspetto dell'interfaccia utente. Questo è identico alla struttura dei `<component>`.
- **Passi del Test:** Racchiusi nel tag `<steps>`, definiscono le azioni e le asserzioni per il test.

Tipi di Passi

Sono attualmente supportati i seguenti tipi di passi (con parametri di esempio):

- `<move_to x="10" y="30"/>`: Sposta il mouse alle coordinate dello schermo specificate.
- `<click_at x="10" y="30"/>`: Simula un clic alle coordinate dello schermo specificate.
- `<press/>`: Preme sulla coordinata dello schermo corrente.
- `<release/>`: Rilascia sulla coordinata dello schermo corrente.
- `<wait ms="100"/>`: Attende il numero di millisecondi specificato. LVGL continua a funzionare, incluse animazioni e timer.
- `<freeze ms="100"/>`: Mette in pausa l'interfaccia utente e il tempo interno di LVGL. Utile per il debug visivo.
- `<subject_set subject="subject1" value="5">` Imposta un valore intero o stringa per l'oggetto.
- `<subject_compare subject="subject1" value="10">` Confronta un valore intero o stringa per l'oggetto con un valore di riferimento.
- `<screenshot_COMPARE path="path/to/image.png"/>`: Confronta la schermata corrente con un'immagine di riferimento. - Se l'immagine non esiste, viene creata. - Se il confronto fallisce, viene salvata un'immagine con suffisso `_err` per l'ispezione.

Esempio:

```
<test width="300" height="400">
  <view width="100%" height="100%" flex_flow="column">
    <lv_checkbox text="First one!" />
    <lv_slider />
  </view>

  <steps>
    <screenshot_COMPARE path="imgs/before.png"/>
    <click_at x="32" y="32"/>
    <subject_COMPARE subject="subject2" value="50"/>
    <click_at x="50" y="64"/>
    <subject_set subject="subject2" value="10"/>
    <wait ms="300"/>
    <screenshot_COMPARE path="imgs/after.png"/>
  </steps>
</test>
```

Registrazione dei Test

I test possono essere registrati in due modi:

- Da file: `lv_xml_test_register_from_file("path/to/test.xml", "A:ref_images")`
- Da stringa: `lv_xml_test_register_from_data(xml_data, "A:ref_images")`

Il secondo parametro specifica un prefisso per i path di confronto degli screenshot.

Per annullare la registrazione dei test, si usa:

- `lv_xml_test_unregister()`

Questo metodo viene richiamato automaticamente quando si registra un nuovo test, quindi, le chiamate manuali sono solitamente superflue.

Esecuzione dei Test

Tutti i passi

Per eseguire il test registrato, si usa `lv_xml_test_run_all(slowdown)`.

Pulirà lo schermo e creerà una nuova istanza della view da testare.

Il parametro `slowdown` controlla la velocità di riproduzione:

- 0: Velocità massima
- 1: Velocità in tempo reale
- 2: Velocità dimezzata
- 10: 10 volte più lento
- ...e così via

`lv_xml_test_run_all()` si blocca fino al completamento di tutti i passaggi. È possibile richiamarlo più volte.

Il valore restituito è il numero di test falliti.

Passo dopo passo

È anche possibile eseguire ogni passo separatamente, uno dopo l'altro.

Per prima cosa, si chiama `lv_xml_test_run_init()` per preparare l'esecuzione dei passi. Pulirà lo schermo e creerà una nuova istanza della view da testare.

Ottiene il numero di passi usando `lv_xml_test_get_step_count()` e poi chiama `lv_xml_test_run_next(slowdown)` altrettante volte.

`lv_xml_test_run_next()` restituisce `true` se il passaggio specificato è stato superato, o `false` se non è stato superato.

Infine, chiama `lv_xml_test_run_stop()` per pulire e uscire dalla modalità di test.

Ottenere i Risultati del Test

Attualmente, solo i passi `screenshot_compare` possono fallire. Tuttavia, è possibile interrogare il risultato di ogni passo:

```
uint32_t step_cnt = lv_xml_test_get_step_count();
for(uint32_t i = 0; i < step_cnt; i++) {
    if(lv_xml_test_get_step_type(i) == LV_XML_TEST_STEP_TYPE_SCREENSHOT_COMPARE) {
        if(lv_xml_test_get_status(i))
            printf("Step %d passed\n", i);
        else
            printf("Step %d failed\n", i);
    } else {
        printf("Step %d is not a screenshot comparison\n", i);
    }
}
```

API

8.8.3 Traduzioni

Panoramica

Il modulo di traduzione XML consente di definire e utilizzare stringhe tradotte direttamente nei file XML.

È basato sul *modulo di traduzione di LVGL*.

Consultarlo per saperne di più sulla selezione della lingua attiva, sul recupero delle traduzioni e sul comportamento delle alternative.

Utilizzo

Esempio di definizione di traduzione XML:

```
<translations languages="en de hu">
    <translation tag="dog"      en="The dog"      de="Der Hund"      hu="A kutya"/>
    <translation tag="cat"      en="The cat"      de="Die Katze"    hu="A cicá"/>
    <translation tag="snake"   en="A snake"     de="Eine Schlange" hu="A kígyó"/>
</translations>
```

Nel tag radice `<translations>`, l'attributo `languages` definisce le lingue disponibili, ad esempio `languages="en de hu"`. I codici delle lingue sono in formato libero, ma si consigliano codici in stile ISO.

Ogni `<translation>` definisce un tag, che funge da chiave di ricerca, e attributi per ciascuna lingua.

Le traduzioni possono essere omesse: verranno applicati dei fallback [alternative] quando necessario. Per maggiori dettagli, vedere [Fallback \[alternative\]](#).

Per registrare le traduzioni XML:

- `lv_xml_register_translation_from_file("path/to/file.xml")`
- `lv_xml_register_translation_from_data(xml_string)`

È possibile registrare più sorgenti XML; queste verranno unite e ricercate collettivamente.

8.9 Tool

8.9.1 CLI

L'editor LVGL fornisce uno strumento a riga di comando basato su Node, `lved-cli.js`, per generare/compilare codice da XML, convalidare progetti, sincronizzare da Figma, confrontare alberi ed eseguire test dell'interfaccia utente headless.

Nota

Build pre-release. Destinato allo sviluppo e alla CI. Scade a fine 2025.

Panoramica

- **Binari:** `lved-cli.js` (script Node) [Download CLI](#)
- **Piattaforma:** Node 18+ consigliato
- **Container engine:** Podman (se non su Windows, Scaricare Podman)

Avvio Rapido

Generare file C e H da XML:

```
lved-cli.js generate path/to/project
```

Compilare (web o node) il Binario runtime per visualizzare in anteprima o testare l'interfaccia utente:

```
lved-cli.js compile path/to/project --target web --start-service
```

Validare il contenuto dei file XML (limita gli errori visualizzati):

```
lved-cli.js validate path/to/project --errorlimit 25
```

Eseguire i <test> dai file XML

```
lved-cli.js run-all-tests path/to/project
lved-cli.js run-test path/to/project tests/test-file.xml
```

Sincronizzare stili e immagini da Figma:

```
lved-cli.js figma-sync path/to/project --start-service
```

Comandi

generate

Generare codice da XML.

```
lved-cli.js generate <project-path> [--start-service]
```

compile

Compila per un target.

```
lved-cli.js compile <project-path> [--start-service] [--target <web|node>]
```

- --target per default è web (scelte: web, node)

compare

Confronta due directory (presenza + contenuto normalizzato).

```
lved-cli.js compare <first-project-path> <second-project-path>
```

figma-sync

Sincronizza il progetto con Figma.

```
lved-cli.js figma-sync <project-path> [--start-service]
```

validate

Convalida XML e limita gli errori visualizzati.

```
lved-cli.js validate <project-path> [--errorlimit <n>] [--start-service]
```

- --errorlimit per default è 10 (min 1)

run-test

Esegue test da un singolo file.

```
lved-cli.js run-test <project-path> <testing-file>
```

run-all-tests

Rileva ed esegue tutti i file `test*.xml`.

```
lved-cli.js run-all-tests <project-path>
```

Argomenti Comuni

- <project-path> – path di un progetto LVGL Editor
- <testing-file> – XML di test relativo a <project-path> (run-test)

Opzioni

- `-ss, --start-service` – prepara/aggiorna l'immagine del container prima dell'esecuzione
- `--target <web|node>` – target della compilazione (compile)
- `-l, --errorlimit <n>` – numero massimo di errori visualizzati (validate)

Esempi

Confronto con un albero di riferimento:

```
lved-cli.js compare build/generated ./ci/reference
```

Compilazione per Node:

```
lved-cli.js compile ./examples/my-project --target node --start-service
```

CI/CD

Vedere un esempio di [workflow GitHub Actions](#)

8.9.2 Condivisione Online

Panoramica

Online Share è uno strumento complementare all'Editor di LVGL per semplificare la condivisione delle interfacce utente sviluppate.

Utilizzando Online Share, i repository GitHub possono essere visualizzati e persino modificati nel browser senza dover configurare alcun ambiente di sviluppo.

Online Share supporta tutte le funzionalità dell'Editor (Soggetti, Traduzioni, Animazioni, ecc.) tranne la compilazione del codice, in modo che le modifiche al codice C non possano essere applicate.

Sebbene nel browser sia possibile visualizzare e modificare i file XML, non è possibile salvare le modifiche. In futuro, verranno aggiunti i pulsanti "Commit and Push to GitHub" e "Open Pull Request" per semplificare il flusso di lavoro.

Utilizzo

Online Share è disponibile all'indirizzo <https://viewer.lvgl.io>

Nella schermata iniziale, sono disponibili progetti di esempio e tutorial che possono essere aperti e testati direttamente nel browser.

Nel campo di input "Open Project", è possibile incollare l'URL di un repository GitHub o di una cartella e, cliccando su "Load Project", in pochi secondi verrà aperto.

La cartella a cui punta l'URL deve contenere i file `project.xml` e `globals.xml`.

Alcuni URL di esempio:

- <https://github.com/ORG/REPO>: apre la root del repository sul branch di default
- <https://github.com/ORG/REPO/tree/BRANCH>: apre la root del repository da un branch specifico
- <https://github.com/ORG/REPO/tree/BRANCH/FOLDER>: apre una cartella del repository da un branch specifico

Licenza

Online Share è disponibile gratuitamente per i repository pubblici. Tuttavia, per utilizzarlo per i repository privati, è richiesto un abbonamento.

In entrambi i casi, è necessario accedere con un account GitHub per evitare i limiti di velocità delle API e accedere ai repository privati, se necessario.

Repository Pubblici

Utilizzare Online Share con i repository pubblici è molto semplice: basta incollare l'URL del repository e premere il pulsante "Load Project".

È un ottimo modo per:

- provare Online Share
- utilizzare l'Editor per scopi didattici
- utilizzare l'Editor per progetti open source

Repository Privati

L'abbonamento è basato sul repository e può essere acquistato per un mese o un anno. Finché si ha un abbonamento valido, non ci sono limiti al numero di visitatori. In pratica, chiunque abbia accesso al repository privato può accedere con un account GitHub e visualizzarne il contenuto.

8.9.3 Integrazione con Figma

Panoramica

Il plugin [Figma](#) di LVGL Provisualizza le proprietà di stile degli elementi dell'interfaccia utente selezionati in Figma. Queste proprietà possono essere facilmente copiate e incollate nell'Editor come attributi di stile, consentendo un'implementazione di design pixel-perfect.

Il plugin mostra anche lo "ID nodo Figma" dell'elemento selezionato, consentendo la sincronizzazione degli stili. Ciò significa che, con un solo clic nell'Editor o con un comando CLI, le proprietà di stile di Figma possono essere scaricate e aggiornate nei file XML.

Poiché non si tratta di un importatore automatico magico, lo sviluppatore mantiene il pieno controllo su come vengono organizzati file, stili, componenti, ecc. Allo stesso tempo, il progettista è libero di lavorare in Figma senza vincoli. Lo sviluppatore dovrebbe assumersi la responsabilità di ristrutturare il design con una mentalità da sviluppatore per garantire una struttura manutenibile, pulita e ottimale.

Il plugin Figma-LVGL

Il plugin Figma-LVGL è disponibile pubblicamente nello store dei plugin di Figma e può essere aggiunto a progetti sia pubblici che privati, anche senza un abbonamento a Figma.

Per installare il plugin Figma-LVGL, si va al menù "Figma -> Plugins -> Manage plugins" e lo si cerca nell'elenco.

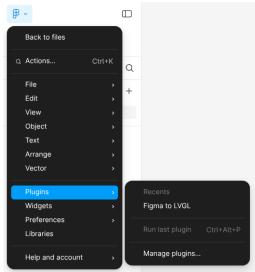


Fig. 1: Cliccare su "Manage Plugins ..." per cercare e installare Figma-LVGL

Se è già stato utilizzato in precedenza, apparirà tra i "Recents".

Una volta installato, apparirà una piccola finestra che mostra le proprietà di stile dell'elemento selezionato.

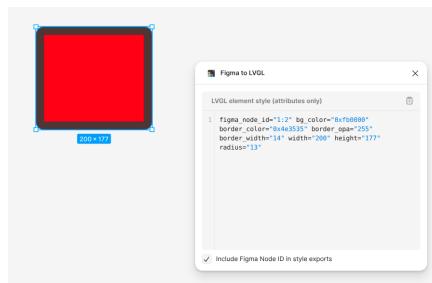


Fig. 2: Elenco delle proprietà di stile dell'elemento UI selezionato da Figma a LVGL

Utilizzo nell'Editor

Le proprietà di stile elencate dal plugin possono essere copiate e incollate in qualsiasi sezione di stile XML. Ad esempio:

```
<component>
  <styles>
    <style name="style_from_figma"
      figma_node_id="1:2" bg_color="0xfb0000"
      border_color="0x4e3535"
      border_opa="255"
      border_width="14"
      width="200"
      height="177"
      radius="13"
    />
  </styles>

  <view>
    <style name="style_from_figma" />
  </view>
</component>
```

Sincronizzazione delle Proprietà di Stile

Se è stata copiata la parte `figma_node_id="..."` dal plugin, lo stile nell'XML è collegato a un elemento specifico in Figma e l'Editor può scaricare le proprietà di stile.

Per abilitare la sincronizzazione, si aggiunge un file `project_local.xml` col seguente contenuto:

```
<project_local>
  <figma document="XXXXXX" token="YYYYY"/>
</project_local>
```

dove:

XXXXXX

è la stringa dopo "design" nell'URL di Figma. Ad esempio, se l'URL di Figma è <https://www.figma.com/design/P8NM0qOqgWfYpbENhAKqYb>, allora XXXXXX sarebbe "P8NM0qOqgWfYpbENhAKqYb".

YYYYY

è il "Personal Access Token" generabile come descritto nell'articolo [Manage Personal Access Tokens](#) nella documentazione di Figma.

Poiché `project_local.xml` contiene un token personale, dovrebbe essere aggiunto a `.gitignore` e memorizzato solo localmente.

Dopodiché, cliccando sul pulsante FigmaSync (in alto a destra), ogni stile verrà aggiornato con un `figma_node_id` dall'ultimo design di Figma.

Utilizzo con la CLI

Lo strumento `CLI tool` può anche sincronizzare gli stili da Figma in CI/CD:

```
lvgl-cli.js figma-sync path/to/project -start-service
```

CAPITOLO 9

Moduli Ausiliari

9.1 Esplora File

`lv_file_explorer` fornisce un'interfaccia utente che consente all'utente finale di esplorare il contenuto di un file system. La sua area principale è chiamata "Browsing Area" [Area di Navigazione] e fornisce l'elenco dei file contenuti nella directory attualmente visualizzata.

Se abilitato, sulla sinistra è presente anche un pannello di "Accesso rapido", che offre un modo comodo per raggiungere le parti del file system a cui si accede di frequente. Le destinazioni "Quick-Access" disponibili sono:

- File System,
- HOME,
- Video,
- Pictures,
- Music, e
- Documents.

È possibile specificare a quali path conducono durante l'inizializzazione di `lv_file_explorer`.

`lv_file_explorer` fornisce solo la navigazione dei file e gli eventi causati dall'attività dell'utente (ad esempio, il clic su un file), ma non fornisce le operazioni effettive sui file. Il codice client deve agganciare vari eventi e decidere cosa fare quando vengono generati (ad esempio, un clic o un doppio clic su un file). Le azioni intraprese potrebbero aprire il file, visualizzarlo, inviarlo a un'altra parte dell'applicazione, ecc. `lv_file_explorer` passa il path completo e il nome del file su cui è stato fatto clic alle funzioni di callback dell'evento. Ciò che accade dopo è a discrezione del progettista dell'applicazione.

`lv_file_explorer` utilizza il Widget *Tabella* (`lv_table`) per la "Browsing Area" [Area di navigazione] e il Widget *List* (`lv_list`) per il pannello "Quick-Access" quando è abilitato. Pertanto, la macro `LV_USE_TABLE` deve essere impostata su un valore diverso da zero in `lv_conf.h` per poter utilizzare `lv_file_explorer`, e `LV_USE_LIST` deve essere impostata su un valore diverso da zero per utilizzare il pannello "Quick-Access".

Nota

Per poter utilizzare "File Explorer", è necessario impostare *File System* (`lv_fs_drv`) e conoscere tutte le lettere di unità utilizzate quando si passano i path al file system (descritto di seguito).

9.1.1 Prerequisiti

Se non è stato già fatto, si dovrà imparare a conoscere l'*astrazione del File System* LVGL, poiché deve essere configurata e funzionante affinché File Explorer funzioni.

9.1.2 Utilizzo

Si imposta `LV_USE_FILE_EXPLORER` su un valore diverso da zero in `lv_conf.h`.

Per prima cosa, si usa `lv_file_explorer_create(lv_screen_active())` per creare un File Explorer. La dimensione di default è quella dello schermo. Dopodiché, si può personalizzare lo stile come qualsiasi Widget.

La dimensione del buffer `current_path` è impostata da `LV_FILE_EXPLORER_PATH_MAX_LEN` in `lv_conf.h`.

La gerarchia degli oggetti di un File Explorer appena creato si presenta così:

- File Explorer: occupa l'intera area del Widget genitore, in genere uno Screen (Flex-Flow COLUMN)
 - Container: occupa l'intera area di File Explorer (Flex grow 1)
 - * Quick-Access Panel:
 - Device List: si espande per contenere i sotto-elementi
 - -__``File System``: pulsante
 - Places List: si espande per contenere i sotto-elementi
 - -__``HOME``: pulsante
 - -__``Video``: pulsante
 - -__``Pictures``: pulsante
 - -__``Music``: pulsante
 - -__``Documents``: pulsante
 - * Browser Panel:
 - Header: 14% dell'altezza del Browser Panel
 - -__``Current Path``: label
 - File Table: con 1 colonna, 86% dell'altezza del Browser Panel
 - Campi:
 - * `home_dir = NULL`
 - * `video_dir = NULL`
 - * `pictures_dir = NULL`
 - * `music_dir = NULL`
 - * `docs_dir = NULL`
 - * `fs_dir = NULL`
 - * `current_path = [buffer vuoto]`
 - * `sel_fn` (file selezionato)
 - * `sort` (il default `LV_EXPLORER_SORT_NONE`)

Accesso alle Parti

Questo elenco di funzioni fornisce l'accesso alle parti mostrate nel diagramma sopra:

- `lv_file_explorer_get_selected_file_name(explorer)` (puntatore a una stringa terminata da NULL contenente il path del file selezionato dall'utente; in genere utilizzato all'interno di un evento `LV_EVENT_CLICKED`)

- `lv_file_explorer_get_current_path(explorer)` (puntatore al buffer `current_path` char)
- `lv_file_explorer_get_file_table(explorer)` (puntatore al Widget File Table *Tabella (lv_table)*)
- `lv_file_explorer_get_header(explorer)` (puntatore al Widget Header *Widget Base (lv_obj)*)
- `lv_file_explorer_get_path_label(explorer)` (puntatore al Widget Current Path Label *Label (lv_label)*)
- `lv_file_explorer_get_quick_access_area(explorer)` (puntatore al Quick-Access Panel *Widget Base (lv_obj)*)
- `lv_file_explorer_get_places_list(explorer)` (puntatore al Widget Places List *List (lv_list)*)
- `lv_file_explorer_get_device_list(explorer)` (puntatore al Widget Device List *List (lv_list)*)

Pannello Quick-Access

Il Pannello Quick-Access si comporta come un tipico pannello di navigazione e appare a sinistra, mentre il Pannello Browser appare a destra.

Questo pannello è facoltativo. Se si imposta `LV_FILE_EXPLORER_QUICK_ACCESS` a 0 in `lv_conf.h`, il Pannello Quick-Access non verrà creato. Questo consente di risparmiare solo una piccola quantità di memoria.

Subito dopo la creazione di File Explorer, in genere si usa `lv_file_explorer_set_quick_access_path(explorer, LV_EXPLORER_XXX_DIR, "path")` per impostare il percorso a cui si accederà quando si clicca sui pulsanti nel Pannello Quick-Access, che attualmente è un elenco fisso. I valori corrispondenti da passare come secondo argomento sono i seguenti:

- `LV_EXPLORER_HOME_DIR`
- `LV_EXPLORER_MUSIC_DIR`
- `LV_EXPLORER_PICTURES_DIR`
- `LV_EXPLORER_VIDEO_DIR`
- `LV_EXPLORER_DOCS_DIR`
- `LV_EXPLORER_FS_DIR`

Ordinamento

È possibile utilizzare `lv_file_explorer_set_sort(explorer, LV_EXPLORER_SORT_XX)` per impostare il metodo di "sort" [ordinamento].

Ecco i possibili metodi di ordinamento:

- `LV_EXPLORER_SORT_NONE` (il default)
- `LV_EXPLORER_SORT_KIND`

`lv_file_explorer_get_sort(explorer)` restituisce il metodo di ordinamento corrente.

9.1.3 Eventi

- `LV_EVENT_READY` Inviato all'apertura di una directory, che può verificarsi:
 - all'apertura iniziale di File Explorer,
 - dopo che un utente ha cliccato su un pulsante di navigazione del Pannello Quick-Access, e
 - dopo che l'utente ha cliccato su una directory visualizzata nel Pannello Browser.
- Lo si può utilizzare, ad esempio, per personalizzare l'ordinamento dei file.
- `LV_EVENT_VALUE_CHANGED` Inviato una volta quando si clicca su un elemento (file) nell'elenco dei file del Pannello Browser.

- **LV_EVENT_CLICKED** Inviato due volte quando si fa clic su un elemento nel Pannello Browser: una volta come risultato dell'evento **LV_EVENT_RELEASED** del dispositivo di input e una seconda come risultato dell'evento **LV_EVENT_CLICKED** del dispositivo di input. Questo vale per file, directory e l'elemento "< Back" nel Pannello Browser.

In questi eventi è possibile utilizzare `lv_file_explorer_get_current_path()` per ottenere il path corrente e `lv_file_explorer_get_selected_file_name()` per ottenere il nome del file attualmente selezionato nella funzione di elaborazione degli eventi. Ad esempio:

```
static void file_explorer_event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        char * cur_path = lv_file_explorer_get_current_path(widget);
        char * sel_fn = lv_file_explorer_get_selected_file_name(widget);
        LV_LOG_USER("%s%s", cur_path, sel_fn);
    }
}
```

È anche possibile salvare il **path** e il nome del **file** ottenuti in un array tramite funzioni come `strcpy()` e `strcat()` per un utilizzo successivo.

9.1.4 Esempio

File Explorer Semplice

```
#include "../../lv_examples.h"

#if LV_USE_TABLE && LV_USE_FILE_EXPLORER && (LV_USE_FS_STDIO || LV_USE_FS_POSIX || LV_USE_FS_WIN32 || LV_USE_FS_FATFS) && LV_BUILD_EXAMPLES

#include <stdlib.h>
#include <string.h>

static void file_explorer_event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        const char * cur_path = lv_file_explorer_get_current_path(obj);
        const char * sel_fn = lv_file_explorer_get_selected_file_name(obj);
        LV_LOG_USER("%s%s", cur_path, sel_fn);
    }
}

void lv_example_file_explorer_1(void)
{
    lv_obj_t * file_explorer = lv_file_explorer_create(lv_screen_active());
    lv_file_explorer_set_sort(file_explorer, LV_EXPLORER_SORT_KIND);

#endif LV_USE_FS_WIN32
/* Note to Windows users: the initial "C:" on these paths corresponds to
 * the value of 'LV_FS_WIN32_LETTER' in 'lv_conf.h', and should not be
 * confused with the Windows/DOS drive letter. It is an identifier that
 * is used to enable LVGL to look up the appropriate driver from a list of
 * registered file-system drivers. `lv_fs_win32_init()` happens to use the
 * identifier letter 'C' so "C:" is the driver-identifier-prefix used here.
 * The "C:" following that is indeed the Windows/DOS drive letter and is
 * part of the actual path that gets passed to the OS-level functions.
 *
 * See https://docs.lvgl.io/master/details/main-components/fs.html for details.
 * File Explorer uses 'lv_fs' internally, thus the required prefix in path strings.
 */
lv_file_explorer_open_dir(file_explorer, "C:/");
#if LV_FILE_EXPLORER_QUICK_ACCESS
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_HOME_DIR, "C:/Users/Public/Desktop");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_VIDEO_DIR, "C:/Users/Public/Videos");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_PICTURE_DIR, "C:/Users/Public/Pictures");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_MUSIC_DIR, "C:/Users/Public/Music");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_DOCS_DIR, "C:/Users/Public/Documents");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_FS_DIR, "C:/");
#endif

#else
/* linux */
lv_file_explorer_open_dir(file_explorer, "A:/");

#if LV_FILE_EXPLORER_QUICK_ACCESS
const char * envvar = "HOME";
char home_dir[LV_FS_MAX_PATH_LENGTH];
strcpy(home_dir, "A:/");
/* get the user's home directory from the HOME environment variable*/
strcat(home_dir, getenv(envvar));
LV_LOG_USER("home_dir: %s\n", home_dir);
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_HOME_DIR, home_dir);
char video_dir[LV_FS_MAX_PATH_LENGTH];
strcpy(video_dir, home_dir);
strcat(video_dir, "/Videos");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_VIDEO_DIR, video_dir);
char picture_dir[LV_FS_MAX_PATH_LENGTH];
strcpy(picture_dir, home_dir);
strcat(picture_dir, "/Pictures");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_PICTURE_DIR, picture_dir);
char music_dir[LV_FS_MAX_PATH_LENGTH];
strcpy(music_dir, home_dir);
strcat(music_dir, "/Music");
#endif

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_MUSIC_DIR, music_dir);
char document_dir[LV_FS_MAX_PATH_LENGTH];
strcpy(document_dir, home_dir);
strcat(document_dir, "/Documents");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_DOCS_DIR, document_dir);

lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_FS_DIR, "A:/");
#endif
#endif

lv_obj_add_event_cb(file_explorer, file_explorer_event_handler, LV_EVENT_ALL, NULL);
}

#endif

```

File Explorer di Controllo

```

#include "../../lv_examples.h"

#if LV_USE_TABLE && LV_USE_DROPDOWN && LV_USE_FILE_EXPLORER && (LV_USE_FS_STDIO || LV_USE_FS_POSIX || LV_USE_FS_WIN32 || LV_USE_FS_FATFS) && LV_BUILD_EXAMPLES
#include <stdlib.h>
#include <string.h>

static void file_explorer_event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        const char * cur_path = lv_file_explorer_get_current_path(obj);
        const char * sel_fn = lv_file_explorer_get_selected_file_name(obj);

        LV_LOG_USER("%s", cur_path, sel_fn);
    }
}

#if LV_FILE_EXPLORER_QUICK_ACCESS
static void btn_event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * btn = lv_event_get_target_obj(e);
    lv_obj_t * file_explorer = (lv_obj_t *) lv_event_get_user_data(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        if(lv_obj_has_state(btn, LV_STATE_CHECKED))
            lv_obj_add_flag(file_explorer, LV_OBJ_FLAG_HIDDEN);
        else
            lv_obj_remove_flag(file_explorer, LV_OBJ_FLAG_HIDDEN);
    }
}

static void dd_event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * dd = lv_event_get_target_obj(e);
    lv_obj_t * fe_quick_access_obj = (lv_obj_t *) lv_event_get_user_data(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_dropdown_get_selected_str(dd, buf, sizeof(buf));
        if(strcmp(buf, "NONE") == 0) {
            lv_file_explorer_set_sort(fe_quick_access_obj, LV_EXPLORER_SORT_NONE);
        }
        else if(strcmp(buf, "KIND") == 0) {
            lv_file_explorer_set_sort(fe_quick_access_obj, LV_EXPLORER_SORT_KIND);
        }
    }
}
#endif

void lv_example_file_explorer_2(void)
{
    lv_obj_t * file_explorer = lv_file_explorer_create(lv_screen_active());

#if LV_USE_FS_WIN32
    /* Note to Windows users: the initial "C:" on these paths corresponds to
     * the value of 'LV_FS_WIN32_LETTER' in 'lv_conf.h', and should not be
     * confused with the Windows/DOS drive letter. It is an identifier that
     * is used to enable LVGL to look up the appropriate driver from a list of
     * registered file-system drivers. `lv_fs_win32_init()` happens to use the
     * identifier letter 'C' so "C:" is the driver-identifier-prefix used here.
     * The "C:" following that is indeed the Windows/DOS drive letter and is
     * part of the actual path that gets passed to the OS-level functions.
     *
     * See https://docs.lvgl.io/master/details/main-components/fs.html for details.
     * File Explorer uses 'lv_fs' internally, thus the required prefix in path strings.
     */
    lv_file_explorer_open_dir(file_explorer, "C:C:/");
#endif
#if LV_FILE_EXPLORER_QUICK_ACCESS
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_HOME_DIR, "C:C:/Users/Public/Desktop");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_VIDEO_DIR, "C:C:/Users/Public/Videos");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_PICTURE_DIR, "C:C:/Users/Public/Pictures");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_MUSIC_DIR, "C:C:/Users/Public/Music");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_DOCS_DIR, "C:C:/Users/Public/Documents");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_FS_DIR, "C:C:/");
#endif

#else
    /* linux */
    lv_file_explorer_open_dir(file_explorer, "A:/");
#endif
#if LV_FILE_EXPLORER_QUICK_ACCESS
    const char * envvar = "HOME";
    char home_dir[LV_FS_MAX_PATH_LENGTH];
    strcpy(home_dir, "A:/");
    /* get the user's home directory from the HOME environment variable*/
    strcat(home_dir, getenv(envvar));

```

(continues on next page)

(continua dalla pagina precedente)

```

LV_LOG_USER("home_dir: %s\n", home_dir);
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_HOME_DIR, home_dir);
char video_dir[LV_FS_MAX_PATH_LENGTH];
strcpy(video_dir, home_dir);
strcat(video_dir, "/Videos");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_VIDEO_DIR, video_dir);
char picture_dir[LV_FS_MAX_PATH_LENGTH];
strcpy(picture_dir, home_dir);
strcat(picture_dir, "/Pictures");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_PICTURE_DIR, picture_dir);
char music_dir[LV_FS_MAX_PATH_LENGTH];
strcpy(music_dir, home_dir);
strcat(music_dir, "/Music");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_MUSIC_DIR, music_dir);
char document_dir[LV_FS_MAX_PATH_LENGTH];
strcpy(document_dir, home_dir);
strcat(document_dir, "/Documents");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_DOCS_DIR, document_dir);

lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_FS_DIR, "A:/");

#endif
#endif

lv_obj_add_event_cb(file_explorer, file_explorer_event_handler, LV_EVENT_ALL, NULL);

#if LV_FILE_EXPLORER_QUICK_ACCESS
/*Quick access status control button*/
lv_obj_t * fe_quick_access_obj = lv_file_explorer_get_quick_access_area(file_explorer);
lv_obj_t * fe_header_obj = lv_file_explorer_get_header(file_explorer);
lv_obj_t * btn = lv_button_create(fe_header_obj);
lv_obj_set_style_radius(btn, 2, 0);
lv_obj_set_style_pad_all(btn, 4, 0);
lv_obj_align(btn, LV_ALIGN_LEFT_MID, 0, 0);
lv_obj_add_flag(btn, LV_OBJ_FLAG_CHECKABLE);

lv_obj_t * label = lv_label_create(btn);
lv_label_set_text(label, LV_SYMBOL_LIST);
lv_obj_center(label);

lv_obj_add_event_cb(btn, btn_event_handler, LV_EVENT_VALUE_CHANGED, fe_quick_access_obj);

/*Sort control*/
static const char * opts = "NONE\n"
                           "KIND";

lv_obj_t * dd = lv_dropdown_create(fe_header_obj);
lv_obj_set_style_radius(dd, 4, 0);
lv_obj_set_style_pad_all(dd, 0, 0);
lv_obj_set_size(dd, LV_PCT(30), LV_SIZE_CONTENT);
lv_dropdown_set_options_static(dd, opts);
lv_obj_align(dd, LV_ALIGN_RIGHT_MID, 0, 0);

lv_obj_add_event_cb(dd, dd_event_handler, LV_EVENT_VALUE_CHANGED, file_explorer);
#endif
}

#endif
}

```

Ordinamento Personalizzato

```

#include "../../lv_examples.h"

#if LV_USE_TABLE && LV_USE_FILE_EXPLORER && (LV_USE_FS_STDIO || LV_USE_FS_POSIX || LV_USE_FS_WIN32 || LV_USE_FS_FATFS) && LV_BUILD_EXAMPLES
#include <stdlib.h>
#include <string.h>

static void exch_table_item(lv_obj_t * tb, int16_t i, int16_t j)
{
    const char * tmp;
    tmp = lv_table_get_cell_value(tb, i, 0);
    lv_table_set_cell_value(tb, 0, 2, tmp);
    lv_table_set_cell_value(tb, i, 0, lv_table_get_cell_value(tb, j, 0));
    lv_table_set_cell_value(tb, j, 0, lv_table_get_cell_value(tb, 0, 2));

    tmp = lv_table_get_cell_value(tb, i, 1);
    lv_table_set_cell_value(tb, 0, 2, tmp);
    lv_table_set_cell_value(tb, i, 1, lv_table_get_cell_value(tb, j, 1));
    lv_table_set_cell_value(tb, j, 1, lv_table_get_cell_value(tb, 0, 2));
}

/*Quick sort 3 way*/
static void sort_by_file_kind(lv_obj_t * tb, int16_t lo, int16_t hi)
{
    if(lo >= hi) return;

    int16_t lt = lo;
    int16_t i = lo + 1;
    int16_t gt = hi;
    const char * v = lv_table_get_cell_value(tb, lo, 1);
    while(i <= gt) {
        if(strcmp(lv_table_get_cell_value(tb, i, 1), v) < 0)
            exch_table_item(tb, lt++, i++);
        else if(strcmp(lv_table_get_cell_value(tb, i, 1), v) > 0)
            exch_table_item(tb, i, gt--);
        else
            i++;
    }
    sort_by_file_kind(tb, lo, lt - 1);
    sort_by_file_kind(tb, gt + 1, hi);
}

static void file_explorer_event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

if(code == LV_EVENT_VALUE_CHANGED) {
    const char * cur_path = lv_file_explorer_get_current_path(obj);
    const char * sel_fn = lv_file_explorer_get_selected_file_name(obj);

    LV_LOG_USER("%s%s", cur_path, sel_fn);
}
else if(code == LV_EVENT_READY) {
    lv_obj_t * tb = lv_file_explorer_get_file_table(obj);
    uint16_t sum = lv_table_get_row_count(tb);

    sort_by_file_kind(tb, 0, (sum - 1));
}
}

void lv_example_file_explorer_3(void)
{
    lv_obj_t * file_explorer = lv_file_explorer_create(lv_screen_active());
    /*Before custom sort, please set the default sorting to NONE. The default is NONE.*/
    lv_file_explorer_set_sort(file_explorer, LV_EXPLORER_SORT_NONE);

#if LV_USE_FS_WIN32
    /* Note to Windows users: the initial "C:" on these paths corresponds to
     * the value of 'LV_FS_WIN32 LETTER' in 'lv_conf.h', and should not be
     * confused with the Windows/DOS drive letter. It is an identifier that
     * is used to enable LVGL to look up the appropriate driver from a list of
     * registered file-system drivers. `lv_fs_win32.init()` happens to use the
     * identifier letter 'C' so "C:" is the driver-identifier-prefix used here.
     * The "C:" following that is indeed the Windows/DOS drive letter and is
     * part of the actual path that gets passed to the OS-level functions.
     *
     * See https://docs.lvgl.io/master/details/main-components/fs.html for details.
     * File Explorer uses 'lv_fs' internally, thus the required prefix in path strings.
     */
    lv_file_explorer_open_dir(file_explorer, "C:/");
#endif

#if LV_FILE_EXPLORER_QUICK_ACCESS
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_HOME_DIR, "C:/Users/Public/Desktop");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_VIDEO_DIR, "C:/Users/Public/Videos");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_PICTURE_DIR, "C:/Users/Public/Pictures");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_MUSIC_DIR, "C:/Users/Public/Music");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_DOCS_DIR, "C:/Users/Public/Documents");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_FS_DIR, "C:/");
#endif

#else
    /* linux */
    lv_file_explorer_open_dir(file_explorer, "A:/");
#endif

#if LV_FILE_EXPLORER_QUICK_ACCESS
    const char * envvar = "HOME";
    char home_dir[LV_FS_MAX_PATH_LENGTH];
    strcpy(home_dir, "A:");
    /* get the user's home directory from the HOME environment variable*/
    strcat(home_dir, getenv(envvar));
    LV_LOG_USER("home_dir: %s\n", home_dir);
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_HOME_DIR, home_dir);
    char video_dir[LV_FS_MAX_PATH_LENGTH];
    strcpy(video_dir, "Videos");
    strcat(video_dir, "/Videos");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_VIDEO_DIR, video_dir);
    char picture_dir[LV_FS_MAX_PATH_LENGTH];
    strcpy(picture_dir, home_dir);
    strcat(picture_dir, "/Pictures");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_PICTURE_DIR, picture_dir);
    char music_dir[LV_FS_MAX_PATH_LENGTH];
    strcpy(music_dir, home_dir);
    strcat(music_dir, "/Music");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_MUSIC_DIR, music_dir);
    char document_dir[LV_FS_MAX_PATH_LENGTH];
    strcpy(document_dir, home_dir);
    strcat(document_dir, "/Documents");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_DOCS_DIR, document_dir);

    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_FS_DIR, "A:/");
#endif

    lv_obj_add_event_cb(file_explorer, file_explorer_event_handler, LV_EVENT_ALL, NULL);
}
#endif

```

9.1.5 API

lv_file_explorer.h

lv_file_explorer_private.h

9.2 Frammento

Il "Fragment" è un concetto copiato da [Android](#).

Rappresenta una porzione riutilizzabile dell'interfaccia utente della propria app. Un frammento definisce e gestisce il proprio layout, ha un proprio ciclo di vita e può gestire i propri eventi. Come il frammento di Android, che deve essere ospitato da un'attività o da un altro frammento, il frammento in LVGL deve essere ospitato da un Widget o da un altro frammento. La gerarchia di visualizzazione del frammento diventa parte o si collega alla gerarchia di visualizzazione dell'host.

Questo concetto presenta anche alcune somiglianze con `UiViewController` su iOS.

Il Fragment Manager è un gestore che contiene i riferimenti ai frammenti ad esso associati e ha uno stack interno per consentire la navigazione in avanti e indietro. È possibile utilizzare il gestore dei frammenti per creare facilmente uno stack di navigazione o un'applicazione multi-pannello.

9.2.1 Utilizzo

Abilitare `LV_USE_FRAGMENT` in `lv_conf.h`.

Creare una Classe Fragment

```
struct sample_fragment_t {
    /* IMPORTANT: don't miss this part */
    lv_fragment_t base;
    /* States, object references and data fields for this fragment */
    const char *title;
};

const lv_fragment_class_t sample_cls = {
    /* Initialize something needed */
    .constructor_cb = sample_fragment_ctor,
    /* Create view objects */
    .create_obj_cb = sample_fragment_create_obj,
    /* IMPORTANT: size of your fragment struct */
    .instance_size = sizeof(struct sample_fragment_t),
};
```

Usare `lv_fragment_manager`

```
/* Create fragment instance, and Widgets will be added to container */
lv_fragment_manager_t *manager = lv_fragment_manager_create(container, NULL);
/* Replace current fragment with instance of sample_cls, and init_argument is user defined pointer */
lv_fragment_manager_replace(manager, &sample_cls, init_argument);
```

Navigazione Basata sui Frammenti

```
/* Add one instance into manager stack. View object of current fragment will be destroyed,
 * but instances created in class constructor will be kept.
 */
lv_fragment_manager_push(manager, &sample_cls, NULL);

/* Remove the top most fragment from the stack, and bring back previous one. */
lv_fragment_manager_pop(manager);
```

9.2.2 Esempio

Utilizzo di base dei frammenti

```
/**
 * @file lv_example_fragment_1.c
 * @brief Basic usage of obj fragment
 */
#include "../lv_examples.h"

#if LV_USE_FRAGMENT && LV_BUILD_EXAMPLES

static void sample_fragment_ctor(lv_fragment_t * self, void * args);

static lv_obj_t * sample_fragment_create_obj(lv_fragment_t * self, lv_obj_t * parent);

static void sample_container_delete(lv_event_t * e);

static lv_obj_t * root = NULL;

struct sample_fragment_t {
    lv_fragment_t base;
    const char * name;
};

static const lv_fragment_class_t sample_cls = {
    .constructor_cb = sample_fragment_ctor,
    .create_obj_cb = sample_fragment_create_obj,
    .instance_size = sizeof(struct sample_fragment_t),
};

void lv_example_fragment_1(void)
{
    root = lv_obj_create(lv_screen_active());
    lv_obj_set_size(root, LV_PCT(100), LV_PCT(100));
    lv_fragment_manager_t * manager = lv_fragment_manager_create(NULL);
    /* Clean up the fragment manager before objects in containers got deleted */
    lv_obj_add_event_cb(root, sample_container_delete, LV_EVENT_DELETE, manager);

    lv_fragment_t * fragment = lv_fragment_create(&sample_cls, (void *) "Fragment");
    lv_fragment_manager_replace(manager, fragment, &root);
}

static void sample_fragment_ctor(lv_fragment_t * self, void * args)
{
```

(continues on next page)

(continua dalla pagina precedente)

```

((struct sample_fragment_t *) self)->name = (const char *) args;
}

static lv_obj_t * sample_fragment_create_obj(lv_fragment_t * self, lv_obj_t * parent)
{
    lv_obj_t * label = lv_label_create(parent);
    lv_obj_set_style_bg_opa(label, LV_OPA_COVER, 0);
    lv_label_set_text_fmt(label, "Hello, %s!", ((struct sample_fragment_t *) self)->name);
    return label;
}

static void sample_container_delete(lv_event_t * e)
{
    lv_fragment_manager_t * manager = (lv_fragment_manager_t *) lv_event_get_user_data(e);
    lv_fragment_manager_delete(manager);
}

#endif

```

Esempio di navigazione dello stack

```

/**
 * @file lv_example_fragment_2.c
 * @brief Navigation stack using obj fragment
 */
#include "../lv_examples.h"

#if LV_USE_FRAGMENT && LV_USE_WIN && LV_USE_GRID && LV_BUILD_EXAMPLES

static void sample_fragment_ctor(lv_fragment_t * self, void * args);

static lv_obj_t * sample_fragment_create_obj(lv_fragment_t * self, lv_obj_t * parent);

static void sample_push_click(lv_event_t * e);

static void sample_pop_click(lv_event_t * e);

static void sample_container_delete(lv_event_t * e);

static void sample_fragment_inc_click(lv_event_t * e);

typedef struct sample_fragment_t {
    lv_fragment_t base;
    lv_obj_t * label;
    int depth;
    int counter;
} sample_fragment_t;

static const lv_fragment_class_t sample_cls = {
    .constructor_cb = sample_fragment_ctor,
    .create_obj_cb = sample_fragment_create_obj,
    .instance_size = sizeof(sample_fragment_t),
};

static lv_obj_t * container = NULL;

void lv_example_fragment_2(void)
{
    lv_obj_t * root = lv_obj_create(lv_screen_active());
    lv_obj_set_size(root, LV_PCT(100), LV_PCT(100));
    lv_obj_set_layout(root, LV_LAYOUT_GRID);
    static const int32_t col_dsc[] = {LV_GRID_FR(1), LV_GRID_FR(1), LV_GRID_TEMPLATE_LAST};
    static const int32_t row_dsc[] = {LV_GRID_FR(1), LV_GRID_CONTENT, LV_GRID_TEMPLATE_LAST};
    lv_obj_set_grid_dsc_array(root, col_dsc, row_dsc);
    container = lv_obj_create(root);
    lv_obj_remove_style_all(container);
    lv_obj_set_grid_cell(container, LV_GRID_ALIGN_STRETCH, 0, 2, LV_GRID_ALIGN_STRETCH, 0, 1);

    lv_obj_t * push_btn = lv_button_create(root);
    lv_obj_t * push_label = lv_label_create(push_btn);
    lv_label_set_text(push_label, "Push");

    lv_obj_t * pop_btn = lv_button_create(root);
    lv_obj_t * pop_label = lv_label_create(pop_btn);
    lv_label_set_text(pop_label, "Pop");
    lv_obj_set_grid_cell(push_btn, LV_GRID_ALIGN_START, 0, 1, LV_GRID_ALIGN_CENTER, 1, 1);
    lv_obj_set_grid_cell(pop_btn, LV_GRID_ALIGN_END, 1, 1, LV_GRID_ALIGN_CENTER, 1, 1);

    lv_fragment_manager_t * manager = lv_fragment_manager_create(NULL);
    /* Clean up the fragment manager before objects in containers got deleted */
    lv_obj_add_event_cb(root, sample_container_delete, LV_EVENT_DELETE, manager);

    int depth = 0;
    lv_fragment_t * fragment = lv_fragment_create(&sample_cls, &depth);
    lv_fragment_manager_push(manager, fragment, &container);
    lv_obj_add_event_cb(push_btn, sample_push_click, LV_EVENT_CLICKED, manager);
    lv_obj_add_event_cb(pop_btn, sample_pop_click, LV_EVENT_CLICKED, manager);
}

static void sample_fragment_ctor(lv_fragment_t * self, void * args)
{
    LV_UNUSED(args);
    ((sample_fragment_t *) self)->depth = *((int *) args);
    ((sample_fragment_t *) self)->counter = 0;
}

static lv_obj_t * sample_fragment_create_obj(lv_fragment_t * self, lv_obj_t * parent)
{
    sample_fragment_t * fragment = (sample_fragment_t *) self;
    lv_obj_t * content = lv_obj_create(parent);
    lv_obj_remove_style_all(content);
    lv_obj_set_style_bg_opa(content, LV_OPA_50, 0);
    lv_obj_set_style_bg_color(content, lv_palette_main(LV_PALETTE_YELLOW), 0);
    lv_obj_set_size(content, LV_PCT(100), LV_PCT(100));
    lv_obj_set_flex_flow(content, LV_FLEX_FLOW_COLUMN);
    lv_obj_t * depth = lv_label_create(content);
    lv_label_set_text_fmt(depth, "Depth: %d", fragment->depth);
    lv_obj_t * label = lv_label_create(content);

```

(continues on next page)

(continua dalla pagina precedente)

```

fragment->label = label;
lv_label_set_text_fmt(label, "The button has been pressed %d times", fragment->counter);

lv_obj_t * inc_btn = lv_button_create(content);
lv_obj_t * inc_label = lv_label_create(inc_btn);
lv_label_set_text(inc_label, "+1");
lv_obj_add_event_cb(inc_btn, sample_fragment_inc_click, LV_EVENT_CLICKED, fragment);

return content;
}

static void sample_push_click(lv_event_t * e)
{
    lv_fragment_manager_t * manager = (lv_fragment_manager_t *) lv_event_get_user_data(e);
    size_t stack_size = lv_fragment_manager_get_stack_size(manager);
    lv_fragment_t * fragment = lv_fragment_create(&sample_cls, &stack_size);
    lv_fragment_manager_push(manager, fragment, container);
}

static void sample_pop_click(lv_event_t * e)
{
    lv_fragment_manager_t * manager = (lv_fragment_manager_t *) lv_event_get_user_data(e);
    lv_fragment_manager_pop(manager);
}

static void sample_container_delete(lv_event_t * e)
{
    lv_fragment_manager_t * manager = (lv_fragment_manager_t *) lv_event_get_user_data(e);
    lv_fragment_manager_delete(manager);
}

static void sample_fragment_inc_click(lv_event_t * e)
{
    sample_fragment_t * fragment = (sample_fragment_t *) lv_event_get_user_data(e);
    fragment->counter++;
    lv_label_set_text(fragment->label, "The button has been pressed %d times", fragment->counter);
}

#endif

```

9.2.3 API

lv_fragment.h

lv_fragment_private.h

9.3 Proprietà Widget

Le Proprietà Widget offrono un modo per ridurre notevolmente le dimensioni dell'interfaccia tra LVGL e qualsiasi livello logico immediatamente superiore, per ottenere e impostare le proprietà più importanti dei Widget. Lo scopo è:

- semplificare (riducendo i tempi di sviluppo) la scrittura di binding per LVGL in un altro linguaggio, come:
 - Micropython,
 - Lua,
 - Python,
 - Perl,
 - .NET
- rendere possibile il controllo dell'interfaccia utente (o parti di essa, ad esempio l'animazione) tramite input esterno, senza modificare il firmware, come:
 - un file di testo esterno (YAML, JSON, XML, personalizzato)
 - qualsiasi sorgente di input esterna (ad esempio seriale)

Sebbene il suo utilizzo consumi più spazio di programma e un maggiore sovraccarico della CPU durante l'impostazione e l'ottenimento delle proprietà del Widget, è progettato in modo da ridurre al minimo tale sovraccarico aggiuntivo della CPU.

9.3.1 Cos'è una Proprietà di un Widget?

Le proprietà di un Widget sono l'insieme combinato di *Stili* più ulteriori proprietà, uniche per ogni tipo di Widget, che determinano l'aspetto e il comportamento del Widget. Esempi: dimensione, posizione e colore sono proprietà di tutti i Widget, mentre testo, modalità estesa, inizio selezione e fine selezione sono proprietà esclusive dei Widget Label. Anche gli *stili locali* di un Widget sono proprietà valide in questo contesto.

Le proprietà Widget non di stile disponibili per un dato Widget vengono implementate all'inizio del file .c primario del Widget come array di ricerca const da-id-a-puntatore-a-funzione, come in questo esempio per il Widget Label:

```
#if LV_USE_OBJ_PROPERTY
static const lv_property_ops_t properties[] = {
{
    .id = LV_PROPERTY_LABEL_TEXT,
    .setter = lv_label_set_text,
    .getter = lv_label_get_text,
},
{
    .id = LV_PROPERTY_LABEL_LONG_MODE,
    .setter = lv_label_set_long_mode,
    .getter = lv_label_get_long_mode,
},
{
    .id = LV_PROPERTY_LABEL_TEXT_SELECTION_START,
    .setter = lv_label_set_text_selection_start,
    .getter = lv_label_get_text_selection_start,
},
{
    .id = LV_PROPERTY_LABEL_TEXT_SELECTION_END,
    .setter = lv_label_set_text_selection_end,
    .getter = lv_label_get_text_selection_end,
},
};
#endif
```

Questo array è allegato al campo `properties` della classe del Widget, quindi tutti i Widget dello stesso tipo condividono lo stesso array di ricerca da-id-a-puntatore-a-funzione.

Alcune proprietà sono di sola lettura. In questo caso, solo il campo `getter` nell'elemento dell'array corrispondente verrà inizializzato con un puntatore a funzione. Esempio: il conteggio dei Widget figli o la posizione di scorrimento di un oggetto devono essere controllati tramite altri tipi di input, ma i loro valori sono leggibili tramite questa API.

9.3.2 Utilizzo

Per default, questa funzionalità di LVGL è disattivata. Può essere attivata configurando `LV_USE_OBJ_PROPERTY` a 1 in `lv_conf.h`.

Le 3 funzioni che diventano disponibili sono:

- `lv_result_t lv_obj_set_property(widget, lv_property_t * value)` Imposta la proprietà specificata del Widget.
- `lv_property_t lv_obj_get_property(widget, lv_prop_id_t id)` Legge il valore della proprietà dal Widget.
- `lv_result_t lv_obj_set_properties(widget, lv_property_t * values, count)` Imposta più proprietà Widget da un array di `lv_property_t`.

Un `lv_prop_id_t` è un *ID Proprietà*, mentre un `lv_property_t` è una struttura che associa un *ID Proprietà* a un *Valore della Proprietà*.

Di seguito è riportato un esempio di array che potrebbe essere utilizzato come argomento `values` in `lv_obj_set_properties()`:

```
lv_property_t values[] = {
    { .id = LV_PROPERTY_IMAGE_SRC, .ptr = &img_demo_widgets_avatar, },
    { .id = LV_PROPERTY_IMAGE_PIVOT, .ptr = &pivot_50, },
    { .id = LV_PROPERTY_IMAGE_SCALE, .num = 128, },
    { .id = LV_PROPERTY_OBJ_FLAG_CLICKABLE, .num = 1, },
    { .id = LV_STYLE_IMAGE_OPA, .num = 128, },
    { .id = LV_STYLE_BG_COLOR, .color = (lv_color_t){.red = 0x11, .green = 0x22, .blue = 0x33}, },
}
```

In alternativa, `lv_obj_set_property(widget, value)` potrebbe essere chiamata utilizzando i singoli elementi `value` di questo array all'interno di un ciclo.

ID Proprietà

`lv_prop_id_t` identifica la proprietà da ottenere/impostare. Si tratta di un valore enum definito nel file .h primario del Widget in questione. Poiché i nomi effettivi vengono "assemblati" da una macro di concatenazione di stringhe del preprocessore e sono quindi difficili da visualizzare, è possibile trovare i nomi anche nel file .c primario del Widget, nell'array `properties[]` che inizializza i campi `.id` nell'array. Ad esempio, `LV_PROPERTY_LABEL_TEXT` si trova in `lv_label.c`, mentre le proprietà disponibili per tutti i Widget si trovano nella parte superiore del file `lv_obj.c`.

Questo array è allegato alla classe del Widget, consentendo la ricerca delle funzioni "getter" e "setter" per ogni tipo di Widget in cui sono state implementate le proprietà Widget. (Nota: questa operazione viene eseguita internamente, quindi non è necessario farla).

Se la proprietà da impostare o ottenere tramite questa API non è ancora implementata, è possibile aggiungere un ID di proprietà Widget personalizzato seguendo le stesse regole e utilizzando una delle due macro helper presenti in enum nel file .h principale del Widget. In entrambi i casi, il valore "assemblato" è un valore a 32 bit:

- LV_PROPERTY_ID (per valori singoli -- vedere *Valori Singoli* di seguito); i bit <31:28> contengono il tipo di valore della proprietà e i bit <27:0> contengono l'ID della proprietà.
- LV_PROPERTY_ID2 (per valori accoppiati -- vedere *Valori Abbinati* di seguito); i bit <31:28> contengono il tipo per il primo valore della proprietà, i bit <27:24> contengono il tipo per il secondo valore e i bit <23:0> contengono l'ID della proprietà.

Assicurarsi solo che l'ID sia univoco in tutti i Widget.

Notare che `lv_style_prop_t` (i valori dell'enumeratore che iniziano con `LV_PROPERTY_STYLE_...`) sono anche ID di proprietà validi e possono essere utilizzati per impostare o ottenere i valori di stile di un Widget.

Valore della Proprietà

`lv_property_t` è una struttura che inizia con un campo `id` il cui significato è lo stesso dell'ID della proprietà descritto sopra, abbinato a un valore, che è un'unione di tutti i possibili tipi di proprietà, inclusi intero, puntatore e colore. Il campo `value` è anche in grado di contenere i diversi tipi di valore per gli stili. Lo fa essendo un'unione di tutti i diversi tipi che potrebbero essere necessari. L'elenco dei campi "union-ed" al momento della stesura di questo articolo è:

Valori Singoli

```
int32_t      num;      /**< Signed integer number (enums or "normal" numbers) */
uint32_t     num_u;    /**< Unsigned integer number (opacity, Booleans) */
bool         enable;   /**< Booleans */
const void*  ptr;      /**< Constant pointers (font, come text, etc.) */
lv_color_t   color;    /**< Colors */
lv_value_precise_t precise; /**< float or int for precise value */
lv_point_t   point;   /**< Point, contains two int32_t */

struct {
    /**
     * Note that place struct member 'style' at first place is intended.
     * 'style' shares same memory with 'num', 'ptr', 'color'.
     * So we set the style value directly without using 'prop.style.num'.
     *
     * E.g.
     *
     * static const lv_property_t obj_pos_x = {
     *     .id = LV_PROPERTY_STYLE_X,
     *     .num = 123,
     *     .selector = LV_STATE_PRESSED,
     * }
     *
     * instead of:
     * static const lv_property_t obj_pos_x = {
     *     .id = LV_PROPERTY_STYLE_X,
     *     .style.num = 123, // note this line.
     *     .selector = LV_STATE_PRESSED,
     * }
     */
    lv_style_value_t style;    /**< Make sure it's the first element in struct. */
    uint32_t        selector;  /**< Style selector, lv_part_t | lv_state_t */
};
```

Valori Abbinati

```
/***
 * For some properties like slider range, it contains two simple (4-byte) values
 * so we can use 'arg1.num' and 'arg2.num' to set the argument.
 */
struct {
    union {
        int32_t      num;
        uint32_t     num_u;
        bool         enable;
        const void*  ptr;
        lv_color_t   color;
        lv_value_precise_t precise;
    } arg1, arg2;
};
```

Si può trovare la struttura `lv_property_t` corrente nel file `lv_obj_property.h`.

Ricerca ID di Proprietà per Nome

Impostando la macro di configurazione `LV_USE_OBJ_PROPERTY_NAME` a 1, le seguenti funzioni possono cercare gli ID proprietà passando il nome della proprietà (una stringa):

- `lv_prop_id_t lv_obj_property_get_id(widget, name)` Ottiene l'ID proprietà cercando ricorsivamente `name` nella gerarchia di classi del Widget e, se non viene trovato, cerca le proprietà di stile.
- `lv_prop_id_t lv_obj_class_property_get_id(class_p, name)` Ottiene l'ID della proprietà eseguendo una ricerca non ricorsiva per `name` direttamente nelle proprietà della classe Widget.
- `lv_prop_id_t lv_style_property_get_id(name)` Ottiene l'ID della proprietà di stile in base al nome.

Le ultime due funzioni sono utili quando si sa già che `name` è tra le proprietà di una specifica classe Widget o è un nome di stile, poiché un nome di proprietà può esistere in entrambi gli elenchi. A causa della sequenza di ricerca in `lv_obj_property_get_id(widget, name)`, se un nome esiste in entrambi gli elenchi, l'utilizzo di questa funzione impone al nome nelle proprietà della gerarchia di classi del Widget di avere la precedenza sul nome dello stile.

È possibile verificare quali nomi sono disponibili esaminando i file .c nella directory `./src/widgets/property/`. Si noti che per supportare le ricerche di nomi binari, questi array vengono generati in modo da garantire che siano in ordine alfabetico. Se è necessario aggiungere una proprietà non presente, si consiglia di aggiungerla in enum nella parte superiore del file .h primario del Widget e di rigenerare questi elenchi utilizzando `./scripts/properties.py` per garantire che l'ordinamento alfabetico venga mantenuto.

9.3.3 Note Aggiuntive

Per l'argomento `lv_property_t * value` della funzione `lv_obj_set_property()`, il linguaggio utilizzato per chiamare tale funzione (ad esempio in una libreria statica o caricata dinamicamente) potrebbe richiedere codice aggiuntivo per convertire i valori dal loro tipo di dati locale (ad esempio, dict, table, ecc.) a una struttura C prima di passarli alla funzione `lv_obj_set_property()`.

9.3.4 API

`lv_obj_property_names.h`

9.4 Traduzione

9.4.1 Panoramica

LVGL supporta due metodi per gestire le traduzioni:

- `lv_i18n`: Uno strumento completo che estrae stringhe traducibili da file C in file YAML e genera da essi file di traduzione in C. Supporta anche le forme plurali. Vedere il suo README per i dettagli.
- `lv_translation`: Una soluzione più semplice ma flessibile che consente di aggiungere traduzioni staticamente o dinamicamente. Questo è il metodo documentato qui.

9.4.2 Aggiungere Traduzioni

Traduzioni Statiche

Se la maggior parte delle traduzioni è nota in fase di compilazione, è possibile definirle utilizzando array di stringhe:

```
/* Arrays are defined 'const' to place them in program space instead of RAM. */
static const char * const languages[] = {"en", "de", "es", NULL};
static const char * const tags[] = {"tiger", "lion", "rabbit", "elephant", NULL};
static const char * const translations[] = {
    "The Tiger", "Der Tiger", "El Tigre",
    "The Lion", "Der Löwe", "El León",
    "The Rabbit", "Das Kaninchen", "El Conejo",
    "The Elephant", "Der Elefant", "El Elefante",
};

lv_translation_add_static(languages, tags, translations);
```

Questo metodo utilizza solo una piccola quantità di RAM aggiuntiva, poiché vengono memorizzati solo i puntatori alle stringhe.

Traduzioni Dinamiche

Se le traduzioni sono disponibili solo in fase di esecuzione (ad esempio, da file, porte seriali o fonti online), possono essere aggiunte dinamicamente.

Questo approccio prevede l'allocazione di memoria. Consultare l'esempio in fondo a questa pagina come riferimento.

9.4.3 Selezionare una Lingua

Una volta registrate le traduzioni, si usa:

```
lv_translation_set_language("language")
```

per impostare la lingua corrente. Il parametro deve corrispondere a uno dei nomi di lingua forniti durante la registrazione.

9.4.4 Traduzione di Stringhe

Per recuperare una traduzione per un determinato tag, si usa:

- `lv_translation_get("tag")`
- o la sua abbreviazione: `lv_tr("tag")`

Questi restituiscono una stringa tradotta che può essere utilizzata con i widget:

```
lv_label_set_text(label, lv_tr("settings"));
lv_dropdown_set_options(dd, lv_tr("color_list"));
```

Fallback [alternativa]

Se un tag esiste ma manca la traduzione per la lingua selezionata, verrà restituito il tag stesso.

Anche se il tag non viene trovato, verrà utilizzato come alternativa.

9.4.5 Aggiornamento Dinamico del Testo dell'Interfaccia Utente

Quando viene chiamata `lv_translation_set_language("language")`, LVGL invia `LV_EVENT_TRANSLATION_LANGUAGE_CHANGED` a ogni widget, consentendo di aggiornare automaticamente il testo.

La nuova lingua può essere recuperata chiamando `lv_translation_get_language()` o ottenendo il parametro evento nella callback dell'evento con `lv_event_get_param(e)`

Esempio di Base

```
static void on_language_change(lv_event_t * e)
{
    lv_obj_t * label = lv_event_get_target_obj(e);
    const char * tag = (const char *) lv_event_get_user_data(e);
    lv_label_set_text(label, lv_tr(tag));

    /* You can get the new language with `lv_event_get_param`/
    /* const char * language = (const char *) lv_event_get_param(e); */
    /* or with `lv_translation_get_language` */
    /* const char * language = lv_translation_get_language(); */
}

lv_obj_t * label = lv_label_create(lv_screen_active());
lv_obj_add_event_cb(label, on_language_change, LV_EVENT_TRANSLATION_LANGUAGE_CHANGED, "tag1");
```

In fondo a questa pagina c'è un esempio completo.

9.4.6 Esempio

Semplice esempio di traduzione

```
#include "../../lv_examples.h"
#ifndef LV_USE_TRANSLATION || !LV_BUILD_EXAMPLES
static void add_static(void)
{
    /* Arrays are defined `const` to place them in program space instead of RAM. */
    static const char * const languages[] = {"en", "de", "es", NULL};
    static const char * const tags[] = {"tiger", "lion", "rabbit", "elephant", NULL};
}
```

(continues on next page)

(continua dalla pagina precedente)

```

static const char * const translations[] = {
    "The Tiger", "Der Tiger", "El Tigre",
    "The Lion", "Der Löwe", "El León",
    "The Rabbit", "Das Kaninchen", "El Conejo",
    "The Elephant", "Der Elefant", "El Elefante",
};

lv_translation_add_static(languages, tags, translations);
}

static void add_dynamic(void)
{
    lv_translation_pack_t * pack = lv_translation_add_dynamic();
    lv_translation_add_language(pack, "en");
    lv_translation_add_language(pack, "de");

    lv_translation_tag_dsc_t * tag;
    tag = lv_translation_add_tag(pack, "table");
    lv_translation_set_tag_translation(pack, tag, 0, "It's a table");
    lv_translation_set_tag_translation(pack, tag, 1, "Das ist ein Tisch");

    tag = lv_translation_add_tag(pack, "chair");
    lv_translation_set_tag_translation(pack, tag, 0, "It's a chair");
    lv_translation_set_tag_translation(pack, tag, 1, "Das ist ein Stuhl");
}

/**
 * Create and use translations
 */
void lv_example_translation_1(void)
{
    add_static();
    add_dynamic();

    lv_translation_set_language("de");

    lv_obj_t * label;

    label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, lv_tr("tiger"));

    label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, lv_tr("chair"));
    lv_obj_set_y(label, 50);
}

#endif /*LV_USE_TRANSLATION & LV_BUILD_EXAMPLES*/

```

Selezione dinamica della lingua

```

#include "../../lv_examples.h"

#if LV_USE_TRANSLATION && LV_USE_DROPDOWN && LV_USE_LABEL && LV_BUILD_EXAMPLES

/* Arrays are defined 'const' to place them in program space instead of RAM. */
static const char * const tags[] = {"tiger", "lion", "rabbit", "elephant", NULL};
static const char * const languages[] = {"English", "Deutsch", "Español", NULL};

static void add_static_translations(void)
{
    static const char * const translations[] = {
        "The Tiger", "Der Tiger", "El Tigre",
        "The Lion", "Der Löwe", "El León",
        "The Rabbit", "Das Kaninchen", "El Conejo",
        "The Elephant", "Der Elefant", "El Elefante",
    };

    lv_translation_add_static(languages, tags, translations);
}

static void on_language_change(lv_event_t * e)
{
    lv_obj_t * label      = lv_event_get_target_obj(e);
    const char * tag      = (const char *) lv_event_get_user_data(e);
    /* You can get the new language with `lv_event_get_param`*/
    const char * language = (const char *) lv_event_get_param(e);
    LV_UNUSED(language);

    lv_label_set_text(label, lv_tr(tag));
}

static void language_change_cb(lv_event_t * e)
{
    static char selected_lang[20];

    lv_obj_t * dropdown = lv_event_get_target_obj(e);
    lv_dropdown_get_selected_str(dropdown, selected_lang, sizeof(selected_lang));
    lv_translation_set_language(selected_lang);
}

/**
 * Change label text when the translation language changes
 */
void lv_example_translation_2(void)
{
    lv_obj_set_flex_flow(lv_screen_active(), LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(lv_screen_active(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    add_static_translations();
    const size_t tag_count = sizeof(tags) / sizeof(tags[0]) - 1;
    const size_t lang_count = sizeof(languages) / sizeof(languages[0]) - 1;

    /* Create a dropdown to be able to select the language */
    lv_obj_t * language_dropdown = lv_dropdown_create(lv_screen_active());
    lv_dropdown_clear_options(language_dropdown);

    for(size_t i = 0; i < lang_count; ++i) {
        lv_dropdown_add_option(language_dropdown, languages[i], i);
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```
}

lv_obj_add_event_cb(language_dropdown, language_change_cb, LV_EVENT_VALUE_CHANGED, NULL);

/* Create a label for each tag */
for(size_t i = 0; i < tag_count; ++i) {
    lv_obj_t * label = lv_label_create(lv_screen_active());

    /* Bind to the language change event so that we can change the label when the language changes */
    lv_obj_add_event_cb(label, on_language_change, LV_EVENT_TRANSLATION_LANGUAGE_CHANGED, (void *)tags[i]);
}

lv_translation_set_language("English");

#endif /*LV_USE_TRANSLATION && LV_USE_DROPDOWN && LV_USE_LABEL && LV_BUILD_EXAMPLES*/
```

9.4.7 API

lv_translation.h

lv_translation_private.h

CAPITOLO 10

Librerie di Terze Parti

10.1 Supporto per i Font

10.2 Supporto del File System

10.2.1 Interfacce del File System

Il modulo *File System (lv_fs_drv)* di LVGL fornisce un'astrazione che consente di collegare qualsiasi tipo di file system per l'utilizzo con LVGL. I file system già supportati sono:

- **FATFS**
- STDIO (Linux e Windows che utilizzano le funzioni standard del C, ad esempio `fopen`, `fread`)
- POSIX Linux e Windows che utilizzano le funzioni POSIX, ad esempio `open`, `read`)
- WIN32 (Windows che utilizza le funzioni API Win32, ad esempio `CreateFileA`, `ReadFile`)
- MEMFS (legge un file da un buffer di memoria)
- LITTLEFS (un piccolo file system a prova di errore progettato per microcontrollori)
- Arduino ESP LITTLEFS (un piccolo file system a prova di errore progettato per Arduino ESP)
- Arduino SD (consente la lettura e la scrittura su schede SD)

È comunque necessario fornire i driver e le librerie, questa estensione fornisce solo il "ponte" tra LVGL e questi file system.

Utilizzo

In `lv_conf.h` abilitare `LV_USE_FS_...` (impostandone il valore a 1) e assegnare una lettera maiuscola a `LV_FS_..._DRIVER LETTER` (ad esempio 'S'). Se è abilitato più di un file system, ogni driver dovrà avere una lettera identificativa univoca. Dopodiché si potrà accedere ai file utilizzando quella lettera. Esempio con la lettera identificativa del driver 'S':

Path relativo simile a Linux
"S:/path/to/file.txt"

Path assoluto simile a Linux
"S:/path/to/file.txt"

Path relativo simile a Windows

```
"S:C:/path/to/file.txt"
```

Path assoluto simile a Windows

```
"S:C:/path/to/file.txt"
```

Non confondere la lettera identificativa del driver con la "lettera del drive" di Windows/DOS/FAT, che fa parte del path passato alle funzioni a livello di sistema operativo. Per maggiori dettagli, vedere [Lettere Identificative](#).

Il [Cached reading](#) è supportato anche se `LV_FS_..._CACHE_SIZE` è impostato su un valore diverso da zero.

Per ulteriori dettagli, tra cui come creare il supporto per il proprio file system, vedere [File System \(lv_fs_drv\)](#).

10.2.2 Arduino ESP littlefs

LittleFS è un piccolo file system fail-safe progettato per microcontrollori e integrato nel framework Arduino quando utilizzato con ESP32 ed ESP8266.

Per un'introduzione dettagliata, vedere:

- <https://github.com/esp8266/Arduino>
- <https://github.com/espressif/arduino-esp32>

Utilizzo

Abilitare `LV_USE_FS_ARDUINO_ESP_LITTLEFS` e definire una `LV_FS_ARDUINO_ESP_LITTLEFS LETTER` in `lv_conf.h`.

API

10.2.3 Arduino SD

Abilita la lettura e la scrittura su schede SD. Una volta collegata una scheda di memoria SD all'interfaccia SPI della scheda Arduino o Genuino, è possibile creare file, leggerli e scriverci sopra. È anche possibile spostarsi tra le directory sulla scheda SD.

Per un'introduzione dettagliata, vedere:

- <https://www.arduino.cc/reference/en/libraries/sd/>

Utilizzo

Abilitare `LV_USE_FS_ARDUINO_SD` e definire una `LV_FS_ARDUINO_SD LETTER` in `lv_conf.h`. Sarà necessario inizializzare la scheda SD prima che LVGL possa utilizzarla (ad esempio: `SD.begin(0, SPI, 40000000)`).

API

10.2.4 FrogFS

frogfs è un [filesystem](#) di sola lettura per impacchettare una directory di file in un singolo "blob" facilmente distribuibile o integrabile in un firmware. È adatto per impacchettare file XML nell'app. Se si è interessati a utilizzare XML, consultare l'introduzione a [XML in LVGL](#).

Una copia del codice sorgente di frogfs si trova all'interno di LVGL. È stato ridotto a un sottoinsieme delle funzionalità originali per il semplice caricamento di blob in LVGL.

Creare un blob frogfs

Si può creare un blob frogfs da utilizzare in LVGL utilizzando il progetto frogfs upstream. Scaricarlo da [GitHub](#) con `git clone`.

```
git clone https://github.com/jkent/frogfs
cd frogfs
```

Creare un file `frogfs.yaml` al suo interno. Ecco il contenuto minimo richiesto del file. `files` in questo caso è una directory accanto a `frogfs.yaml` che contiene tutti i file da includere nel blob. La struttura delle directory verrà mantenuta.

```
collect:
  - files/*
```

```
rm -rf build_dir
mkdir build_dir
python3 tools/mkfrogfs.py frogfs.yaml build_dir frogfs.bin
```

Se l'operazione ha esito positivo, si avrà `frogfs.bin`, che è il blob del file system da incorporare nell'applicazione LVGL. Un modo per ottenerlo nell'app C è convertire il file binario in un array C.

```
echo const > frogfs_bin.c
xxd -i frogfs.bin >> frogfs_bin.c
```

Questo metodo creerà un file C con due variabili globali chiamate `frogfs_bin` e `frogfs_bin_len` che possono essere dichiarate in altri file C come:

```
extern const unsigned char frogfs_bin[];
extern unsigned int frogfs_bin_len;
```

Consultare il `README.md` di `frogfs` per informazioni sull'utilizzo di CMake per automatizzare le build dei blob.

Utilizzo in LVGL

Impostare `LV_USE_FS_FROGFS` a 1 in `lv_conf.h`. Impostare `LV_FS_FROGFS LETTER` su una lettera come 'F'.

```
extern const unsigned char frogfs_bin[];
lv_fs_frogfs_register_blob(frogfs_bin, "main_blob"); /* returns LV_RESULT_OK if the blob is ok */

lv_obj_t * img = lv_image_create(lv_screen_active());
lv_image_set_src(img, "F:main_blob/my_image.png"); /* the blob is specified by the path prefix */

extern const unsigned char frogfs_bin_2[];
lv_fs_frogfs_register_blob(frogfs_bin_2, "another_blob");

lv_obj_t * img2 = lv_image_create(lv_screen_active());
lv_image_set_src(img2, "F:another_blob/my_image.png");

lv_obj_delete(img);
lv_fs_frogfs_unregister_blob("main_blob"); /* optional ability to unregister blobs */
```

Se si dispone di un blob `frogfs` memorizzato in un *filesystem* diverso accessibile a LVGL, è possibile caricare il blob in RAM e registrarlo da lì. Assicurarsi che ci sia RAM sufficiente per il blob. La lettera 'A' in questo esempio potrebbe essere una scheda SD o un filesystem del sistema operativo, ecc. Vedere gli *altri filesystem disponibili in LVGL*.

```
uint32_t blob_size;
lv_fs_path_get_size("A:blobs/new_blob.bin", &blob_size);
void * blob = malloc(blob_size);
lv_fs_load_to_buf(blob, blob_size, "A:blobs/new_blob.bin");

lv_fs_frogfs_register_blob(blob, "new_blob");
```

Compressione e Minimizzazione

Il `frogfs` in LVGL non supporta la decompressione come il `frogfs` upstream. In genere, i formati di risorse come PNG sono già compressi.

Rispetto alle risorse immagine, i file XML hanno dimensioni relativamente ridotte, anche quando l'interfaccia utente descritta è complessa. In ogni caso, le dimensioni del file XML possono essere ridotte tramite la cosiddetta "minification". Nel caso più semplice, significa rimuovere tutti i commenti e gli spazi vuoti non significativi. Si tratta di una forma di compressione che non richiede una fase di decompressione. L'unico inconveniente è che in caso di errore di analisi verrà registrato il numero di riga errata relativo alla versione non minimizzata. Vedere l'introduzione a *XML in LVGL*.

Si può preparare un blob `frogfs` con file XML minimizzati. Nella directory clone `frogfs`, creare una copia di `tools/transform-html-minifier.js` chiamata `transform-xml-minifier.js` e impostare `const options = { ... };` come segue. Non tutte le "minificazioni" HTML aggressive originali sono XML standard, quindi questo è un sottoinsieme che preserva la sintassi XML standard.

```
const options = {
  keepClosingSlash: true,
  collapseWhitespace: true,
  removeComments: true
};
```

È quindi possibile aggiungere il filtro al file `frogfs.yaml`. Le versioni minificate dei file XML nella directory `files/` in questo esempio verranno inserite nel blob.

```
collect:
  - files/*
filter:
  *.xml:
    - xml-minifier
```

È possibile creare filtri personalizzati molto facilmente. Consultare il file `README.md` di `frogfs` per maggiori informazioni. Qualsiasi file Python o JavaScript in `tools` che inizia con `transform-` verrà trattato come una possibile trasformazione, a cui verrà passato il file da minificare tramite `stdin` e l'output minificato è atteso da `stdout`. La parte `filter-name` di `transform-filter-name.py` deve essere specificata nell'elenco `filter` di `frogfs.yaml` per determinati tipi di file o per tutti i tipi di file.

10.2.5 littlefs

littlefs è una piccola libreria di file system a prova di errore progettata per microcontrollori.

L'estensione `lv_fs_littlefs` è un'interfaccia alla libreria `littlefs`.

Per un'introduzione dettagliata, vedere: <https://github.com/littlefs-project/littlefs> .

Utilizzo

Impostare `LV_USE_FS_LITTLEFS` in `lv_conf.h` a 1 e definire una lettera maiuscola (come un carattere di tipo C) per `LV_FS_LITTLEFS LETTER` nell'intervallo ['A'..'Z'].

Se abilitato, `lv_littlefs_set_handler()` può essere utilizzato per impostare un "mount point" [punto di montaggio].

Esempio

```
#include "lfs.h"

// configuration of the filesystem is provided by this struct
const struct lfs_config cfg = {
    // block device operations
    .read = user_provided_block_device_read,
    .prog = user_provided_block_device_prog,
    .erase = user_provided_block_device_erase,
    .sync = user_provided_block_device_sync,

    // block device configuration
    .read_size = 16,
    .prog_size = 16,
    .block_size = 4096,
    .block_count = 128,
    .cache_size = 16,
    .lookahead_size = 16,
    .block_cycles = 500,
};

// mount the filesystem
int err = lfs_mount(&lfs, &cfg);

// reformat if we can't mount the filesystem
// this should only happen on the first boot
if (err) {
    lfs_format(&lfs, &cfg);
    lfs_mount(&lfs, &cfg);
}

lv_littlefs_set_handler(&lfs);
```

API

Vedere anche: `lvgl/src/libs/fsdrv/lv_fs_littlefs.c`

`lv_fsdrv.h`

10.3 Supporto alle Immagini

10.3.1 Decoder BMP

Questa utility di decodifica BMP consente di utilizzare immagini da file .BMP in LVGL.

Sorgenti della libreria: <https://github.com/caj-johnson/bmp-decoder>

Invece di caricare l'intera immagine in una volta, i pixel BMP vengono letti su richiesta, quindi, l'utilizzo di immagini BMP richiede pochissima RAM.

Se abilitato in `lv_conf.h` impostando `LV_USE_BMP` a 1, LVGL registrerà automaticamente un nuovo decoder di immagini in modo che i file BMP possano essere utilizzati direttamente come sorgenti di immagini. Esempio:

```
lv_image_set_src(my_img, "S: path/to/picture.bmp");
```

Si noti che per aprire le immagini da file è necessario registrare un driver *File System* (`lv_fs_drv`). Seguire le istruzioni in *File System* (`lv_fs_drv`).

Limitazioni

- Sono supportati solo i file BMP. Le immagini BMP come array C (`lv_image_dsc_t`) non lo sono. Questo perché non ci sono differenze pratiche tra il modo in cui i file BMP e il formato immagine di LVGL memorizzano i dati dell'immagine.
- I file BMP possono essere caricati solo da file .BMP. Per memorizzarli in flash, è meglio convertirli in un array C con il [convertitore di immagini di LVGL](#).
- Il formato colore del file BMP deve corrispondere al `LV_COLOR_DEPTH` configurato per il display su cui verrà visualizzato. È possibile utilizzare GIMP per salvare l'immagine nel formato richiesto. Sia RGB888 che ARGB888 funzionano con `LV_COLOR_DEPTH` 32
- Le "palette" [tavolozze] di colori non sono supportate.
- Poiché l'immagine non è caricata per intero, non è possibile ingrandirla o ruotarla.

Esempio

Aprire un'immagine BMP da file

```
#include "../../lv_examples.h"
#if LV_USE_BMP && LV_BUILD_EXAMPLES

/***
 * Open a BMP file from a file
 */
void lv_example_bmp_1(void)
{
    lv_obj_t * img = lv_image_create(lv_screen_active());
    /* Assuming a File system is attached to letter 'A'
     * E.g. set LV_USE_FS_STDIO 'A' in lv_conf.h */
    lv_image_set_src(img, "A:/lvgl/examples/libs/bmp/example_32bit.bmp");
    lv_obj_center(img);
}

#endif
```

API

[lv_bmp.h](#)

10.3.2 Decoder GIF

GIF Decoder è un'estensione LVGL che consente di utilizzare immagini GIF nell'interfaccia utente LVGL.

L'implementazione utilizza la libreria [AnimatedGIF library](#).

Utilizzo

Una volta abilitato in `lv_conf.h` impostando `LV_USE_GIF` a 1, `lv_gif_create(parent)` può essere utilizzato per creare un widget GIF.

Si imposta il formato colore del framebuffer GIF con `lv_gif_set_color_format(widget, color_format)`. Si usa `LV_COLOR_FORMAT_ARGB8888` per le GIF con trasparenza. È impostato su `LV_COLOR_FORMAT_ARGB8888` per default. È possibile risparmiare una quantità significativa di RAM utilizzando un formato colore più piccolo. È possibile salvare una conversione del formato colore durante il rendering corrisponde al formato colore visualizzato. Questa funzione può essere chiamata prima di `lv_gif_set_src()` per impedire l'allocazione del framebuffer ARGB8888 iniziale di default.

I formati colore supportati sono:

- *LV_COLOR_FORMAT_RGB565*
- *LV_COLOR_FORMAT_RGB565_SWAPPED*
- *LV_COLOR_FORMAT_RGB888*
- *LV_COLOR_FORMAT_ARGB8888*

lv_gif_set_src(widget, src) funziona in modo molto simile a *lv_image_set_src()*. Come sorgente, accetta anche immagini come variabili (*lv_image_dsc_t*) o file.

Conversione di File GIF in Array C

La conversione di file GIF in un array di byte non è supportata dal convertitore di immagini online a partire dalla versione LVGL v9.0. Tuttavia, esiste ancora un modo per farlo tramite lo script Python *./scripts/LVGLImage.py*. Questa riga di comando:

```
python ./scripts/LVGLImage.py --cf RAW --ofmt C -o . --name my_gif_image_array input.gif
```

produrrà tutti i byte di *input.gif* in forma inalterata in un array C, pronti per essere utilizzati in questo codice:

```
LV_IMAGE_DECLARE(my_gif_image_array);
lv_obj_t * img;

img = lv_gif_create(lv_screen_active());
lv_gif_set_color_format(img, LV_COLOR_FORMAT_ARGB8888);
lv_gif_set_src(img, &my_gif_image_array);
```

Utilizzo di Immagini GIF da File

Esempio:

```
lv_gif_set_src(widget, "S:/path/to/example.gif");
```

Si noti che è necessario registrare un driver del file system per aprire le immagini da file. Per farlo, seguire le istruzioni in *File System (lv_fs_drv)*.

Requisiti di Memoria

Per decodificare e visualizzare un'animazione GIF sono necessari circa 25 kB di RAM più (dimensione pixel del formato colore + 1) × larghezza immagine × altezza immagine. RGB565 ha una dimensione pixel di 2, RGB888 ha una dimensione pixel di 3 e ARGB8888 ha una dimensione pixel di 4.

Esempio

Aprire un'immagine GIF da file e variabile

```
#include "../../lv_examples.h"
#if LV_USE_GIF && LV_BUILD_EXAMPLES

/***
 * Open a GIF image from a file and a variable
 */
void lv_example_gif_1(void)
{
    LV_IMAGE_DECLARE(img_bulb_gif);
    lv_obj_t * img;

    img = lv_gif_create(lv_screen_active());
    lv_gif_set_color_format(img, LV_COLOR_FORMAT_ARGB8888);
    lv_gif_set_src(img, &img_bulb_gif);
    lv_obj_align(img, LV_ALIGN_LEFT_MID, 20, 0);

    img = lv_gif_create(lv_screen_active());
    lv_gif_set_color_format(img, LV_COLOR_FORMAT_ARGB8888);
    /* Assuming a File system is attached to letter 'A'
     * E.g. set LV_USE_FS_STDIO 'A' in lv_conf.h */
    lv_gif_set_src(img, "A:/lvgl/examples/labs/gif/bulb.gif");
    lv_obj_align(img, LV_ALIGN_RIGHT_MID, -20, 0);
}

#endif
```

API

`lv_gif.h`

10.3.3 Decoder libjpeg-turbo

libjpeg-turbo è un'interfaccia LVGL alla libreria libjpeg-turbo, un codec di immagini JPEG che utilizza istruzioni SIMD per accelerare la compressione e la decompressione JPEG di base su sistemi x86, x86-64, Arm, PowerPC e MIPS, nonché la compressione JPEG progressiva su sistemi x86, x86-64 e Arm.

Su tali sistemi, libjpeg-turbo è generalmente da 2 a 6 volte più veloce di libjpeg, a parità di altre condizioni.

Per un'introduzione dettagliata, vedere: <https://libjpeg-turbo.org>.

Sorgenti della libreria: <https://github.com/libjpeg-turbo/libjpeg-turbo>

Installazione

```
sudo apt install libjpeg-turbo8-dev
```

Aggiungere libjpeg-turbo al Progetto

Cmake:

```
find_package(JPEG REQUIRED)
include_directories(${JPEG_INCLUDE_DIR})
target_link_libraries(${PROJECT_NAME} PRIVATE ${JPEG_LIBRARIES})
```

Utilizzo

Impostare `LV_USE_LIBJPEG_TURBO` in `lv_conf.h` a 1.

Vedere gli esempi seguenti.

Si noti che ogni immagine decodificata deve consumare:

larghezza immagine × altezza immagine × 3

byte di RAM, e deve essere combinata con la funzionalità *Memorizzazione nella cache delle immagini* per garantire che l'utilizzo della memoria rientri in un intervallo ragionevole.

Esempio

Caricare un'immagine JPG

```
#include "../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES
#endif

#if LV_USE_LIBJPEG_TURBO

/** Load a JPG image
 */
void lv_example_libjpeg_turbo_1(void)
{
    lv_obj_t * wp;

    wp = lv_image_create(lv_screen_active());
    /* Assuming a File system is attached to letter 'A'
     * E.g. set LV_USE_FS_STDIO 'A' in lv_conf.h */
    lv_image_set_src(wp, "A:/lvgl/examples/libs/libjpeg_turbo/flower.jpg");
    lv_obj_center(wp);
}

#else

void lv_example_libjpeg_turbo_1(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "LibJPEG-Turbo is not installed");
    lv_obj_center(label);
}

#endif
#endif
```

API

`lv_libjpeg_turbo.h`

10.3.4 Decoder libpng

libpng è un'interfaccia LVGL per la libreria di riferimento PNG ufficiale, che supporta quasi tutte le funzionalità PNG, è estensibile ed è stata ampiamente testata per oltre 28 anni.

Per un'introduzione dettagliata, consultare: <http://www.libpng.org/pub/png/libpng.html>.

Installazione

```
sudo apt install libpng-dev
```

Aggiungere libpng al Progetto

Cmake:

```
find_package(PNG REQUIRED)
include_directories(${PNG_INCLUDE_DIR})
target_link_libraries(${PROJECT_NAME} PRIVATE ${PNG_LIBRARIES})
```

Utilizzo

Impostare `LV_USE_LIBPNG` in `lv_conf.h` a 1.

Vedere gli esempi seguenti.

Si noti che ogni immagine di questo decoder deve occupare

larghezza × altezza × 4

byte di RAM, e deve essere combinata con la funzionalità *Memorizzazione nella cache delle immagini* per garantire che l'utilizzo della memoria rientri in un intervallo ragionevole. L'immagine decodificata viene memorizzata in formato pixel RGBA.

Esempio

Aprire un'immagine PNG da file e variabile

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_LIBPNG

/***
 * Open a PNG image from a file
 */
void lv_example_libpng_1(void)
{
    LV_IMAGE_DECLARE(img_png_demo);
    lv_obj_t * img;

    img = lv_image_create(lv_screen_active());
    lv_image_set_src(img, &img_png_demo);
    lv_obj_align(img, LV_ALIGN_LEFT_MID, 10, 0);

    img = lv_image_create(lv_screen_active());
    /* Assuming a File system is attached to letter 'A'
     * E.g. set LV_USE_FS_STDIO 'A' in lv.conf.h */
    lv_image_set_src(img, "A:/lvgl/examples/libs/libpng/png_demo.png");
    lv_obj_align(img, LV_ALIGN_RIGHT_MID, -10, 0);
}

#else

void lv_example_libpng_1(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "LibPNG is not installed");
    lv_obj_center(label);
}

#endif
#endif
```

API

lv_libpng.h

10.3.5 Decoder LodePNG

LodePNG è un'interfaccia LVGL per la libreria LodePNG, un codificatore e decoder PNG in C e C++, senza dipendenze, che offre un modo alternativo per utilizzare le immagini PNG nell'interfaccia utente LVGL.

Per un'introduzione dettagliata, vedere: <https://github.com/lvandeve/lodepng>.

Se abilitato in *lv_conf.h* impostando *LV_USE_LODEPNG* a 1, LVGL registrerà automaticamente un nuovo decoder di immagini in modo che i file PNG possano essere utilizzati direttamente come sorgente di immagini.

Nota

Per aprire le immagini dai file, è necessario registrare un driver del file system. Per farlo, seguire le istruzioni in *File System (lv_fs_drv)*.

L'intera immagine PNG viene decodificata, quindi

larghezza × altezza × 4

sono richiesti byte di RAM dall'heap LVGL. L'immagine decodificata viene memorizzata in formato pixel RGBA.

Poiché la decodifica delle immagini PNG potrebbe richiedere molto tempo, la funzione *Memorizzazione nella cache delle immagini* di LVGL può essere utile.

Compressione dei File PNG

Il formato file PNG supporta colori True Color (24/32 bit) e colori della tavolozza a 8 bit. Di solito clipart, disegni, icone e grafica semplice vengono memorizzati in formato PNG, che non utilizza l'intero spazio colore, quindi, è possibile comprimere ulteriormente l'immagine utilizzando colori della tavolozza a 8 bit, invece del formato True Color a 24/32 bit. Poiché i dispositivi embedded hanno una memoria limitata (flash), si consiglia di comprimere le immagini.

Un'opzione è quella di utilizzare un sito online gratuito per la compressione dei file PNG, ad esempio Compress PNG: <https://compresspng.com/>

Esempio

Aprire un'immagine PNG da file e variabile

```
#include "../../lv_examples.h"
#if LV_USE_LODEPNG && LV_USE_IMAGE && LV_BUILD_EXAMPLES

/***
 * Open a PNG image from a file and a variable
 */
void lv_example_lodepng_1(void)
{
    LV_IMAGE_DECLARE(img_wink_png);
    lv_obj_t * img;

    img = lv_image_create(lv_screen_active());
    lv_image_set_src(img, &img_wink_png);
    lv_obj_align(img, LV_ALIGN_LEFT_MID, 20, 0);

    img = lv_image_create(lv_screen_active());
    /* Assuming a File system is attached to letter 'A'
     * E.g. set LV_USE_FS_STDIO 'A' in lv_conf.h */
    lv_image_set_src(img, "A:/lvgl/examples/assets/lodepng/wink.png");
    lv_obj_align(img, LV_ALIGN_RIGHT_MID, -20, 0);
}

#endif
```

API

`lv_lodepng.h`

10.3.6 Decoder WebP

libwebp è un'interfaccia LVGL per il formato immagine WebP, un formato immagine moderno che offre una compressione lossless e lossy superiore per le immagini sul web. WebP offre:

- Dimensioni file ridotte rispetto a JPEG e PNG
- Compressione lossy con trasparenza
- Compressione lossless
- Supporto per le animazioni (non ancora supportato nell'integrazione LVGL)

Per ulteriori informazioni, consultare: <https://developers.google.com/speed/webp>

Sorgenti della libreria: <https://github.com/webmproject/libwebp>

Installazione

```
# Linux
sudo apt install libwebp-dev
# macOS
brew install webp
```

Aggiungere libwebp al progetto

Configurazione di CMake:

```
find_package(PkgConfig REQUIRED)
pkg_check_modules(WebP REQUIRED libwebp)
include_directories(${WebP_INCLUDE_DIRS})
target_link_libraries(main ${WebP_LINK_LIBRARIES})
```

Utilizzo

Impostare `LV_USE_LIBWEBP` in `lv_conf.h` a 1.

Requisiti di memoria per le immagini WebP:

- WebP "lossy" [con perdita di dati]: larghezza × altezza × 4 byte (formato ARGB8888)
- WebP "lossless" [senza perdita di dati]: larghezza × altezza × 4 byte (formato ARGB8888)

Per un utilizzo ottimale della memoria, utilizzare la funzionalità *Memorizzazione nella cache delle immagini* di LVGL.

Esempio

Aprire un'immagine WEBP da file e variabile

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_LIBWEBP

/** 
 * Load a WEBP image
 */
void lv_example_libwebp_1(void)
{
    lv_obj_t * img;

    img = lv_image_create(lv_screen_active());
    /* Assuming a File system is attached to letter 'A'
     * E.g. set LV_USE_FS_STDIO 'A' in lv_conf.h */
    lv_image_set_src(img, "A:/vgl/examples/libs/libwebp/rose.webp");
    lv_obj_center(img);
}

#else

void lv_example_libwebp_1(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Libwebp is not installed");
    lv_obj_center(label);
}
#endif

```

(continues on next page)

(continua dalla pagina precedente)

```
#endif
#endif
```

API

lv_libwebp.h

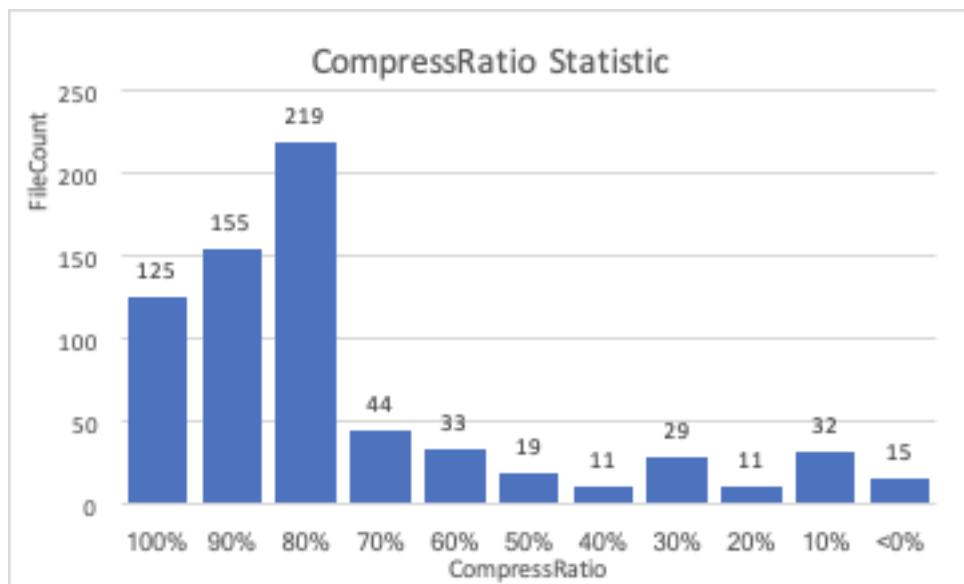
10.3.7 Decompressione RLE

LVGL fornisce un metodo di decompressione RLE (Run-Length Encoding) personalizzato. Può essere utilizzato per ridurre le dimensioni delle immagini binarie. La compressione RLE è un metodo di compressione senza perdita di dati.

Il decoder di immagini binarie nativo in LVGL supporta immagini compresse con RLE. Il decoder supporta sia variabili che file come sorgenti di immagini. I dati binari originali vengono decodificati direttamente nella RAM.

Vantaggi

La maggior parte degli screenshot e delle immagini dell'interfaccia utente (con un numero limitato di colori) può essere compressa per risparmiare oltre il 70% di spazio. Le statistiche seguenti provengono da un progetto di orologio. Mostrano il numero di file per ogni livello di compressione. In rare circostanze, la compressione RLE può aumentare le dimensioni del file se non ci sono grandi ripetizioni nei dati.



Teoria

L'algoritmo RLE è un semplice algoritmo di compressione basato sul fatto che per molti pixel adiacenti il colore è lo stesso. L'algoritmo conta semplicemente quanti pixel ripetuti con lo stesso colore ci sono e memorizza il valore del conteggio e il valore del colore. Se i pixel successivi non sono ripetuti, memorizza il valore del conteggio non ripetuto e i valori del colore originale. Per maggiori dettagli, lo script utilizzato per comprimere l'immagine è disponibile in `lvgl/script/LVGLImage.py`.

```
def rle_compress(self, data: bytearray, blksize: int, threshold=16):
    index = 0
    data_len = len(data)
    compressed_data = []
    while index < data_len:
        memview = memoryview(data)
        repeat_cnt = self.get_repeat_count(
            memview[index:], blksize)
        if repeat_cnt == 0:
            # done
            break
        elif repeat_cnt < threshold:
            nonrepeat_cnt = self.get_nonrepeat_count(
                memview[index:], blksize, threshold)
```

(continues on next page)

(continua dalla pagina precedente)

```

ctrl_byte = uint8_t(nonrepeat_cnt | 0x80)
compressed_data.append(ctrl_byte)
compressed_data.append(
    memview[index: index + nonrepeat_cnt * blksize])
index += nonrepeat_cnt * blksize
else:
    ctrl_byte = uint8_t(repeat_cnt)
    compressed_data.append(ctrl_byte)
    compressed_data.append(memview[index: index + blksize])
    index += repeat_cnt * blksize

return b"".join(compressed_data)

```

Utilizzo

Per utilizzare il decoder RLE, abilitarlo nel file di configurazione `lv_conf.h` impostando `LV_USE_RLE` a 1. L'immagine RLE può quindi essere utilizzata allo stesso modo di altre immagini.

```
lv_image_set_src(img, "path/to/image.rle");
```

Generazione di Immagini Binarie Comprese con RLE

Un'immagine binaria RLE può essere generata direttamente da un'altra immagine utilizzando lo script `lvgl/script/LVGLImage.py`.

```
./scripts/LVGLImage.py --ofmt BIN --cf I8 --compress RLE cogwheel.png
```

Questo decomprimerà `cogwheel.png`, quindi lo ricomprimerà utilizzando RLE e scriverà l'output in `./output/cogwheel.bin`.

API

`lv_rle.h`

10.3.8 Player Rlottie

⚠️ Avvertimento

Rlottie è deprecato. Si consiglia di utilizzare il Widget *Lottie* (`lv_lottie`).

Il [player di animazioni Rlottie per LVGL](#) è un'estensione di terze parti per LVGL che consente di riprodurre le animazioni Lottie in LVGL. Fornisce un'interfaccia all'API C della libreria [Samsung/rllottie](#). Questo player Lottie non fa parte di LVGL; deve essere compilato separatamente.

Compilazione di Rlottie

Per compilare Rlottie di Samsung, è necessario un compilatore compatibile con C++14 e, facoltativamente, CMake 3.14 o superiore.

Per compilare su un desktop, si possono seguire le istruzioni del file README.

Nel caso più semplice, il risultato è questo:

```

mkdir rlottie_workdir
cd rlottie_workdir
git clone https://github.com/Samsung/rllottie.git
mkdir build
cd build
cmake ..../rlottie
make -j
sudo make install

```

E infine, si aggiunge il flag `-lrlottie` al linker.

Sui sistemi embedded, ci si deve occupare di integrare Rlottie nel sistema di Build specificato.

Vedere l'esempio ESP-IDF qui sotto.

Utilizzo

Si può utilizzare l'animazione da file o da dati grezzi (testo). In entrambi i casi, si deve prima abilitare `LV_USE_RLOTTIE` in `lv_conf.h` impostandone il valore a 1.

`width` e `height` del Widget vengono impostate nella funzione `lv_rlottie_create_from_...()` e l'animazione verrà ridimensionata di conseguenza.

Utilizzare Rlottie da File

Per creare un'animazione Lottie da un file, si usa:

```
lv_obj_t * lottie = lv_rlottie_create_from_file(parent, width, height, "path/to/lottie.json");
```

Nota: Rlottie utilizza l'API standard per file C STDIO, quindi è possibile utilizzare il path "normalmente" e non è richiesta alcuna lettera driver specifica per LVGL.

Utilizzare Rlottie da Dati Stringa Grezzi

`lv_example_rlottie_approve.c` contiene un'animazione di esempio in formato raw. Invece di memorizzare la stringa JSON, viene memorizzato un array esadecimale per i seguenti motivi:

- evitare di usare il carattere " nel file JSON
- alcuni compilatori non supportano stringhe molto lunghe

`lvgl/scripts/filetohex.py` può essere utilizzato per convertire un file Lottie in un array esadecimale. Ad esempio:

```
./filetohex.py path/to/lottie.json --filter-character --null-terminate > out.txt
```

`--filter-character` filtra i caratteri non ASCII e `--null-terminate` assicura che venga aggiunto uno zero finale per terminare correttamente la stringa.

Per creare un'animazione da dati grezzi:

```
extern const uint8_t lottie_data[];
lv_obj_t* lottie = lv_rlottie_create_from_raw(parent, width, height, (const char *)lottie_data);
```

Ottenere Animazioni

Lottie è un formato standard e diffuso, quindi è possibile trovare molti file di animazione sul web. Ad esempio: <https://lottiefiles.com/>.

È anche possibile creare animazioni personalizzate con Adobe After Effects o software simili.

Controllare le Animazioni

LVGL fornisce due funzioni per controllare la modalità di animazione: `lv_rlottie_set_play_mode()` e `lv_rlottie_set_current_frame()`. Si combineranno le proprie intenzioni quando si chiamerà il primo metodo, come in questi esempi:

```
lv_obj_t * lottie = lv_rlottie_create_from_file(scr, 128, 128, "test.json");
lv_obj_center(lottie);
// Pause to a specific frame
lv_rlottie_set_current_frame(lottie, 50);
lv_rlottie_set_play_mode(lottie, LV_RLOTTIE_CTRL_PAUSE); // The specified frame will be displayed and then the animation will pause

// Play backward and loop
lv_rlottie_set_play_mode(lottie, LV_RLOTTIE_CTRL_PLAY | LV_RLOTTIE_CTRL_BACKWARD | LV_RLOTTIE_CTRL_LOOP);

// Play forward once (no looping)
lv_rlottie_set_play_mode(lottie, LV_RLOTTIE_CTRL_PLAY | LV_RLOTTIE_CTRL_FORWARD);
```

La modalità di animazione di default è **riproduci in avanti con loop**.

Se non si abilita il loop, viene inviato un `LV_EVENT_READY` quando l'animazione non può procedere oltre senza loop.

Per ottenere il numero di frame in un'animazione o l'indice del frame corrente, è possibile eseguire il "cast" dell'istanza `lv_obj_t` in un'istanza `lv_rlottie_t` e ispezionare i membri `current_frame` e `total_frames`.

Esempio ESP-IDF

Background

Il rendering di Rlottie può essere costoso su hardware embedded. Le animazioni Lottie tendono a utilizzare una grande quantità di CPU e possono richiedere grandi quantità di RAM. Questo varia da Lottie a Lottie, ma in generale per ottenere le migliori prestazioni:

- Limitare il numero totale di frame nell'animazione
- Se possibile, cercare di evitare animazioni di tipo Bézier
- Limitare le dimensioni di rendering dell'animazione

Se il chip ESP32 non dispone di SPIRAM, si incontreranno gravi limitazioni nelle dimensioni di rendering.

Per dare un'idea più precisa, supponiamo di voler eseguire il rendering di un'animazione Lottie di 240x320.

Per superare l'inizializzazione dell'oggetto `lv_rlottie_t`, è necessaria una memoria disponibile di 240x320x32/8 (307k). L'ultimo ESP32-S3 ha 256kb di RAM disponibili per questo (prima che FreeRTOS e qualsiasi altra inizializzazione comincino a prelevare blocchi). Quindi, anche se probabilmente è possibile iniziare a renderizzare un'animazione 50x50 senza SPIRAM, una PSRAM è altamente consigliata.

Inoltre, sebbene sia possibile passare l'inizializzazione dell'oggetto `lv_rlottie_t`, poiché rlottie renderizza frame per frame, questo consuma memoria aggiuntiva. Un'animazione di 30 frame che dura più di 1 secondo probabilmente presenta problemi minimi, ma un'animazione di 300 frame che dura più di 10 secondi potrebbe facilmente bloccarsi a causa della mancanza di memoria durante il rendering di rlottie, a seconda della complessità dell'animazione.

Rlottie non compilerà per l>IDF utilizzando l'opzione del compilatore `-O2` al momento.

Per quanto riguarda la stabilità delle animazioni lottie, l'autore ha scoperto che funzionano meglio nell>IDF quando si abilita `LV_MEM_CUSTOM` (utilizzando `stdlib.h`)

Nonostante tutti i loro difetti, quando si eseguono animazioni di dimensioni corrette, forniscono una meravigliosa utilità per LVGL su LCD embedded e possono avere un aspetto davvero gradevole se eseguite correttamente.

Quando si sceglie/progetta un'animazione lottie, si considerino le seguenti limitazioni:

- Creare l'animazione lottie in modo che abbia le dimensioni desiderate.
- È possibile scalare/ridimensionare, ma le prestazioni saranno migliori quando le dimensioni di base del lottie sono quelle desiderate.
- Limitare il numero totale di frame: più lunga è l'animazione lottie, maggiore sarà la memoria che consumerà per il rendering (rlottie consuma IRAM per il rendering).
- Creare l'animazione lottie per il frame rate desiderato.
- Il lottie di default è 60 fps, gli LCD embedded probabilmente non supereranno i 30 fps.

Configurazione IDF

Se il simulatore LVGL utilizza la libreria rlottie installata, l>IDF funziona al meglio quando si utilizza rlottie come sottomodulo nella directory components.

```
cd 'your/project/directory'
git add submodule
git add submodule https://github.com/Samsung/rlottie.git ./components/rlottie/rlottie
git submodule update --init --recursive
```

Ora, Rlottie è disponibile come componente nell>IDF, ma richiede alcune modifiche aggiuntive e un file CMakeLists per indicare all>IDF come compilare.

File di Patch per Rlottie

Rlottie si basa sul linking dinamico per una libreria di loader di immagini. Questa funzionalità deve essere disabilitata poiché l>IDF non funziona correttamente con il linking dinamico.

Un file di patch è disponibile in LVGL in: /env_support/esp/rlottie/0001-changes-to-compile-with-esp-idf.patch

Applicare il file di patch al sottomodulo rlottie.

CMakeLists per IDF

Un file CMakeLists di esempio è disponibile all'indirizzo /env_support/esp/rlottie/CMakeLists.txt

Copiare questo file CMakeLists in 'your-project-directory'/components/rlottie/

Oltre al file CMakeLists del componente, si dovrà anche specificare al CMakeLists al livello di progetto nel proprio progetto IDF di richiedere rlottie:

```
REQUIRES "lvgl" "rlottie"
```

Da qui, si dovrebbe essere in grado di utilizzare gli oggetti lv_rlottie nel progetto ESP-IDF come qualsiasi altro widget negli esempi ESP LVGL. Ricordare che queste animazioni possono essere fortemente limitate nelle risorse e questo non garantisce che tutte le animazioni funzionino.

Ulteriori considerazioni su Lottie in ESP-IDF

Sebbene non necessaria, la rimozione della cartella rlottie/rlottie/example può rimuovere molti file non necessari per questa applicazione LVGL embedded.

Da qui, è possibile utilizzare le funzioni LVGL `lv_rlottie...()` pertinenti per creare animazioni lottie in LVGL su hardware embedded!

Si prega di notare che, sebbene le animazioni lottie siano in grado di funzionare su molti chip ESP, per prestazioni ottimali si consiglia di seguire le istruzioni riportate di seguito.

- ESP32-S3-WROOM-1-N16R8
 - 16mb quad spi flash
 - 8mb octal spi PSRAM
- IDF4.4 o superiore

Il devkit Esp-box soddisfa questa specifica e <https://github.com/espressif/esp-box> è un ottimo punto di partenza per aggiungere animazioni lottie.

Sarà necessario abilitare `LV_USE_RLOTTIE` tramite `idf.py` di menuconfig nelle impostazioni del componente LVGL.

Ulteriori Modifiche per l'Utilizzo di SPIRAM

`lv_alloc` / `realloc` non utilizza SPIRAM. Dato l'elevato utilizzo di memoria delle animazioni Lottie, si consiglia di spostare il più possibile la memoria dalla DRAM interna alla SPIRAM. Per fare ciò, è necessario abilitare SPIRAM nelle opzioni di menuconfig del chip Espressif in uso.

Potrebbe esserci una soluzione migliore, ma per il momento si consiglia di apportare modifiche locali al componente LVGL nel progetto Espressif. È semplice come scambiare le chiamate `lv_alloc` / `lv_realloc` in `lv_rlottie.c` con `heap_caps_malloc` (per IDF) con la chiamata `MALLOC_CAP_APPROPRIATE` --- per l'utilizzo di SPIRAM questa è `MALLOC_CAP_SPIRAM`.

```
rlottie->allocated_buf = heap_caps_malloc(allocated_buf_size+1, MALLOC_CAP_SPIRAM);
```

Esempio

Caricare un'animazione Lottie da dati grezzi

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_RLOTTIE
/***
 * Load an lottie animation from flash

```

(continues on next page)

(continua dalla pagina precedente)

```
/*
void lv_example_rlottie_1(void)
{
    extern const uint8_t lv_example_rlottie_approve[];
    lv_obj_t * lottie = lv_rlottie_create_from_raw(lv_screen_active(), 100, 100, (const char *)lv_example_rlottie_approve);
    lv_obj_center(lottie);
}

#else
void lv_example_rlottie_1(void)
{
    /*TODO
    *fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Rlottie is not installed");
    lv_obj_center(label);
}

#endif
#endif
```

Caricare un'animazione Lottie da un file

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_RLOTTIE

/**
 * Load an lottie animation from file
 */
void lv_example_rlottie_2(void)
{
    /*The rlottie library uses STUDIO file API, so there is no driver letter for LVGL*/
    lv_obj_t * lottie = lv_rlottie_create_from_file(lv_screen_active(), 100,
                                                    "lvg/examples/libs/rloottie/lv_example_rloottie_approve.json");
    lv_obj_center(lottie);
}

#else
void lv_example_rlottie_2(void)
{
    /*TODO
    *fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Rlottie is not installed");
    lv_obj_center(label);
}

#endif
#endif
```

API

lv_rlottie.h

lv_rlottie_private.h

10.3.9 Decoder SVG

L'estensione `lv_svg` consente di utilizzare immagini SVG nell'interfaccia utente LVGL utilizzando la specifica [Scalable Vector Graphics \(SVG\) Tiny 1.2](#).

Per un'introduzione dettagliata, consultare: <https://www.w3.org/TR/SVGTiny12/>

Utilizzo

Abilitare `LV_USE_SVG` in `lv_conf.h` impostandone il valore a 1.

Vedere gli esempi seguenti.

Se è necessario il supporto per l'analisi degli attributi di animazione SVG, è possibile impostare `LV_USE_SVG_ANIMATION` in `lv_conf.h` a 1.

Come Sorgente Immagine

`lv_image` supporta direttamente le immagini SVG. Ad esempio:

```
lv_image_set_src(widget, "S:/path/to/example.svg");
```

Rendering Diretto

È anche possibile disegnare grafica vettoriale SVG negli eventi di disegno:

```
lv_svg_node_t * svg_doc;
const char* svg_data = "<svg><rect x=\"0\" y=\"0\" width=\"100\" height=\"100\"/></svg>";
/* Create an SVG DOM tree */
svg_doc = lv_svg_load_data(svg_data, svg_len);
...
/* Draw SVG image */
lv_draw_svg(layer, svg_doc);
...
/* Release the DOM tree */
lv_svg_node_delete(svg_doc);
```

Esempio

Caricare e renderizzare dati SVG

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_SVG && LV_USE_VECTOR_GRAPHIC

/***
 * Load an SVG from data
 */
void lv_example_svg_1(void)
{
    static const char svg_data[] = "<svg width=\"12cm\" height=\"4cm\" viewBox=\"0 0 1200 400\">
        <circle cx=\"600\" cy=\"200\" r=\"100\" fill=\"red\" stroke=\"blue\" stroke-width=\"10\"/></svg>";

    static lv_image_dsc_t svg_dsc;
    svg_dsc.header.magic = LV_IMAGE_HEADER_MAGIC;
    svg_dsc.header.w = 450;
    svg_dsc.header.h = 150;
    svg_dsc.data_size = sizeof(svg_data) - 1;
    svg_dsc.data = (const uint8_t *)svg_data;

    lv_obj_t * svg = lv_image_create(lv_screen_active());
    lv_image_set_src(svg, &svg_dsc);
}
#endif

void lv_example_svg_1(void)
{
    /*TODO
     *fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "SVG is not enabled");
    lv_obj_center(label);
}

#endif
#endif
```

Caricare e renderizzare dati SVG da un file

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_SVG && LV_USE_VECTOR_GRAPHIC

/***
 * Load an SVG from a file
 */
void lv_example_svg_2(void)
{
    lv_obj_t * svg = lv_image_create(lv_screen_active());
    lv_image_set_src(svg, "A:lvg/examples/assets/circle.svg");
}
#endif

void lv_example_svg_2(void)
{
    /*TODO
     *fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "SVG is not enabled");
    lv_obj_center(label);
}

#endif
#endif
```

Caricare e renderizzare dati SVG in un evento di disegno

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_SVG && LV_USE_VECTOR_GRAPHIC

/***
 * Draw SVG data in a draw event
 */

```

(continues on next page)

(continua dalla pagina precedente)

```

static void event_cb(lv_event_t * e)
{
    static const char svg_data[] = "<svg width=\"12cm\" height=\"4cm\" viewBox=\"0 0 1200 400\">
        <circle cx=\"600\" cy=\"200\" r=\"100\" fill=\"red\" stroke=\"blue\" stroke-width=\"10\"/></svg>";

    lv_layer_t * layer = lv_event_get_layer(e);
    lv_svg_node_t * svg = lv_svg_load_data(svg_data, sizeof(svg_data) / sizeof(char));
    lv_draw_svg(layer, svg);
    lv_svg_node_delete(svg);
}

void lv_example_svg_3(void)
{
    lv_obj_add_event_cb(lv_screen_active(), event_cb, LV_EVENT_DRAW_MAIN, NULL);
}
#else

void lv_example_svg_3(void)
{
    /*TODO
     *fallback for online examples*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "SVG is not enabled");
    lv_obj_center(label);
}

#endif
#endif

```

API

lv_svg.h
lv_svg_decoder.h
lv_svg_parser.h
lv_svg_render.h
lv_svg_token.h

10.3.10 Decoder Tiny JPEG

Tiny JPEG Decompressor è un'interfaccia LVGL per la libreria TJpgDec, un modulo generico di decompressione di immagini JPEG altamente ottimizzato per piccoli sistemi embedded. Funziona con un consumo di memoria molto basso, quindi può essere integrato in microcontrollori di piccole dimensioni, come AVR, 8051, PIC, Z80, Cortex-M0, ecc.

Per un'introduzione dettagliata, vedere: [TJpgDec](#).

Panoramica

Caratteristiche e restrizioni:

- Il JPEG viene decodificato in riquadri 8x8.
- Sono supportati solo i file JPEG di base (nessun supporto per JPEG progressivi).
- Sono implementate la lettura da file e l'array C.
- Vengono decodificate solo le porzioni richieste delle immagini JPEG, pertanto non è possibile ingrandirle o ruotarle.

Utilizzo

Impostare `LV_USE_TJPGD` a 1 in `lv_conf.h`. LVGL registrerà automaticamente un nuovo decodificatore di immagini in modo che i file JPEG possano essere utilizzati direttamente come sorgenti di immagini.

Ad esempio:

```
lv_image_set_src(my_img, "S:/path/to/picture.jpg");
```

i Nota

Per aprire le immagini dai file, è necessario registrare un driver del file system. Per farlo, seguire le istruzioni in *File System (lv_fs_drv)*.

Convertitore**Conversione di JPEG in array C**

- Utilizzare lo strumento online di lvgl <https://lvgl.io/tools/imageconverter>
- Formato colore = RAW, formato di output = C Array

Esempio**Caricare un'immagine JPG**

```
#include "../../lv_examples.h"
#if LV_USE_TJPGD && LV_BUILD_EXAMPLES

/** 
 * Load a JPG image
 */
void lv_example_tjppgd_1(void)
{
    lv_obj_t * wp;

    wp = lv_image_create(lv_screen_active());
    /* Assuming a File system is attached to letter 'A'
     * E.g. set LV_USE_FS_STDIO 'A' in lv_conf.h */
    lv_image_set_src(wp, "A:test_img_lvgl_logo.jpg");
    lv_obj_center(wp);
}

#endif
```

API**10.4 Supporto Video****10.4.1 Supporto FFmpeg**

FFmpeg è una soluzione completa e multipiattaforma per registrare, convertire e trasmettere in streaming audio e video.

FFmpeg è un'estensione LVGL che si interfaccia con la libreria ufficiale FFmpeg per aiutare ad aggiungere funzionalità di registrazione, conversione e streaming audio e video indipendenti dalla piattaforma nell'interfaccia utente LVGL.

I passaggi di configurazione riportati di seguito sono per Linux, ma possono essere adattati ad altre piattaforme.

Per un'introduzione dettagliata, consultare: <https://www.ffmpeg.org>

Installazione di FFmpeg

```
sudo apt install libavformat-dev libavcodec-dev libswscale-dev libavutil-dev
```

Ottimare scaricare la libreria FFmpeg dalla [pagina di download ufficiale](#), quindi installarla:

```
./configure --disable-all --disable-autodetect --disable-podpages --disable-asym --enable-avcodec --enable-avformat --enable-decoders --enable-encoders --enable-demuxers --enable-parsers --
--enable-protocol='file' --enable-swscale --enable-zlib
make
sudo make install
```

Aggiungere FFmpeg al Progetto

Per utilizzare la libreria FFmpeg in un progetto, si dovranno linkare queste librerie:

libavformat

parte della libreria FFmpeg

libavcodec

parte della libreria FFmpeg

libavutil

parte della libreria FFmpeg

lswscale

parte della libreria FFmpeg

lm**lz****lpthread**

Se si utilizza una toolchain basata su GCC, è possibile risolvere il problema aggiungendo le seguenti opzioni da riga di comando:

```
-lavformat -lavcodec -lavutil -lswscale -lm -lz -lpthread
```

Utilizzo

Impostare `LV_USE_FFMPEG` in `lv_conf.h` a 1.

Impostare anche `LV_FFMPEG_PLAYER_USE_LV_FS` in `lv_conf.h` a 1 integrare l'estensione LVGL *File System* (`lv_fs_drv`) in FFmpeg. Questa libreria può caricare video e immagini. Il file system LVGL verrà sempre utilizzato quando un'immagine viene caricata con `lv_image_set_src()`, indipendentemente dal valore di `LV_FFMPEG_PLAYER_USE_LV_FS`.

Il lettore ffmpeg utilizza la decodifica software per default. Se è necessario un decoder hardware, è necessario specificarlo manualmente utilizzando `lv_ffmpeg_player_set_decoder`, ad esempio `h264_v4l2m2m`.

Vedere gli esempi seguenti per un corretto utilizzo di questa libreria.

Eventi

- `LV_EVENT_READY` Inviato al termine della riproduzione e con riavvio automatico non abilitato.

Scoprire di più sugli *Eventi*.

Esempi**Decodifica immagine**

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_FFMPEG && LV_FFMPEG_PLAYER_USE_LV_FS

/***
 * Open an image from a file
 */
void lv_example_ffmpeg_1(void)
{
    /*It always uses the LVGL filesystem abstraction (not the OS filesystem)
     *to open the image, unlike `lv_ffmpeg_player_set_src` which depends on
     *the setting of `LV_FFMPEG_PLAYER_USE_LV_FS`.*/
    lv_obj_t * img = lv_image_create(lv_screen_active());
    lv_image_set_src(img, "A:lvgl/examples/libs/ffmpeg/ffmpeg.png");
    lv_obj_center(img);
}

#else

void lv_example_ffmpeg_1(void)
{
    /*TODO
     *fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Ffmpeg is not installed");
    lv_obj_center(label);
}

#endif
#endif
```

Decodifica video

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_FFMPEG

#if LV_FFMPEG_PLAYER_USE_LV_FS
#define PATH_PREFIX "A:"
#else
#define PATH_PREFIX "./"
#endif

/**
 * Open a video from a file
 */
void lv_example_ffmpeg_2(void)
{
    /*birds.mp4 is downloaded from http://www.videezy.com (Free Stock Footage by Videezy!)
 *https://www.videezy.com/abstract/44864-silhouettes-of-birds-over-the-sunset*/
/*It will use the LVGL filesystem abstraction (not the OS filesystem)
 *if 'LV_FFMPEG_PLAYER_USE_LV_FS' is set.*/
lv_obj_t * player = lv_ffmpeg_player_create(lv_screen_active());
lv_ffmpeg_player_set_src(player, PATH_PREFIX "lvgl/examples/libs/ffmpeg/birds.mp4");
lv_ffmpeg_player_set_auto_restart(player, true);
lv_ffmpeg_player_set_cmd(player, LV_FFMPEG_PLAYER_CMD_START);
lv_obj_center(player);
}

#else

void lv_example_ffmpeg_2(void)
{
    /*TODO
     *fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "FFmpeg is not installed");
    lv_obj_center(label);
}

#endif
#endif
```

API

lv_ffmpeg.h

lv_ffmpeg_private.h

10.4.2 GStreamer

GStreamer è un framework multimediale basato su pipeline che collega un'ampia gamma di sistemi di elaborazione multimediale per completare flussi di lavoro complessi.

Il framework utilizza un modello di pipeline in cui i dati multimediali fluiscono dagli elementi sorgente attraverso vari elementi di elaborazione (decodificatori, convertitori, filtri) fino agli elementi sink (display, altoparlanti, file).

L'estensione GStreamer in LVGL offre funzionalità di riproduzione video e audio con supporto per vari formati multimediali, protocolli di streaming e sorgenti multimediali. Sfrutta la potente architettura di pipeline di GStreamer per gestire la decodifica, l'elaborazione e il rendering dei contenuti multimediali.

Per informazioni dettagliate su GStreamer, consultare: <https://gstreamer.freedesktop.org/>

Funzionalità

L'implementazione di GStreamer di LVGL offre funzionalità complete di riproduzione multimediale:

Supporto per Sorgenti Multimediali:

- File locali tramite URI `file://`
- Streaming di rete con supporto HTTP/HTTPS
- Streaming RTSP per feed video in diretta
- Streaming UDP per applicazioni a bassa latenza
- Streaming multicast per una distribuzione di rete efficiente
- Dispositivi videocamera Video4Linux2 (V4L2) su Linux
- Acquisizione audio da dispositivi ALSA e PulseAudio
- Sorgenti di test per lo sviluppo audio e video

Supporto per Schemi URI:

Utilizzando la factory URI (LV_GSTREAMER_FACTORY_URI_DECODE), è possibile specificare diversi schemi URI come sorgenti multimediali:

- **File locali:** file:///path/to/video.mp4
- **Stream web:** http://example.com/stream.webm, https://secure.example.com/video.mp4
- **Flussi RTSP:** rtsp://camera.local/stream
- **Flussi UDP:** udp://239.255.12.42:1234
- **Multicast:** multicast://239.255.12.42:1234
- **Telecamere V4L2:** v4l2:///dev/video0
- **Dispositivi audio:** alsa://hw:0,0, pulse://default

uridecodebin di GStreamer seleziona automaticamente l'elemento sorgente e il decoder appropriati in base allo schema URI e al formato multimediale.

Controllo della Riproduzione:

- Operazioni di riproduzione, pausa e arresto
- Ricerca precisa di posizioni specifiche
- Controllo del volume con intervallo 0-100%
- Controllo della velocità di riproduzione (rallentatore e avanzamento rapido)
- Query di posizione e durata in tempo reale
- Gestione dello stato (NULL, READY, PAUSED, PLAYING)

Supporto Formati Multimediali:

GStreamer supporta un'ampia varietà di formati multimediali tramite il suo sistema di plugin:

- **Video:** H.264, H.265/HEVC, VP8, VP9, AV1, MPEG-4, WebM e molti altri
- **Audio:** AAC, MP3, Ogg Vorbis, FLAC, Opus, PCM e altri
- **Contenitori:** MP4, WebM, AVI, MKV, MOV, FLV e altri

Requisiti

L'estensione GStreamer richiede **GStreamer 1.0** o versione successiva con i seguenti componenti:

- gstreamer-1.0**
Framework GStreamer principale
- gstreamer-video-1.0**
Utility di gestione ed elaborazione video
- gstreamer-app-1.0**
Utility di integrazione delle applicazioni

Dipendenze

Seguire la documentazione ufficiale di GStreamer per installare le sue librerie di sviluppo sul proprio sistema: <https://gstreamer.freedesktop.org/documentation/installing/index.html?gi-language=c>

Configurazione

1. Installazione delle Dipendenze

Installare le librerie di sviluppo GStreamer per la propria piattaforma come mostrato nella precedente sezione "Dipendenze".

2. Abilitare il Supporto GStreamer

Settare `LV_USE_GSTREAMER` a 1 in `lv_conf.h`.

3. Integrazione con CMake

Opzione 1: Link diretto con LVGL (Consigliato)

```
find_package(PkgConfig REQUIRED)

# Find GStreamer packages
pkg_check_modules(GSTREAMER REQUIRED gstreamer-1.0)
pkg_check_modules(GSTREAMER_VIDEO REQUIRED gstreamer-video-1.0)
pkg_check_modules(GSTREAMER_APP REQUIRED gstreamer-app-1.0)

# Link with LVGL
target_include_directories(lvgl PUBLIC
    ${GSTREAMER_INCLUDE_DIRS}
    ${GSTREAMER_VIDEO_INCLUDE_DIRS}
    ${GSTREAMER_APP_INCLUDE_DIRS})
target_link_libraries(lvgl PUBLIC
    ${GSTREAMER_LIBRARIES}
    ${GSTREAMER_VIDEO_LIBRARIES}
    ${GSTREAMER_APP_LIBRARIES})
```

4. Compilazione Manuale con pkg-config

È anche possibile compilare manualmente utilizzando `pkg-config` per interrogare i flag necessari:

```
# Get compilation flags
gcc $(pkg-config --cflags gstreamer-1.0 gstreamer-video-1.0 gstreamer-app-1.0) \
-o your_app your_app.c lvgl.a
```

5. Esempio di Configurazione di Base

```
int main(void)
{
    /* Initialize LVGL */
    lv_init();

    /* Setup display driver */
    lv_display_t *display = lv_display_create(800, 480);
    /* ... configure display driver ... */

    /* Create and run your GStreamer application */
    lv_example_gstplayer_1();

    while (1) {
        lv_timer_handler();
    }

    return 0;
}
```

Utilizzo

Creazione di Base del Player GStreamer

Ecco come creare un player GStreamer di base e caricare i contenuti multimediali:

```
/* Create a GStreamer object */
lv_obj_t *streamer = lv_gstplayer_create(lv_screen_active());

/* Set the media source using URI factory */
lv_result_t result = lv_gstplayer_set_src(streamer,
    LV_GSTREAMER_FACTORY_URI_DECODE,
    LV_GSTREAMER_PROPERTY_URI_DECODE,
    "https://example.com/video.webm");

if (result != LV_RESULT_OK) {
    LV_LOG_ERROR("Failed to set GStreamer source");
    return;
}

/* Start playback */
lv_gstplayer_play(streamer);
```

Configurazione della Sorgente Multimediale

Il widget GStreamer supporta diverse sorgenti multimediali tramite diverse factory:

URI Factory (Consigliata):

```
/* Load from web URL */
lv_gstplayer_set_src(streamer, LV_GSTREAMER_FACTORY_URI_DECODE,
    LV_GSTREAMER_PROPERTY_URI_DECODE,
    "https://example.com/stream.webm");

/* Load from local file */
lv_gstplayer_set_src(streamer, LV_GSTREAMER_FACTORY_URI_DECODE,
```

(continues on next page)

(continua dalla pagina precedente)

```
/* RTSP stream */
lv_gstreamer_set_src(streamer, LV_GSTREAMER_FACTORY_URI_DECODE,
                     "file:///path/to/video.mp4");

/* RTSP stream */
lv_gstreamer_set_src(streamer, LV_GSTREAMER_FACTORY_URI_DECODE,
                     "LV_GSTREAMER_PROPERTY_URI_DECODE",
                     "rtsp://camera.local/stream");
```

File Factory:

```
/* Direct file access */
lv_gstreamer_set_src(streamer, LV_GSTREAMER_FACTORY_FILE,
                     LV_GSTREAMER_PROPERTY_FILE,
                     "/path/to/video.mp4");
```

Controllo della Riproduzione

La riproduzione multimediale si controlla con queste funzioni:

```
/* Basic playback control */
lv_gstreamer_play(streamer);
lv_gstreamer_pause(streamer);
lv_gstreamer_stop(streamer);

/* Get current state */
lv_gstreamer_state_t state = lv_gstreamer_get_state(streamer);

/* Seek to position (in milliseconds) */
lv_gstreamer_set_position(streamer, 30000); /* Seek to 30 seconds */

/* Get current position and duration */
uint32_t position = lv_gstreamer_get_position(streamer);
uint32_t duration = lv_gstreamer_get_duration(streamer);

/* Set playback rate - values relative to 256 (1x speed) */
lv_gstreamer_set_rate(streamer, 128); /* 0.5x speed */
lv_gstreamer_set_rate(streamer, 256); /* 1.0x speed (normal) */
lv_gstreamer_set_rate(streamer, 512); /* 2.0x speed */
```

Controllo del Volume

Il volume audio si gestisce con i controlli integrati:

```
/* Set volume (0-100% */
lv_gstreamer_set_volume(streamer, 75);

/* Get current volume */
uint8_t volume = lv_gstreamer_get_volume(streamer);
```

Gestione degli Eventi

Gli eventi GStreamer si gestiscono utilizzando il sistema di eventi di LVGL:

```
static void gstreamer_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * streamer = lv_event_get_target_obj(e);

    if(code == LV_EVENT_READY) {
        LV_LOG_USER("Stream ready - Duration: %" LV_PRIu32 " ms",
                   lv_gstreamer_get_duration(streamer));
        LV_LOG_USER("Resolution: %" LV_PRId32 "x%" LV_PRId32,
                   lv_image_get_src_width(streamer),
                   lv_image_get_src_height(streamer));
    }
}

/* Add event callback */
lv_obj_add_event_cb(streamer, gstreamer_event_cb, LV_EVENT_ALL, NULL);
```

Architettura del Widget

Il widget GStreamer estende il widget `lv_image`, il che significa:

- Tutte le funzioni standard `lv_obj` funzionano con i widget GStreamer (posizionamento, dimensionamento, stile, eventi).
- Tutte le funzioni `lv_image` sono disponibili per operazioni relative alle immagini.
- I "frame" [fotogrammi] video vengono renderizzati come contenuto immagine che si aggiorna automaticamente durante la riproduzione.

Gestione degli Stati

Il widget GStreamer mantiene i seguenti stati:

- LV_GSTREAMER_STATE_NULL: Stato iniziale, nessun contenuto multimediale caricato.
- LV_GSTREAMER_STATE_READY: Contenuto multimediale caricato e pronto per la riproduzione.
- LV_GSTREAMER_STATE_PAUSED: Riproduzione in pausa.
- LV_GSTREAMER_STATE_PLAYING: In riproduzione.

Accesso alle Informazioni sui Media

Una volta caricato il contenuto multimediale (LV_EVENT_READY), è possibile accedere a:

- Risoluzione video tramite `lv_image_get_src_width()` e `lv_image_get_src_height()`
- Durata del contenuto multimediale tramite `lv_gstreamer_get_duration()`
- Posizione di riproduzione corrente tramite `lv_gstreamer_get_position()`
- Livello del volume corrente tramite `lv_gstreamer_get_volume()`
- Stato di riproduzione corrente tramite `lv_gstreamer_get_state()`

Esempi

Caricamento di un video da Internet utilizzando il widget gstreamer

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_GSTREAMER
typedef struct {
    lv_obj_t * streamer;
    lv_obj_t * pp_button;
    lv_obj_t * button_label;
    lv_obj_t * position_label;
    lv_obj_t * duration_label;
    lv_subject_t position_subject;
} event_data_t;

static void volume_setter_create(event_data_t * event_data);
static void control_bar_create(event_data_t * event_data);
static void update_duration_label(lv_obj_t * label, uint32_t duration);
static void volume_observer_cb(lv_observer_t * observer, lv_subject_t * subject);
static void update_position_slider(lv_timer_t * timer);
static void play_pause_pressed(lv_event_t * e);
static void streamer_ready(lv_event_t * e);

/***
 * Loads a video from the internet using the gstreamer widget
 */
void lv_example_gstreamer_1(void)
{
    static event_data_t event_data;

    event_data.streamer = lv_gstreamer_create(lv_screen_active());

    /* the gstreamer widget inherits the 'lv_image' widget,
     * meaning you can also provide it lv_image functions, like
     lv_image_set_scale(event_data.streamer, 100);
     lv_image_set_rotation(event_data.streamer, 100);
    */

    /* Set the current src of the streamer.
     * Using the 'URI' "factory", we can
     * specify various URI schemes as media sources including local files (file://),
     * web streams (http://, https://), RTSP streams (rtsp://), UDP streams (udp://),
     * and many others. GStreamer's uridecodebin automatically selects the appropriate
     * source element and decoder based on the URI scheme and media format. */
    lv_gstreamer_set_src(event_data.streamer, LV_GSTREAMER_FACTORY_URI_DECODE, LV_GSTREAMER_PROPERTY_URI_DECODE,
                        "https://gstreamer.freedesktop.org/data/media/sintel_trailer-480p.webm");

    lv_obj_center(event_data.streamer);

    /* The LV_EVENT_READY will fire when the stream is ready at that point you can query the stream
     * information like its resolution and duration. See 'streamer_ready' */
    lv_obj_add_event_cb(event_data.streamer, streamer_ready, LV_EVENT_READY, &event_data);

    /* Play the stream immediately */
    lv_gstreamer_play(event_data.streamer);

    /* Create a slider to modify the stream volume and a label to visualize the current value */
    volume_setter_create(&event_data);

    /* Create a slider to see the position in the stream with 2 text label on each side
     * One for the current position in the stream and the other for the total duration of the stream
     * Also add a pause/play button*/
    control_bar_create(&event_data);

    /* Create a timer that will update the slider position based on the stream position
     * (continues on next page)
```

(continua dalla pagina precedente)

```

 * Make it 3 times faster than the refresh rate for a smoother effect */
lv_timer_create(update_position_slider, LV_DEF_REFR_PERIOD, &event_data);
}

static void volume_setter_create(event_data_t * event_data)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_remove_style_all(cont);
    lv_obj_set_style_pad_all(cont, 8, 0);
    lv_obj_set_style_pad_gap(cont, 8, 0);
    lv_obj_set_style_radius(cont, 8, 0);
    lv_obj_set_style_bg_color(cont, lv_color_white(), 0);
    lv_obj_set_style_bg_opa(cont, LV_OPA_70, 0);
    lv_obj_set_size(cont, 40, lv_pct(60));
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(cont, LV_FLEX_ALIGN_SPACE_BETWEEN, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);
    lv_obj_align(cont, LV_ALIGN_RIGHT_MID, -8, -40);

    lv_obj_t * volume_slider = lv_slider_create(cont);
    lv_obj_set_size(volume_slider, 13, LV_PCT(50));
    lv_obj_set_flex_grow(volume_slider, 1);

    lv_obj_t * volume_label = lv_label_create(cont);
    lv_obj_set_style_text_align(volume_label, LV_TEXT_ALIGN_CENTER, 0);
    lv_obj_set_width(volume_label, 50);

    /* We use `lv_subject` to simplify binding the data between multiple objects.
     * Here the data is shared between the slider, the label and the gstreamer widgets */
    static lv_subject_t volume_subject;
    lv_subject_init_int(&volume_subject, 50);
    lv_subject_add_observer_obj(&volume_subject, volume_observer_cb, event_data->streamer, NULL);
    lv_slider_bind_value(volume_slider, &volume_subject);
    lv_label_bind_text(volume_label, &volume_subject, LV_SYMBOL_VOLUME_MID "\n%3* LV_PRIId32 %%");

}

static void control_bar_create(event_data_t * event_data)
{
    lv_subject_init_int(&event_data->position_subject, 0);

    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_remove_style_all(cont);
    lv_obj_set_style_pad_all(cont, 8, 0);
    lv_obj_set_style_margin_hor(cont, 8, 0);
    lv_obj_set_style_pad_gap(cont, 8, 0);
    lv_obj_set_style_radius(cont, 8, 0);
    lv_obj_set_style_bg_color(cont, lv_color_white(), 0);
    lv_obj_set_style_bg_opa(cont, LV_OPA_70, 0);

    lv_obj_set_size(cont, lv_pct(100), LV_SIZE_CONTENT);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW);
    lv_obj_set_flex_align(cont, LV_FLEX_ALIGN_SPACE_BETWEEN, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);
    lv_obj_align(cont, LV_ALIGN_BOTTOM_MID, 0, -10);

    event_data->position_label = lv_label_create(cont);
    lv_obj_set_width(event_data->position_label, 80);
    lv_label_set_text_static(event_data->position_label, "0:00:000");

    /* Create a button a button to play/pause the stream */
    event_data->pp_button = lv_button_create(cont);
    lv_obj_center(event_data->pp_button);
    lv_obj_add_event_cb(event_data->pp_button, play_pause_pressed, LV_EVENT_CLICKED, event_data);
    event_data->button_label = lv_label_create(event_data->pp_button);
    lv_label_set_text_static(event_data->button_label, LV_SYMBOL_PAUSE);

    lv_obj_t * position_slider = lv_bar_create(cont);
    lv_bar_set_range(position_slider, 0, 1000);
    lv_obj_set_flex_grow(position_slider, 1);
    lv_slider_bind_value(position_slider, &event_data->position_subject);

    event_data->duration_label = lv_label_create(cont);
    lv_obj_set_width(event_data->duration_label, 80);
    lv_label_set_text_static(event_data->duration_label, "0:00:000");
    lv_obj_set_style_text_align(event_data->duration_label, LV_TEXT_ALIGN_RIGHT, 0);

}

static void update_duration_label(lv_obj_t * label, uint32_t duration)
{
    const uint32_t minutes = duration / 60000;
    const uint32_t seconds = (duration / 1000) % 60;
    const uint32_t milliseconds = duration % 1000;

    lv_label_set_text_fmt(label, "%s LV_PRIu32 :%02 LV_PRIu32 :%03 LV_PRIu32, minutes, seconds, milliseconds);
}

static void volume_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    lv_obj_t * streamer = lv_observer_get_target_obj(observer);
    int32_t volume = lv_subject_get_int(subject);
    LV_LOG_USER("Setting volume %" PRIId32 ", volume");
    lv_gstreamer_set_volume(streamer, (uint8_t)volume);
}

static void update_position_slider(lv_timer_t * timer)
{
    event_data_t * event_data = (event_data_t *)lv_timer_get_user_data(timer);

    uint32_t duration = lv_gstreamer_get_duration(event_data->streamer);
    uint32_t position = lv_gstreamer_get_position(event_data->streamer);
    int32_t position_perc = lv_map(position, 0, duration, 0, 1000);
    lv_subject_set_int(&event_data->position_subject, position_perc);
    update_duration_label(event_data->position_label, position);
}

static void play_pause_pressed(lv_event_t * e)
{
    event_data_t * event_data = (event_data_t *)lv_event_get_user_data(e);

    if(lv_streq(lv_label_get_text(event_data->button_label), LV_SYMBOL_PLAY)) {

```

(continues on next page)

(continua dalla pagina precedente)

```

    lv_label_set_text(event_data->button_label, LV_SYMBOL_PAUSE);
    lv_gstreamer_play(event_data->streamer);
}
else {
    lv_label_set_text(event_data->button_label, LV_SYMBOL_PLAY);
    lv_gstreamer_pause(event_data->streamer);
}
}

static void streamer_ready(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    event_data_t * event_data = (event_data_t *)lv_event_get_user_data(e);
    lv_obj_t * btn = event_data->pp.button;
    lv_obj_t * streamer = event_data->streamer;

    if(code == LV_EVENT_READY) {
        lv_obj_align(btn, LV_ALIGN_BOTTOM_MID, 0, 0);
        uint32_t duration = lv_gstreamer_get_duration(streamer);
        LV_LOG_USER("Video is starting");
        LV_LOG_USER("Stream resolution %" LV_PRId32 "x%" LV_PRId32, lv_image_get_src_width(streamer),
                    lv_image_get_src_height(streamer));
        LV_LOG_USER("\tStream duration %" LV_PRIu32, duration);
        update_duration_label(event_data->duration_label, duration);
    }
}

#endif
#endif

```

API

lv_gstreamer.h

lv_gstreamer_internal.h

10.5 Barcode

L'utilità LVGL Barcode consente di generare codici a barre Code-128. Utilizza la libreria [code128 library by fhunleth](#).

10.5.1 Utilizzo

Impostare `LV_USE_BARCODE` a 1 in `lv_conf.h`.

Utilizzare `lv_barcode_create()` per creare un oggetto barcode e `lv_barcode_update()` per generare un codice a barre.

Chiamare `lv_barcode_set_scale()` per regolare la scala, chiamare `lv_barcode_set_dark_color()` e `lv_barcode_set_light_color()` per regolare i colori. Si chiama `lv_barcode_set_direction()` per impostare l'orientamento del codice a barre.

Per default, viene utilizzata la codifica `LV_BARCODE_ENCODING_CODE128_GS1`, che elimina [FCN1] e gli spazi. Facoltativamente, si usa `lv_barcode_set_encoding()` per impostare `LV_BARCODE_ENCODING_CODE128_RAW`.

Dopo aver modificato una qualsiasi delle impostazioni di un codice a barre, richiamare nuovamente `lv_barcode_update()` per rigenerarlo.

10.5.2 Note

- È consigliabile non impostare manualmente la larghezza del codice a barre, perché, quando la larghezza del Widget è inferiore a quella del codice a barre, la visualizzazione risulterà incompleta a causa del troncamento.
- La regolazione della scala può essere solo un multiplo intero, ad esempio, `lv_barcode_set_scale(barcode, 2)` significa una scala 2x.
- L'impostazione della direzione può essere `LV_DIR_HOR` o `LV_DIR_VER`.

10.5.3 Esempio

Creare un Codice a Barre

```
#include "../../lv_examples.h"
#if LV_USE_BARCODE && LV_BUILD_EXAMPLES

/***
 * Create a Barcode
 */
void lv_example_barcode_1(void)
{
    lv_color_t bg_color = lv_palette_lighten(LV_PALETTE_LIGHT_BLUE, 5);
    lv_color_t fg_color = lv_palette_darken(LV_PALETTE_BLUE, 4);

    lv_obj_t * barcode = lv_barcode_create(lv_screen_active());
    lv_obj_set_height(barcode, 50);
    lv_obj_center(barcode);

    /*Set color*/
    lv_barcode_set_dark_color(barcode, fg_color);
    lv_barcode_set_light_color(barcode, bg_color);

    /*Add a border with bg_color*/
    lv_obj_set_style_border_color(barcode, bg_color, 0);

    /*Set data*/
    lv_barcode_update(barcode, "https://lvgl.io");
}

#endif
```

10.5.4 API

lv_barcode.h

lv_barcode_private.h

10.6 glTF

gltf (GL Transmission Format) è una specifica royalty-free per la trasmissione e il caricamento efficienti di scene e modelli 3D.

L'estensione glTF in LVGL offre funzionalità di caricamento, rendering e animazione di modelli 3D. Supporta la specifica glTF 2.0 completa con funzionalità di rendering avanzate.

Per un'introduzione dettagliata a glTF, vedere: <https://www.khronos.org/gltf/>

10.6.1 Funzionalità

L'implementazione glTF di LVGL offre funzionalità complete di rendering 3D:

Supporto Formati File:

- Caricamento di file glTF (.gltf) e GLB binari (.glb) dal file system locale
- Supporto per byte codificati in esadecimale nei file "include" quando il file system non è disponibile
- Supporto texture compresse JPG, PNG e WebP
- File texture esterni o texture incorporate nel sorgente glTF

Rendering e Illuminazione:

- "Image-Based Lighting (IBL)" [Illuminazione basata su immagini] per un'illuminazione ambientale realistica
- Supporto illuminazione puntuale con sorgenti luminose animate
- Materiali "Physically Based Rendering (PBR)" [rendering basati sulla fisica] con supporto completo delle texture:
 - Texture diffuse/albedo
 - Texture per workflow di rugosità e metalliche
 - Mappe delle normali per i dettagli delle superfici
 - Texture per occlusione ambientale

Materiali Avanzati:

- Materiali emissivi per effetti luminosi e superfici autoilluminate
- Materiali rifrangenti con effetti di distorsione realistici che rivelano la geometria dietro le superfici
- Supporto per materiali trasparenti per effetti di superficie multistrato (come finiture di vernice per auto)
- Supporto completo della trasparenza con alfa blending

Sistema di Animazione:

- Animazioni "keyframe-based" per le trasformazioni degli oggetti
- Animazioni di personaggi con un "Single-skeleton skinned" [scheletro singolo rivestito]
- Luci puntuali animate con effetti di illuminazione dinamici
- Binding di parametri per il monitoraggio e il controllo in tempo reale delle proprietà degli oggetti (posizione, rotazione, scala)
- Override dinamico dei parametri dall'applicazione host

Telecamera & Viewport:

- Controllo programmatico del viewport con manipolazione completa della telecamera
- Supporto per telecamere definite all'interno del software di editing 3D
- Comutazione di più telecamere per diverse prospettive di scena
- Modalità di proiezione ortografica e prospettica

Qualità del Rendering:

- Antialiasing configurabile con più modalità:
 - Sempre attivo per una qualità costante
 - Sempre disattivato per le massime prestazioni
 - Accensione automatica che si attiva all'arresto del movimento della scena
- Rendering del background flessibile:
 - Background basati sull'ambiente utilizzando IBL
 - Background a tinta unita
 - Background trasparenti per effetti di sovrapposizione

10.6.2 Cos'è glTF?

glTF (Graphics Library Transmission Format) è un formato di file 3D che memorizza le informazioni del modello 3D in formato JSON. Supporta:

- **Geometria:** Mesh, primitive e dati sui vertici
- **Materiali:** Materiali PBR (Physically Based Rendering) con texture
- **Animazioni:** Animazioni basate su keyframe per i nodi
- **Telecamere:** Definizioni di telecamere prospettiche e ortografiche
- **Scene:** Grafi gerarchici di scene con trasformazioni dei nodi

I file glTF possono essere memorizzati come:

- **.gltf:** Formato JSON con file binari e di immagine esterni
- **.glb:** Formato binario con tutte le risorse incorporate in un singolo file

10.6.3 Funzionalità

L'implementazione glTF di LVGL offre funzionalità complete di rendering 3D:

Supporto Formati File:

- Caricamento di file glTF (.gltf) e GLB binari (.glb) dal file system locale
- Supporto per byte codificati in esadecimale nei file "include" quando il file system non è disponibile
- Supporto texture compresse JPG, PNG e WebP
- File texture esterni o texture incorporate nel sorgente glTF

Rendering e Illuminazione:

- "Image-Based Lighting (IBL)" [Illuminazione basata su immagini] per un'illuminazione ambientale realistica
- Supporto illuminazione puntuale con sorgenti luminose animate
- Materiali "Physically Based Rendering (PBR)" [rendering basati sulla fisica] con supporto completo delle texture:
 - Texture diffuse/albedo
 - Texture per workflow di rugosità e metalliche
 - Mappe delle normali per i dettagli delle superfici
 - Texture per occlusione ambientale

Materiali Avanzati:

- Materiali emissivi per effetti luminosi e superfici autoilluminate
- Materiali rifrangenti con effetti di distorsione realistici che rivelano la geometria dietro le superfici
- Supporto per materiali trasparenti per effetti di superficie multistrato (come finiture di vernice per auto)
- Supporto completo della trasparenza con alfa blending

Sistema di Animazione:

- Animazioni "keyframe-based" per le trasformazioni degli oggetti
- Animazioni di personaggi con a "Single-skeleton skinned" [scheletro singolo rivestito]
- Luci puntuali animate con effetti di illuminazione dinamici
- Binding di parametri per il monitoraggio e il controllo in tempo reale delle proprietà degli oggetti (posizione, rotazione, scala)
- Override dinamico dei parametri dall'applicazione host

Telecamera & Viewport:

- Controllo programmatico del viewport con manipolazione completa della telecamera
- Supporto per telecamere definite all'interno del software di editing 3D
- Comutazione di più telecamere per diverse prospettive di scena
- Modalità di proiezione ortografica e prospettica

Qualità del Rendering:

- Antialiasing configurabile con più modalità:
 - Sempre attivo per una qualità costante
 - Sempre disattivato per le massime prestazioni
 - Accensione automatica che si attiva all'arresto del movimento della scena
- Rendering del background flessibile:

- Background basati sull'ambiente utilizzando IBL
- Background a tinta unita
- Background trasparenti per effetti di sovrapposizione

10.6.4 Requisiti

L'estensione glTF si basa su **OpenGL ES 3.0** per il rendering 3D. LVGL include il supporto integrato per OpenGL ES tramite il driver GLFW, che fornisce la gestione multipiattaforma delle finestre e la creazione di contesti OpenGL.

Il renderer utilizza shader OpenGL ES 3.0 (GLSL versione 300 es) per fornire moderne funzionalità PBR (Physically Based Rendering).

Dipendenze

L'estensione glTF richiede le seguenti librerie esterne:

fastgltf

Una libreria C++20 per il parsing dei file glTF (<https://github.com/spnda/fastgltf>)

libwebp

Supporto del formato immagine WebP per le texture (<https://github.com/webmproject/libwebp>)

10.6.5 Configurazione

1. Installare le Dipendenze con CMake

Il metodo consigliato per integrare le librerie richieste è utilizzare FetchContent di CMake:

```
include(FetchContent)
# Set policy to allow to run the target_link_libraries cmd on targets that are
# build in another directory. Currently, the linking is not handled by
cmake_policy(SET CMP0079 NEW)
# Fetch fastgltf library
FetchContent_Declare(
    fastgltf
    GIT_REPOSITORY https://github.com/spnda/fastgltf
    GIT_TAG 4e226135088bae7c35a1f39991f6233d57795f5)
set(FASTGLTF_ENABLE_DEPRECATED_EXT
    ON
    CACHE BOOL "" FORCE)
set(FASTGLTF_DIFFUZE_TRANSMISSION_SUPPORT
    ON
    CACHE BOOL "" FORCE)
FetchContent_MakeAvailable(fastgltf)

# Fetch libwebp library
FetchContent_Declare(
    webp
    GIT_REPOSITORY https://github.com/webmproject/libwebp
    GIT_TAG fa6f56496a442ed59b103250021e4b14ebf1427)
FetchContent_MakeAvailable(webp)

# Link libraries to LVGL
target_link_libraries(lvgl PUBLIC webp fastgltf)
```

2. Abilitare il Supporto glTF

Impostare `LV_USE_GLTF` a 1 in `lv_conf.h`.

Abilitare anche altre dipendenze richieste impostando le seguenti definizioni a 1:

- `LV_USE_OPENGL`
- `LV_USE_DRAW_OPENGL`
- `LV_USE_3DTEXTURE`

3. Configurazione del Driver OpenGL ES

Seguire la documentazione di configurazione del driver OpenGL ES (*Driver OpenGL*) per configurare il supporto GLFW e OpenGL ES per la propria piattaforma.

4. Esempio di Configurazione di Base

```

int main(void)
{
    /* Initialize LVGL */
    lv_init();

    /* GLFW setup */
    lv_opengles_window_t *window = lv_opengles_glfw_window_create(WINDOW_WIDTH, WINDOW_HEIGHT, true);
    lv_display_t *display = lv_opengles_texture_create(WINDOW_WIDTH, WINDOW_HEIGHT);
    unsigned int texture_id = lv_opengles_texture_get_texture_id(display);
    lv_opengles_window_add_texture(window, texture_id, WINDOW_WIDTH, WINDOW_HEIGHT);

    /* Load and display glTF demo */
    lv_demo_gltf("A:<path/to/gltf>");

    while (1) {
        uint32_t time_until_next = lv_timer_handler();
        if (time_until_next == LV_NO_TIMER_READY) {
            time_until_next = LV_DEF_REFR_PERIOD;
        }
        lv_delay_ms(time_until_next);
    }
    return 0;
}

```

10.6.6 Utilizzo

Applicazione Demo

LVGL fornisce una demo glTF completa che mostra le funzionalità del visualizzatore 3D:

```
lv_demo_gltf("A:<path/to/your/model.glb>");
```

Questa demo crea un visualizzatore 3D interattivo con:

- Controlli mouse/touch per la rotazione della telecamera
- Controlli zoom
- Controlli di riproduzione dell'animazione
- Cambio telecamera
- Regolazione delle impostazioni visive

Creazione di Base del Visualizzatore glTF

Ecco come creare un visualizzatore glTF di base e caricare un modello:

```

/* Create a glTF viewer object */
lv_obj_t *gltf = lv_gltf_create(lv_screen_active());

/* Load a glTF model from file */
lv_gltf_model_t *model = lv_gltf_load_model_from_file(gltf,
    "A:path/to/your/model.glb");

if (model == NULL) {
    LV_LOG_ERROR("Failed to load glTF model");
    return;
}

```

Controlli Telecamera

Il visualizzatore glTF offre controlli telecamera completi:

```

/* Set camera rotation */
lv_gltf_set_yaw(gltf, 45.0f);      /* Horizontal rotation */
lv_gltf_set_pitch(gltf, -30.0f);   /* Vertical rotation */

/* Set camera distance */
lv_gltf_set_distance(gltf, 5.0f);

/* Set field of view (0 for orthographic) */
lv_gltf_set_fov(gltf, 60.0f);

/* Set the focal point - The position the camera is pointing at */
lv_gltf_set_focal_x(gltf, 0.0f);
lv_gltf_set_focal_y(gltf, 0.0f);
lv_gltf_set_focal_z(gltf, 0.0f);

/* Recenter camera on model */
lv_gltf_recenter(gltf, model);

```

Controllo Animazione

Controllare le animazioni del modello con queste funzioni:

```
/* Get number of animations */
size_t anim_count = lv_gltf_model_get_animation_count(model);

/* Play a specific animation */
lv_gltf_model_play_animation(model, 0);

/* Control animation speed */
lv_gltf_set_animation_speed(gltf, LV_GLTF_ANIM_SPEED_2X);

/* Pause/resume animation */
lv_gltf_model_pause_animation(model);
bool is_paused = lv_gltf_model_is_animation_paused(model);
```

Impostazioni Visive

Personalizzare l'aspetto visivo della scena 3D:

```
/* Background settings */
lv_gltf_set_background_mode(gltf, LV_GLTF_BG_ENVIRONMENT);
lv_gltf_set_background_blur(gltf, 50);

/* Lighting and exposure */
lv_gltf_set_env_brightness(gltf, 150);
lv_gltf_set_image_exposure(gltf, 1.2f);

/* Anti-aliasing */
lv_gltf_set_antialiasing_mode(gltf, LV_GLTF_AA_DYNAMIC);
```

Supporto Multi-Modello

Caricare e gestire più modelli glTF in un unico visualizzatore:

```
/* Load multiple models */
lv_gltf_model_t * model1 = lv_gltf_load_model_from_file(gltf, "A:modell.glb");
lv_gltf_model_t * model2 = lv_gltf_load_model_from_file(gltf, "A:modell2.glb");

/* Get model information */
size_t model_count = lv_gltf_get_model_count(gltf);
lv_gltf_model_t * primary = lv_gltf_get_primary_model(gltf);
lv_gltf_model_t * specific = lv_gltf_get_model_by_index(gltf, 1);
```

Ispezione del Modello

Interrogare le proprietà del modello per comprenderne la struttura:

```
/* Get model component counts */
size_t mesh_count = lv_gltf_model_get_mesh_count(model);
size_t material_count = lv_gltf_model_get_material_count(model);
size_t texture_count = lv_gltf_model_get_texture_count(model);
size_t image_count = lv_gltf_model_get_image_count(model);
size_t node_count = lv_gltf_model_get_node_count(model);
size_t scene_count = lv_gltf_model_get_scene_count(model);
size_t camera_count = lv_gltf_model_get_camera_count(model);
```

10.6.7 Architettura del Widget

Il widget glTF estende il widget `lv_3dtexture`, il che significa:

- Tutte le funzioni standard `lv_obj` funzionano con i widget glTF (posizionamento, dimensionamento, stile, ecc.)
- Tutte le funzioni `lv_3dtexture` sono disponibili anche per la gestione avanzata delle texture 3D

Sistema di Velocità delle Animazioni

Le velocità delle animazioni utilizzano valori interi per evitare calcoli aritmetici in virgola mobile:

- Il rapporto di velocità è calcolato come: `speed_value / LV_GLTF_ANIM_SPEED_NORMAL`
- `LV_GLTF_ANIM_SPEED_NORMAL` è uguale a 1000, che rappresenta una velocità 1,0x (normale)
- Valori superiori a 1000 accelerano le animazioni (ad esempio, 2000 = 2,0x speed)
- Valori inferiori a 1000 rallentano le animazioni (ad esempio, 500 = 0,5x speed)

10.6.8 Esempi

Aprire un GLTF da un file e farlo girare all'infinito come un "disco"

```
#include "lv_example_gltf.h"

#if LV_BUILD_EXAMPLES

#if LV_USE_GLTFS
/***
 * Timer callback to continuously rotate the model
 */
static void spin_timer_cb(lv_timer_t * timer)
{
    lv_obj_t * gltf = lv_timer_get_user_data(timer);

    float yaw = lv_gltf_get_yaw(gltf);
    yaw += 1.0f;
    if(yaw >= 360.0f) {
        yaw = 0.0f;
    }
    lv_gltf_set_yaw(gltf, yaw);
}

/***
 * Open a GLTF from a file and make it spin forever like a platter
 */
void lv_example_gltf_1(void)
{
    lv_obj_t * gltf = lv_gltf_create(lv_screen_active());
    lv_gltf_model_t * model = lv_gltf_load_model_from_file(gltf,
        "A:lvgl/examples/libs/gltf/webp_diffuse_transmission_plant.gltf");
    lv_obj_set_size(gltf, LV_PCT(100), LV_PCT(100));

    lv_gltf_model_play_animation(model, 0);
    lv_gltf_set_pitch(gltf, -45.f);
    lv_timer_create(spin_timer_cb, LV_DEF_REFR_PERIOD, gltf);
}

#else
void lv_example_gltf_1(void)
{
    /*TODO
     *fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "gltf web support is coming soon");
    lv_obj_center(label);
}
#endif
#endif
```

Aprire un GLTF da un file e guardare attraverso ogni telecamera

```
#include "lv_example_gltf.h"

#if LV_BUILD_EXAMPLES

#if LV_USE_GLTFS
static size_t camera_count;
static size_t current_camera;
static uint32_t current_speed = LV_GLTFS_ANIM_SPEED_HALF;

/***
 * Timer callback that updates the current camera and the current animation speed
 */
static void timer_cb(lv_timer_t * timer)
{
    lv_obj_t * gltf = (lv_obj_t *) lv_timer_get_user_data(timer);
    current_camera = (current_camera + 1) % (camera_count + 1);
    current_speed *= 2;

    if(current_speed > LV_GLTFS_ANIM_SPEED_4X) {
        current_speed = LV_GLTFS_ANIM_SPEED_HALF;
    }

    LV_LOG_USER("Setting camera %zu and animation speed %" PRIu32, current_camera, current_speed);
    lv_gltf_set_camera(gltf, current_camera);
    lv_gltf_set_animation_speed(gltf, current_speed);
}

/***
 * Open a GLTF from a file and loop through the model cameras and multiple animation speeds
 */
void lv_example_gltf_2(void)
{
    lv_obj_t * gltf = lv_gltf_create(lv_screen_active());
    lv_gltf_model_t * model = lv_gltf_load_model_from_file(gltf,
        "A:lvgl/examples/libs/gltf/webp_diffuse_transmission_plant.gltf");

    lv_gltf_set_animation_speed(gltf, current_speed);
    lv_gltf_model_play_animation(model, 0);
    lv_obj_set_size(gltf, LV_PCT(100), LV_PCT(100));
    camera_count = lv_gltf_get_camera_count(gltf);
    lv_gltf_set_pitch(gltf, -45.f);
    lv_timer_create(timer_cb, 5000, gltf);
}

#else
void lv_example_gltf_2(void)
{
    /*TODO
     *fallback for online examples*/
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * label = lv_label_create(lv_screen_active());
lv_label_set_text(label, "gltf web support is coming soon");
lv_obj_center(label);
}

#endif
#endif

```

10.6.9 API

lv_gltf_bind.h
lv_gltf_data_internal.h
lv_gltf_model.h
lv_gltf_view_shader.h
lv_gltf_ibl_sampler.h
lv_gltf.h
lv_gltf_view_internal.h
lv_gltf_math.hpp

10.7 QR Code

La libreria QR-Code-generator di nayuki è una libreria di terze parti che genera bitmap di codici QR.

L'estensione LVGL *lv_qrcode* è un'interfaccia a tale libreria che implementa un widget personalizzato che genera e visualizza codici QR utilizzando la libreria.

10.7.1 Utilizzo

Abilitare *LV_USE_QRCODE* in *lv_conf.h* impostandone il valore a 1.

Usare *lv_qrcode_create()* per creare il Widget QR-Code. Quindi utilizzare *lv_qrcode_update()* per generare il codice QR.

Se è necessario modificare nuovamente le dimensioni e il colore, utilizzare rispettivamente *lv_qrcode_set_size()* e *lv_qrcode_set_dark_color()* o *lv_qrcode_set_light_color()*, quindi richiamare *lv_qrcode_update()* per aggiornare il codice QR.

10.7.2 Note

- I codici QR con meno dati sono più piccoli, ma vengono ridimensionati di un valore intero per adattarsi al meglio alle dimensioni specificate.

10.7.3 Esempio

Creare un codice QR

```

#include "../../lv_examples.h"
#if LV_USE_QRCODE && LV_BUILD_EXAMPLES
#include <string.h>

/***
 * Create a QR Code
 */
void lv_example_qrcode_1(void)
{
    lv_color_t bg_color = lv_palette_lighten(LV_PALETTE_LIGHT_BLUE, 5);
    lv_color_t fg_color = lv_palette_darker(LV_PALETTE_BLUE, 4);

    lv_obj_t * qr = lv_qrcode_create(lv_screen_active());
    lv_qrcode_set_size(qr, 150);
    lv_qrcode_set_dark_color(qr, fg_color);
    lv_qrcode_set_light_color(qr, bg_color);

    /*Set data*/
    const char * data = "https://lvgl.io";
    lv_qrcode_update(qr, data, strlen(data));
}

```

(continues on next page)

(continua dalla pagina precedente)

```
lv_obj_center(qr);
/*Add a border with bg_color*/
lv_obj_set_style_border_color(qr, bg_color, 0);
lv_obj_set_style_border_width(qr, 5, 0);
}

#endif
```

10.7.4 API

lv_qrcode.h

lv_qrcode_private.h

CAPITOLO 11

Debugging

LVGL include vari strumenti che possono aiutare a risolvere i problemi che si potrebbero incontrare durante lo sviluppo e il test.

11.1 Plug-In GDB

11.1.1 Debug di LVGL con GDB

Per facilitare il debug di LVGL con GDB, è disponibile un plugin GDB. Questo plugin si trova nella directory `lvgl/scripts/gdb`. Il plugin GDB può essere utilizzato con qualsiasi target in cui GDB sia disponibile. Ad esempio, è possibile utilizzarlo per eseguire il debug di un dispositivo connesso a un PC tramite JLink, che fornisce un server GDB. Inoltre, se il dispositivo si blocca e si verifica un core dump, è possibile utilizzare GDB per analizzarlo. Per caricare il plugin GDB LVGL dalla riga di comando di GDB, digitare il seguente comando:

```
source lvgl/scripts/gdb/gdbinit.py
```

Esempio di utilizzo:

```
(gdb) source lvgl/scripts/gdb/gdbinit.py
(gdb) dump obj -L 2
obj@0x607000000dd10 (0, 0, 799, 599)
tabview@0x6080000204ca0 (0, 0, 799, 599)
    obj@0x607000025da0 (0, 0, 799, 599)
    obj@0x607000025e80 (0, 78, 799, 599)
    obj@0x60700002bd70 (743, 543, 791, 591)
    btn@0x60700002c7f0 (747, 547, 787, 587)
keyboard@0x60d00000f7040 (0, 380, 799, 599)
dropdown-list@0x6080000205420 (0, 0, 129, 129)
    label@0x60d00000f7ba0 (22, 22, 56, 39)
(gdb)
```

Il plugin fornisce i seguenti comandi.

- `dump obj`: Esegue il dump dell'albero degli oggetti.
- `info style`: Mostra lo stile dell'oggetto.
- `info draw_unit`: Visualizza tutte le informazioni sull'unità di disegno corrente.

Nota

Alcune versioni di gdb su Windows (ad esempio quelle fornite con varie versioni di Perl) sono compilate senza il supporto Python, quindi il comando `source` non sarà supportato.

11.1.2 Dump dell'Obj Tree

`dump obj`: Esegue il dump dell'albero degli oggetti.

`dump obj -L 2`: Esegue il dump dell'albero degli oggetti con una profondità di 2.

`dump obj -a 0x60700000dd10`: Esegue il dump dell'albero degli oggetti a partire dall'indirizzo specificato.

11.1.3 Mostrare lo Stile di Obj

Questo comando può eseguire il dump dello stile locale dell'oggetto, poiché il valore dello stile è un'unione, e viene visualizzato in tutti i formati possibili.

`info style address_of_obj`: Mostra lo stile dell'oggetto.

Esempio:

```
(gdb) info style 0x60700000dd10
32 = {num = 90, ptr = 0x5a, color = {blue = 90 '\Z', green = 0 '\000', red = 0 '\000'}}
158 = {num = 32767, ptr = 0xffff, color = {blue = 255 '\377', green = 127 '\177', red = 0 '\000'}}
(gdb) p lv_global->disp_default->act_scr
$4 = ({lv_obj_t *) 0x60700000dd10
(gdb) info style $4
32 = {num = 90, ptr = 0x5a, color = {blue = 90 '\Z', green = 0 '\000', red = 0 '\000'}}
158 = {num = 32767, ptr = 0xffff, color = {blue = 255 '\377', green = 127 '\177', red = 0 '\000'}}
(gdb)
```

11.1.4 Connettersi al Debugger

Questo comando consente di connettersi ed eseguire il debug di script Python GDB tramite IDE.

Connessione a PyCharm / VSCode / Eclipse (non ancora supportato)

`debugger -t pycharm`

`debugger -t vscode`

`debugger -t eclipse`

Eseguire una ricerca sul web per `pydevd_pycharm` o `debugpy` per dettagli su come utilizzare il debugger.

11.2 Logging

LVGL ha un modulo *Logging* integrato per informare l'utente su ciò che accade nella libreria.

11.2.1 Configurazione del Logging

Livello di log

Per abilitare il logging, impostare `LV_USE_LOG` in `lv_conf.h` su un valore diverso da zero e impostare `LV_LOG_LEVEL` su uno dei seguenti valori. Sono ordinati come segue (dal più dettagliato al meno dettagliato):

- `LV_LOG_LEVEL_TRACE`: Molti log per fornire informazioni dettagliate
- `LV_LOG_LEVEL_INFO`: Log di eventi importanti.
- `LV_LOG_LEVEL_WARN`: Log se si è verificato un evento indesiderato che non ha causato problemi.
- `LV_LOG_LEVEL_ERROR`: Log dei soli problemi critici, in cui il sistema potrebbe non funzionare.
- `LV_LOG_LEVEL_USER`: Log dei soli messaggi di log personalizzati aggiunti dall'utente.
- `LV_LOG_LEVEL_NONE`: Non logga nulla.

Quando si impone `LV_LOG_LEVEL` su un determinato livello, verranno loggati solo i messaggi con quel livello o una priorità superiore (meno dettagliati).

Esempio: se si impone `LV_LOG_LEVEL` su `LV_LOG_LEVEL_WARN`, verranno registrati i messaggi `LV_LOG_LEVEL_WARN`, `LV_LOG_LEVEL_ERROR` e `LV_LOG_LEVEL_USER`.

Output del Log

Se il sistema supporta `printf`, è sufficiente abilitare `LV_LOG_PRINTF` in `lv_conf.h` per generare messaggi di log con `printf`.

Se non è possibile utilizzare `printf` o per utilizzare una funzione personalizzata per il log, si può registrare una funzione di "logging" con `lv_log_register_print_cb()`.

Ad esempio:

```
void my_log_cb(lv_log_level_t level, const char * buf)
{
    serial_send(buf, strlen(buf));
}

...

lv_log_register_print_cb(my_log_cb);
```

11.2.2 Utilizzo del Logging

Il modulo di log può essere utilizzato tramite le seguenti macro:

- `LV_LOG_TRACE(text)`
- `LV_LOG_INFO(text)`
- `LV_LOG_WARN(text)`
- `LV_LOG_ERROR(text)`
- `LV_LOG_USER(text)`
- `LV_LOG(text)`

Le prime 5 macro aggiungono le seguenti informazioni al `text`:

- Log Level name ("Trace", "Info", "Warn", "Error", "User")
- `__FILE__`
- `__LINE__`
- `__func__`

`LV_LOG(text)` è simile a `LV_LOG_USER` ma senza aggiungere informazioni aggiuntive.

11.2.3 API

`lv_log.h`

11.3 Monkey

Il modulo Monkey fornisce alle applicazioni LVGL un semplice test Monkey. Il Monkey Testing è una tecnica in cui l'utente testa l'applicazione o il sistema fornendo input casuali e verificandone il comportamento o verificando se l'applicazione o il sistema si arresteranno in modo anomalo. Questo modulo fornisce questo servizio come input casuale simulato per testare un'applicazione LVGL.

11.3.1 Utilizzo

Per prima cosa, si abilita `LV_USE_MONKEY` in `lv_conf.h`.

Poi, si dichiara una variabile (può essere locale) di tipo `lv_monkey_config_t` per definire la struttura di configurazione, inizializzarla usando `lv_monkey_config_init(cfg)`, quindi se ne imposta il membro `type` sul tipo desiderato di `input device` e si impostano i valori `min` e `max` per i suoi membri `period_range` e `input_range` per impostare gli intervalli di tempo (in millisecondi) e gli intervalli di input che il modulo Monkey utilizzerà per generare input casuali in momenti casuali.

Successivamente, si chiama `lv_monkey_create(cfg)` per creare il Monkey. Restituisce un puntatore al `lv_monkey_t` creato.

Infine, si chiama `lv_monkey_set_enable(monkey, true)` per abilitare Monkey.

Per mettere in pausa Monkey, si chiama `lv_monkey_set_enable(monkey, false)`. Per eliminarlo, si chiama `lv_monkey_delete(monkey)`.

Notare che `input_range` ha significati diversi a seconda del type del dispositivo di input:

- **`LV_INDEV_TYPE_POINTER`**: Nessun effetto, clic casuale entro i pixel della risoluzione dello schermo.
- **`LV_INDEV_TYPE_ENCODER`**: I valori minimo e massimo di `enc_diff`.
- **`LV_INDEV_TYPE_BUTTON`**: I valori minimo e massimo di `btn_id`. Si usa `lv_monkey_get_indev()` per ottenere il dispositivo di input e `lv_indev_set_button_points()` per mappare l'ID del tasto alle coordinate.
- **`LV_INDEV_TYPE_KEYPAD`**: Nessun effetto, invia *Tasti* casuali.

11.3.2 Esempio

Esempio di "touchpad monkey"

```
#include "../../lv_examples.h"
#if LV_USE_MONKEY && LV_BUILD_EXAMPLES

void lv_example_monkey_1(void)
{
    /*Create pointer monkey test*/
    lv_monkey_config_t config;
    lv_monkey_config_init(&config);
    config.type = LV_INDEV_TYPE_POINTER;
    config.period_range.min = 10;
    config.period_range.max = 100;
    lv_monkey_t * monkey = lv_monkey_create(&config);

    /*Start monkey test*/
    lv_monkey_set_enable(monkey, true);
}

#endif
```

Esempio di "encoder monkey"

```
#include "../../lv_examples.h"
#if LV_USE_MONKEY && LV_BUILD_EXAMPLES

void lv_example_monkey_2(void)
{
    /*Create encoder monkey test*/
    lv_monkey_config_t config;
    lv_monkey_config_init(&config);
    config.type = LV_INDEV_TYPE_ENCODER;
    config.period_range.min = 50;
    config.period_range.max = 500;
    config.input_range.min = -5;
    config.input_range.max = 5;
    lv_monkey_t * monkey = lv_monkey_create(&config);

    /*Set the default group*/
    lv_group_t * group = lv_group_create();
    lv_indev_set_group(lv_monkey_get_indev(monkey), group);
    lv_group_set_default(group);

    /*Start monkey test*/
    lv_monkey_set_enable(monkey, true);
}

#endif
```

Esempio di "button monkey"

```
#include "../../lv_examples.h"
#if LV_USE_MONKEY && LV_BUILD_EXAMPLES

void lv_example_monkey_3(void)
{
    static lv_point_t btn_points[3];
    int32_t hor_res = LV_HOR_RES;

    /*Create button monkey test*/
    lv_monkey_config_t config;
    lv_monkey_config_init(&config);
    config.type = LV_INDEV_TYPE_BUTTON;
    config.period_range.min = 50;
    config.period_range.max = 500;
    config.input_range.min = 0;
    config.input_range.max = sizeof(btn_points) / sizeof(lv_point_t) - 1;
    lv_monkey_t * monkey = lv_monkey_create(&config);

    /*Set the coordinates bound to the button*/
}
```

(continues on next page)

(continua dalla pagina precedente)

```

btn_points[0].x = hor_res / 4;
btn_points[0].y = 10;
btn_points[1].x = hor_res / 2;
btn_points[1].y = 10;
btn_points[2].x = hor_res * 3 / 4;
btn_points[2].y = 10;

lv_indev_set_button_points(lv_monkey_get_indev(monkey), btn_points);

/*Start monkey test*/
lv_monkey_set_enable(monkey, true);
}

#endif

```

11.3.3 API

lv_monkey.h

11.4 ID del Widget

I Widget possono facoltativamente aggiungere identificatori alle loro funzionalità, se necessario per l'applicazione. Il modo esatto in cui ciò avviene è progettato per essere flessibile e può adattarsi alle esigenze dell'applicazione. Può anche essere un timestamp o altri dati correnti al momento della creazione del Widget.

11.4.1 Utilizzo

La funzionalità ID Widget si abilita impostando *LV_USE_OBJ_ID* a 1 in *lv_conf.h*.

Una volta abilitata, diverse cose cambiano:

- ogni Widget avrà ora un campo *void ** chiamato *id*;
- diventano disponibili queste due funzioni API:
 - *lv_obj_get_id(widget)*,
 - *lv_obj_find_by_id(widget, id)*;
- diventano disponibili diverse altre funzioni API relative al Widget-ID se *LV_USE_OBJ_ID_BUILTIN* è diverso da zero (maggiori dettagli di seguito);
- ora sono disponibili due macro di configurazione aggiuntive, *LV_OBJ_ID_AUTO_ASSIGN* e *LV_USE_OBJ_ID_BUILTIN*.

LV_OBJ_ID_AUTO_ASSIGN

Questa macro in *lv_conf.h* assume per default il valore equivalente a *LV_USE_OBJ_ID*. È modificabile. In entrambi i casi, se equivale a un valore diverso da zero, si verificano due cose:

- *lv_obj_assign_id(class_p, widget)* verrà chiamato al termine della creazione di ogni Widget e
- *lv_obj_free_id(widget)* verrà chiamato al termine della sequenza quando ogni Widget viene eliminato.

lv_obj_assign_id(class_p, widget)

Questa funzione (fornita da LVGL o in proprio --- maggiori dettagli di seguito) è responsabile dell'assegnazione di un valore al campo *id* del Widget e, a seconda dell'implementazione, può svolgere anche altre funzioni.

lv_obj_free_id(widget)

Questa funzione (fornita da LVGL o in proprio --- maggiori dettagli di seguito) è responsabile della pulizia di tutte le risorse allocate da `lv_obj_assign_id()`

LV_USE_OBJ_ID_BUILTIN

Questa macro in `lv_conf.h` equivale a 1 per default. È modificabile. Quando equivale a un valore diverso da zero, LVGL fornisce le seguenti implementazioni di funzione:

- `lv_obj_assign_id(class_p, widget)`
- `lv_obj_free_id(widget)`
- `lv_obj_set_id(widget, id)`
- `lv_obj_stringify_id(widget, buf, len)`
- `lv_obj_id_compare(id1, id2)`

Queste forniscono l'implementazione di default per gli ID dei Widget, ovvero che per ogni Widget creato, `lv_obj_stringify_id(widget, buf, len)` produrrà una stringa univoca. Esempio: se i seguenti 6 widget vengono creati in questa sequenza:

- Screen
- Etichetta
- Pulsante
- Etichetta
- Etichetta
- Immagine

Le stringhe prodotte da `lv_obj_stringify_id(widget, buf, len)` sarebbero:

- obj1
- label1
- btn1
- label2
- label3
- image1

rispettivamente.

Utilizzo di un generatore di ID personalizzato

Si possono fornire implementazioni personalizzate per diverse funzioni relative ai Widget-ID. Per farlo, si imposta prima `LV_USE_OBJ_ID_BUILTIN` a 0 in `lv_conf.h`.

Sarà quindi necessario fornire le implementazioni per le seguenti funzioni (e linkarle a LVGL):

```
const char * lv_obj_stringify_id(lv_obj_t * widget, char * buf, uint32_t len);
int lv_obj_id_compare(const void * id1, const void * id2);
```

Se `LV_OBJ_ID_AUTO_ASSIGN` equivale a un valore diverso da zero (o se semplicemente c'è bisogno di utilizzarle), sarà necessario fornire anche le implementazioni per:

```
void lv_obj_assign_id(const lv_obj_class_t * class_p, lv_obj_t * widget);
void lv_obj_free_id(lv_obj_t * widget);
```

Se `LV_BUILD_TEST` equivale a un valore diverso da zero e si include codice di test LVGL nella compilazione (o se semplicemente c'è bisogno di utilizzarle), sarà necessario fornire anche un'implementazione per:

```
void lv_obj_set_id(lv_obj_t * widget, void * id);
```

Esempi di implementazioni di queste funzioni sono disponibili in `lv_obj_id_builtin.c`, ma è possibile utilizzare un design diverso se necessario.

`lv_obj_stringify_id()` converte il widget passato, in una rappresentazione stringa (in genere incorporando il campo `id`) e lo scrive nel buffer fornito nel suo argomento `buf`.

`lv_obj_id_compare()` confronta 2 valori `void * id` e restituisce 0 quando sono considerati uguali, altrimenti un valore diverso da zero.

Se `LV_OBJ_ID_AUTO_ASSIGN` equivale a un valore diverso da zero, `lv_obj_assign_id()` viene chiamato quando viene creato un Widget. È responsabile dell'assegnazione di un valore al campo `id` del Widget. Un puntatore alla classe finale del Widget viene passato nell'argomento `class_p` nel caso in cui sia necessario per determinare il valore del campo `id` o per altre possibili esigenze relative alla progettazione degli ID Widget. Si noti che questo puntatore potrebbe essere diverso da `widget->class_p`, che è la classe del Widget attualmente in fase di creazione.

Se `LV_OBJ_ID_AUTO_ASSIGN` equivale a un valore diverso da zero, `lv_obj_free_id()` viene chiamato quando un Widget viene eliminato. È necessario eseguire la pulizia di tutte le risorse allocate da `lv_obj_assign_id()`.

Dumping di un Widget Tree

Indipendentemente dallo stato di una qualsiasi delle macro sopra indicate, la funzione `lv_obj_dump_tree(widget)` fornisce un "dump" del Widget Tree per il Widget specificato (quello stesso Widget più tutti i suoi figli ricorsivamente) col metodo attualmente configurato usato dalla macro `LV_LOG_USER`. Se viene passato NULL invece di un puntatore a un Widget "root", il dump includerà il Widget Tree per tutti gli `Screen`, per tutti i `Display` nel sistema.

Affinché `LV_LOG_USER` produca un output, in `lv_conf.h` deve essere soddisfatta la seguente condizione:

- `LV_USE_LOG` deve essere uguale a un valore diverso da zero
- `LV_LOG_LEVEL <= LV_LOG_LEVEL_USER`

Esaminerà ricorsivamente tutti i figli del Widget (a partire dal Widget stesso) e stamperà l'indirizzo del genitore del Widget, l'indirizzo del Widget e, se `LV_USE_OBJ_ID` è uguale a un valore diverso da zero, stamperà anche l'output di `lv_obj_stringify_id()` per quel Widget.

Questo può essere utile in caso di crash dell'interfaccia utente. Da quel log è possibile esaminare lo stato del Widget Tree quando è stato chiamato `lv_obj_dump_tree(widget)`.

Ad esempio, se un puntatore a un Widget eliminato viene memorizzato nel campo `timer->user_data` di un Timer quando viene chiamato la callback dell'evento timer, il tentativo di utilizzare quel puntatore causerà probabilmente un crash perché il puntatore non è più valido. Tuttavia, un dump tempestivo del Widget Tree subito prima di quel punto mostrerà che il Widget non esiste più.

Trovare un elemento figlio con l'ID

⚠️ Avvertimento

`lv_obj_find_by_id(widget, id)` è deprecato. Per trovare un widget, utilizzare `obj_name`.

`lv_obj_find_by_id(widget, id)` eseguirà un'esplorazione ricorsiva attraverso i figli di `widget` e restituirà il primo figlio incontrato con l'ID specificato.

11.5 Profiler

Con l'aumentare della complessità dell'applicazione, potrebbero sorgere problemi di prestazioni come bassi FPS e frequenti "cache miss" che causano ritardi. LVGL ha configurato internamente alcuni hook per la misurazione delle prestazioni per aiutare gli sviluppatori ad analizzare e individuare i problemi di prestazioni.

11.5.1 Introduzione

LVGL dispone di un sistema di "tracing" nativo per tracciare e registrare i timestamp di eventi importanti che si verificano durante l'esecuzione, come eventi di rendering ed eventi di input utente. Questi timestamp degli eventi vengono utilizzati nelle misurazioni effettuate durante l'analisi delle prestazioni.

Il sistema di "trace" dispone di un buffer di record configurabile che memorizza i nomi delle funzioni evento e i relativi timestamp. Quando il buffer è pieno, il sistema di trace stampa le informazioni di log tramite l'interfaccia utente fornita.

I log di trace in output sono formattati secondo il formato `systrace` di Android e possono essere visualizzati con [Perfetto](#).

11.5.2 Utilizzo

Configurazione

Per abilitare il profiler, si imposta `LV_USE_PROFILER` in `lv_conf.h` e si configurano le seguenti opzioni:

1. Abilitare la funzionalità del profiler integrata impostando `LV_USE_PROFILER_BUILTIN`. Se si dispone del supporto per l'ambiente POSIX, è possibile abilitare `LV_USE_PROFILER_BUILTIN_POSIX`.
2. Configurazione del buffer: Impostare il valore di `LV_PROFILER_BUILTIN_BUF_SIZE` per configurare la dimensione del buffer. Un buffer più grande può memorizzare più informazioni sugli eventi di tracing, riducendo l'interferenza con il rendering. Tuttavia, comporta anche un maggiore consumo di memoria.
3. Configurazione del timestamp: LVGL utilizza la funzione `lv_tick_get()` con una precisione di 1 ms per default per ottenere timestamp quando si verificano eventi. Pertanto, non può misurare con precisione intervalli inferiori a 1 ms. Se l'ambiente di sistema fornisce una precisione maggiore (ad esempio, 1us), è possibile configurare il profiler come segue:

- Configurazione consigliata in ambienti **Arduino**:

```
static uint64_t my_get_tick_cb(void)
{
    /* Use the microsecond time stamp provided by Arduino */
    return micros();
}

void my_profiler_init(void)
{
    lv_profiler_builtin_config_t config;
    lv_profiler_builtin_config_init(&config);
    config.tick_per_sec = 1000000; /* One second is equal to 1000000 microseconds */
    config.tick_get_cb = my_get_tick_cb;
    lv_profiler_builtin_init(&config);
}
```

4. Configurazione dell'output del log: LVGL utilizza l'interfaccia `LV_LOG()` per default per l'output delle informazioni di trace. Per utilizzare un'altra interfaccia per l'output delle informazioni di log (ad esempio `stdout`), è possibile reindirizzare l'output del log utilizzando il seguente codice:

```
static void my_log_print_cb(const char * buf)
{
    printf("%s", buf);
}

void my_profiler_init(void)
{
    lv_profiler_builtin_config_t config;
    lv_profiler_builtin_config_init(&config);
    ... /* other configurations */
    config.flush_cb = my_log_print_cb;
    lv_profiler_builtin_init(&config);
}
```

Eseguire lo scenario di test

Eseguire lo scenario dell'interfaccia utente da misurare, ad esempio lo scorrimento di una pagina "scrollable" verso l'alto e verso il basso o l'ingresso/uscita da un'applicazione.

Elaborazione dei log

Salvare il log di output come `my_trace.txt`, usare `trace_filter.py` per il filtraggio e la pre-elaborazione:

```
./lvgl/scripts/trace_filter.py my_trace.txt
```

o da

```
python3 ./lvgl/scripts/trace_filter.py my_trace.txt
```

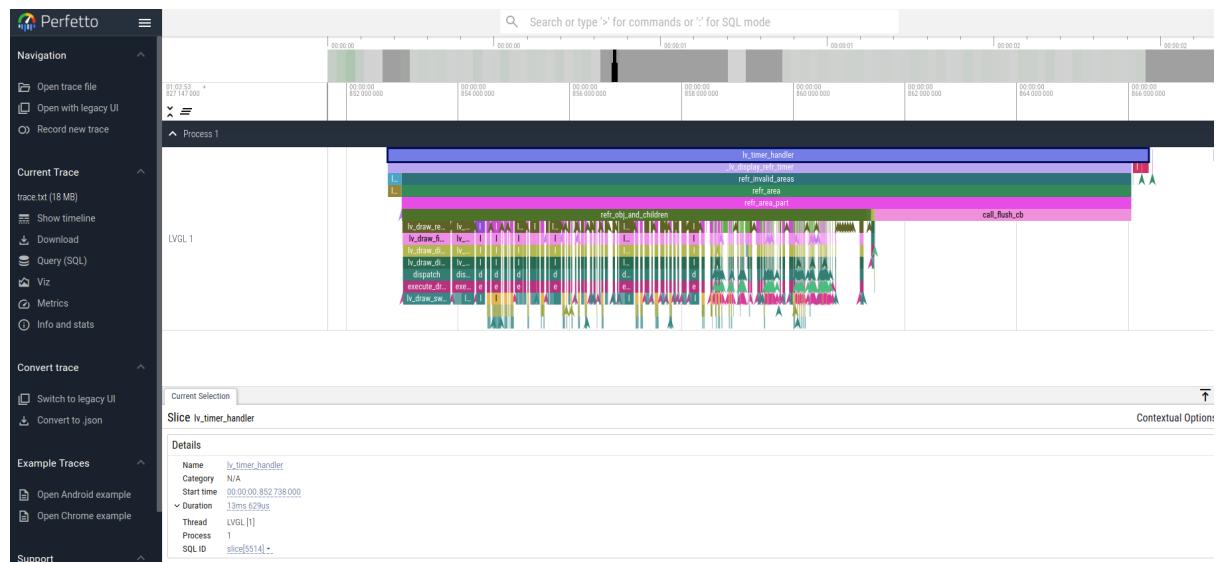
Si otterrà un file di testo processato denominato `trace.systrace`, che contiene approssimativamente il seguente contenuto:

```
# tracer: nop
#
LVGL-1 [0] 2892.002993: tracing_mark_write: B|l|lv_timer_handler
LVGL-1 [0] 2892.002993: tracing_mark_write: B|l|lv_display_refr_timer
LVGL-1 [0] 2892.003459: tracing_mark_write: B|l|refr_invalid_areas
LVGL-1 [0] 2892.003461: tracing_mark_write: B|l|lv_draw_rect
LVGL-1 [0] 2892.003508: tracing_mark_write: E|l|lv_draw_rect
LVGL-1 [0] 2892.003552: tracing_mark_write: B|l|lv_draw_rect
LVGL-1 [0] 2892.003556: tracing_mark_write: E|l|lv_draw_rect
LVGL-1 [0] 2892.003560: tracing_mark_write: B|l|lv_draw_rect
LVGL-1 [0] 2892.003573: tracing_mark_write: E|l|lv_draw_rect
...
```

Importare il file `trace.systrace` elaborato in Perfetto e attendere che venga analizzato.

Analisi delle prestazioni

Se l'analisi del log ha esito positivo, verrà visualizzata una schermata simile alla seguente:



Nell'interfaccia utente di Perfetto, si utilizzano i tasti A e D per spostare orizzontalmente la timeline e i tasti W e S per ingrandire o ridurre la timeline. Utilizzare il mouse per spostare il focus e cliccare sulle funzioni sulla timeline per osservarne il tempo di esecuzione.

11.5.3 Aggiunta di Punti di Misurazione

Gli utenti possono aggiungere le proprie funzioni misurate:

```
void my_function_1(void)
{
    LV_PROFILER_BEGIN();
    do_something();
    LV_PROFILER_END();
}

void my_function_2(void)
{
    LV_PROFILER_BEGIN_TAG("do_something_1");
    do_something_1();
    LV_PROFILER_END_TAG("do_something_1");

    LV_PROFILER_BEGIN_TAG("do_something_2");
    do_something_2();
    LV_PROFILER_END_TAG("do_something_2");
}
```

11.5.4 Implementazione del Profiler Personalizzato

Per utilizzare un metodo di profiler fornito dal sistema operativo, è possibile modificare le seguenti macro in `lv_conf.h`:

- `LV_PROFILER_INCLUDE`: Fornisce un file header per la funzione profiler.
- `LV_PROFILER_BEGIN`: Funzione punto di inizio del profiler.
- `LV_PROFILER_END`: Funzione punto di fine del profiler.
- `LV_PROFILER_BEGIN_TAG`: Funzione punto di inizio del profiler con tag personalizzato.
- `LV_PROFILER_END_TAG`: Funzione punto di fine del profiler con tag personalizzato.

Prendendo come esempio NuttX di RTOS:

```
#define LV_PROFILER_INCLUDE      "nuttx/sched_note.h"
#define LV_PROFILER_BEGIN        sched_note_begin(NOTE_TAG_ALWAYS)
#define LV_PROFILER_END          sched_note_end(NOTE_TAG_ALWAYS)
#define LV_PROFILER_BEGIN_TAG(str) sched_note_beginex(NOTE_TAG_ALWAYS, str)
#define LV_PROFILER_END_TAG(str)  sched_note_endex(NOTE_TAG_ALWAYS, str)
```

11.5.5 FAQ

Il parsing del log di Perfetto fallisce

Verificare la completezza dei log. Se i log fossero incompleti, le cause potrebbero essere le seguenti:

1. Errori di ricezione sulla porta seriale causati da un baud rate elevato. È necessario ridurre il baud rate.
2. Corruzione dei dati causato da log di altri thread inseriti durante la stampa dei log di tracing. È necessario disabilitare l'output del log di altri thread o fare riferimento alla configurazione precedente per utilizzare un'interfaccia di output del log separata.
3. Assicurarsi che la stringa passata da `LV_PROFILER_BEGIN_TAG` o `LV_PROFILER_END_TAG` non sia una variabile locale nello stack o una stringa nella memoria condivisa, poiché attualmente viene registrato solo l'indirizzo della stringa e il contenuto non viene copiato.

Tempo di esecuzione della funzione visualizzato come 0 secondi in Perfetto

Se il tempo di esecuzione della funzione è inferiore alla precisione dei timestamp, questa situazione può verificarsi. È possibile fare riferimento alle istruzioni di configurazione sopra riportate per utilizzare un timestamp con maggiore precisione. Per un esempio, vedere [Configuration](#).

Si verificano notevoli interruzioni durante la profilazione

Quando il buffer utilizzato per memorizzare gli eventi di trace si riempie, il profiler emetterà in output tutti i dati presenti nel buffer, il che può causare blocchi dell'interfaccia utente e interruzioni durante l'output. È possibile ottimizzare questa situazione adottando le seguenti misure:

1. Aumentare il valore di `LV_PROFILER_BUILTIN_BUF_SIZE`. Un buffer più grande può ridurre la frequenza di flushing [svuotamento] dei log, ma consuma anche più memoria.
2. Ottimizzare il tempo di esecuzione delle funzioni di flushing dei log, ad esempio aumentando la velocità in baud della porta seriale o migliorando la velocità di scrittura dei file.

I log di trace non vengono stampati

Se i log di trace non vengono stampati automaticamente quando il buffer non è pieno, è possibile provare i seguenti metodi per forzare l'output del log:

1. Ridurre il valore di `LV_PROFILER_BUILTIN_BUF_SIZE` per riempire il buffer più rapidamente e attivare la stampa automatica.
2. Chiamare manualmente o utilizzare un timer per chiamare la funzione `lv_profiler_builtin_flush()` per forzare lo svuotamento del log.

11.6 Monitor di Sistema (sysmon)

Il modulo System Monitor fornisce il monitoraggio in tempo reale delle metriche delle prestazioni del sistema direttamente sul display. Supporta sia il monitoraggio delle prestazioni (utilizzo della CPU e FPS) sia il monitoraggio della memoria (memoria utilizzata e frammentazione).

11.6.1 Dipendenze

- Richiede LV_USE_LABEL = 1 in lv_conf.h
- Richiede LV_USE_OBSERVER = 1 in lv_conf.h
- Richiede LV_USE_SYSMON = 1 in lv_conf.h

11.6.2 Utilizzo

Configurazione

Abilitare in lv_conf.h:

```
/* Main sysmon enable */
#define LV_USE_SYSMON 1

/* Performance monitor (CPU% and FPS) */
#define LV_USE_PERF_MONITOR 1

/* Memory monitor (used + fragmentation) */
#define LV_USE_MEM_MONITOR 1

/* Optional: refresh period in ms */
#define LV_SYSMON_REFR_PERIOD_DEF 300

/* Optional: log to console instead of screen */
#define LV_USE_PERF_MONITOR_LOG_MODE 0
```

Creazione di Monitor

```
/* Create generic monitor */
lv_obj_t * sysmon = lv_sysmon_create(lv_display_get_default());

/* Create performance monitor */
lv_sysmon_show_performance(NULL); /* NULL = default display */

/* Create memory monitor */
lv_sysmon_show_memory(NULL);
```

Monitor delle Prestazioni

Tracce:

- FPS (Frames Per Second)
- CPU usage (%)
- Render time (ms)
- Flush time (ms)
- Self CPU usage (%) if enabled

Formato del Display:

```
32 FPS, 45% CPU
8 ms
```

Dove:

- Riga 1: FPS, Total CPU%
- Riga 2: Total time (Render | Flush)

11.6.3 Pausa e Ripristino

`lv_sysmon_performance_pause(disp)` mette in pausa il perf monitor.

`lv_sysmon_performance_resume(disp)` riprende il perf monitor.

Monitor della Memoria Memory

Display:

- Current memory usage (kB and %)
- Peak memory usage (kB)
- Fragmentation (%)

Formato del Display:

```
24.8 kB (76%)
32.4 kB max, 18% frag.
```

Posizionamento

Configura le posizioni in lv_conf.h:

```
/* Top-right corner */
#define LV_USE_PERF_MONITOR_POS LV_ALIGN_TOP_RIGHT

/* Bottom-right corner */
#define LV_USE_MEM_MONITOR_POS LV_ALIGN_BOTTOM_RIGHT
```

11.6.4 Dettagli di Implementazione

Inizializzazione

Gestisce:

- Monitoraggio della memoria globale (`sysmon_mem`)
- Strutture delle prestazioni per display

Misurazione delle Prestazioni

Raccolta basata su eventi:

Evento	Misurazione
LV_EVENT_REFR_START	Inizio intervallo di aggiornamento
LV_EVENT_REFR_READY	Registra la durata del refresh
LV_EVENT_RENDER_START	Ora inizio rendering
LV_EVENT_RENDER_READY	Registra la durata del render
LV_EVENT_FLUSH_*	Misurazione delle operazioni di flush

Timer

- Prestazioni: `perf_update_timer_cb`
- Memoria: `mem_update_timer_cb`
- Periodo di default: 300ms (`LV_SYSMON_REFR_PERIOD_DEF`)

11.7 Test dell'interfaccia utente

11.7.1 Panoramica

Il modulo di Test fornisce funzioni per emulare clic, pressioni di tasti, rotazioni dell'encoder, il trascorrere del tempo e confrontare l'interfaccia utente con immagini di riferimento.

Queste funzioni possono essere facilmente utilizzate in qualsiasi framework di test (come Unity, GoogleTest, ecc.) e le asserzioni possono essere eseguite per verificare, ad esempio:

- Il valore di un widget è diverso dal valore previsto dopo l'emulazione degli input dell'utente.
- I valori sono errati dopo un certo periodo di tempo.
- Il contenuto dello schermo è diverso dall'immagine di riferimento.
- Alcune funzioni evento non vengono attivate.
- Ecc.

Si presuppone che i test vengano eseguiti in un ambiente desktop o server, dove non ci sono vincoli di memoria.

11.7.2 Utilizzo

Il modulo Test può essere abilitato configurando `LV_USE_TEST` su un valore diverso da zero ed è costituito dai seguenti componenti:

- Helper
- Emulazione del display
- Emulazione del dispositivo di input
- Confronto degli screenshot

Helper

Tempo

Per emulare il tempo trascorso, è possibile utilizzare due funzioni:

1. `lv_test_wait(ms)`: Emula che siano trascorsi `ms` millisecondi, ma chiama anche `lv_timer_handler` dopo ogni millisecondo. Questo è utile per verificare se gli eventi (ad esempio, pressione prolungata, ripetizione pressione prolungata) e i timer sono stati attivati correttamente nel tempo.
2. `lv_test_fast_forward(ms)`: Salta in avanti di `ms` millisecondi e chiama `lv_timer_handler` solo una volta alla fine.

`lv_refr_now(NULL)` viene chiamato alla fine di entrambe le funzioni per garantire che le animazioni e le coordinate dei widget vengano ricalcolate.

`lv_refr_now(NULL)` può anche essere chiamato manualmente per forzare LVGL ad aggiornare il display emulato.

Utilizzo della Memoria

Se `LV_USE_STDLIB_MALLOC` è impostato su `LV_STDLIB_BUILTIN`, è possibile monitorare l'utilizzo della memoria e i "leak" [perdite] di memoria.

```
size_t mem1 = lv_test_get_free_mem();
<create and delete items>
size_t mem2 = lv_test_get_free_mem();
if(mem1 != mem2) fail();
```

Potrebbe essere opportuno creare ed eliminare elementi in un ciclo più volte e aggiungere una piccola tolleranza al test di perdita di memoria. Questo potrebbe essere necessario a causa della potenziale frammentazione della memoria. Empiricamente, si consiglia una tolleranza di 32 byte.

```
if(LV_ABS((int64_t)mem2 - (int64_t)mem1) > 32) fail();
```

Emulazione del Display

Chiamando `lv_test_display_create(hor_res, ver_res)`, è possibile creare un display fittizio.

Funziona come qualsiasi altro display normale, ma il suo contenuto esiste solo in memoria.

Quando si crea questo display, è necessario passare le risoluzioni orizzontale e verticale. Internamente, verrà allocato un framebuffer per questa dimensione e verrà impostato il formato colore XRGB8888.

La risoluzione e il formato del colore possono essere modificati in qualsiasi momento chiamando `lv_display_set_resolution()` e `lv_display_set_color_format()`.

Emulazione del Dispositivo di Input

Chiamando `lv_test_indev_create_all()`, verranno creati tre dispositivi di input di test:

1. Un puntatore (per touch o mouse)
2. Una tastiera
3. Un encoder

Ad esempio, ecco come può essere emulato una "gesture" di scorrimento:

```
lv_test_mouse_move_to(20, 30);
lv_test_mouse_press();
lv_test_wait(20);
lv_test_mouse_move_by(0, 100);
lv_test_wait(20);
lv_test_mouse_release();
lv_test_wait(20);
```

Si consiglia di aggiungere `lv_test_wait()` dopo le azioni dell'utente per garantire che il nuovo stato e le nuove coordinate vengano letti e applicati dal dispositivo di input.

Dopodiché, l'utente può verificare se il widget specificato è stato effettivamente "scrollato" ottenendo la coordinata Y di un elemento figlio.

```
int32_t y_start = lv_obj_get_y(child);
<scroll emulation>
int32_t y_end = lv_obj_get_y(child);
if(y_start + 100 != y_end) fail();
```

Per l'elenco delle funzioni di emulazione dei dispositivi di input supportate, fare riferimento al link `lv_test_indev.h` qui sotto (nella sezione [API](#)).

Confronto degli Screenshot

`bool lv_test_screenshot_compare(const char * fn_ref)` è una funzione utile per confrontare il contenuto del display emulato con le immagini PNG di riferimento.

Questa funzione ha un funzionamento pratico:

- Se le cartelle a cui si fa riferimento in `fn_ref` non esistono, verranno create automaticamente.
- Se l'immagine di riferimento non viene trovata, verrà creata automaticamente dalla schermata renderizzata.
- Se il confronto fallisce, verrà creato un file `<image_name>_err.png` con il contenuto renderizzato accanto all'immagine di riferimento.
- Se il confronto fallisce, verranno stampate anche le coordinate X e Y del primo pixel divergente, insieme ai colori effettivi e previsti.

Le immagini PNG di riferimento devono avere un **formato colore a 32 bit** e corrispondere alle dimensioni del display.

Il contenuto del display di test verrà convertito in XRGB8888 per semplificare il confronto con le immagini di riferimento. La conversione è supportata dai seguenti formati (ad esempio, la visualizzazione di test dovrebbe avere un formato colore in questo elenco):

- `LV_COLOR_FORMAT_XRGB8888`
- `LV_COLOR_FORMAT_ARGB8888`
- `LV_COLOR_FORMAT_ARGB8888_PREMULTIPLIED`
- `LV_COLOR_FORMAT_RGB888`
- `LV_COLOR_FORMAT_RGB565`

- `LV_COLOR_FORMAT_RGB565_SWAPPED`
- `LV_COLOR_FORMAT_L8`
- `LV_COLOR_FORMAT_AL88`
- `LV_COLOR_FORMAT_I1`

Per leggere e decodificare le immagini PNG e per memorizzare l'immagine renderizzata convertita, alcuni MB di RAM vengono allocati dinamicamente utilizzando il `malloc` standard (non `lv_malloc`).

Il confronto degli screenshot utilizza `lodepng`, integrato in LVGL e che deve essere semplicemente abilitato tramite `LV_USE_LODEPNG`.

Per evitare di rendere l'intero modulo Test dipendente da `lodepng`, il confronto degli screenshot può essere abilitato singolarmente tramite `LV_USE_TEST_SCREENSHOT_COMPARE`.

11.7.3 API

`lv_test_display.h`
`lv_test_helpers.h`
`lv_test_indev.h`
`lv_test_indev_gesture.h`
`lv_test_private.h`
`lv_test_screenshot_compare.h`

11.8 Simulatore GPU VG-Lite

LVGL integra un simulatore di GPU VG-Lite basato su ThorVG. Il suo scopo è semplificare il debug dell'adattamento di GPU VG-Lite e ridurre il tempo impiegato per il debug e l'individuazione dei problemi sui dispositivi hardware.

È stato integrato nel processo di compilazione e test automatizzato di CI per garantire che il backend di rendering VG-Lite possa essere testato completamente dopo ogni Pull Request (PR) inserita nel repository.

11.8.1 Come Funziona

Utilizzando il file header `vg_lite.h`, ThorVG reimplementa l'API VG-Lite, generando le stesse immagini renderizzate dell'hardware GPU VG-Lite reale.

11.8.2 Configurazione

1. Abilitare il backend di rendering VG-Lite, vedere [GPU Generica VG-Lite](#).
2. Abilitare ThorVG e attivare la configurazione `LV_USE_THORVG_INTERNAL` o `LV_USE_THORVG_EXTERNAL`. Si consiglia di utilizzare la libreria ThorVG interna per garantire risultati di rendering uniformi.
3. Abilitare `LV_USE_VG_LITE_THORVG` e impostare `LV_DRAW_BUF_ALIGN` a 64. Le restanti opzioni possono rimanere quelle di default. Assicurarsi che `LV_VG_LITE_USE_GPU_INIT` sia abilitato, perché il contesto di disegno thorvg deve essere inizializzato prima di poter essere utilizzato.

CAPITOLO 12

Contribuire

12.1 Introduzione

Unirsi alla community di LVGL e lasciare il segno nella libreria!

Ci sono molti modi per contribuire a LVGL, anche se si è nuovi della libreria o anche solo un neofita della programmazione.

Potrebbe spaventare fare il primo passo, ma non c'è nulla di cui aver paura. Una community amichevole e disponibile ed in attesa. Si incontrano persone che la pensano in modo simile e si crea qualcosa di grande insieme.

Quindi, scopriamo quale opzione di contributo è più adatta ed è d'aiuto a partecipare allo sviluppo di LVGL!

12.2 Modi per Contribuire

- **Spargere la voce:** Condividere la propria esperienza con LVGL con gli amici o sui social media per aumentarne la visibilità.
- **Star LVGL** Dare una stella su [GitHub!](#) Aiuta molto a rendere LVGL più interessante per i nuovi arrivati.
- **Segnalare un Bug***: Aprire un [GitHub Issue](#) se qualcosa non funziona.
- **Unirsi al Nostro Forum** : Per incontrare altri sviluppatori e porre le proprie domande.
- **Raccontare le proprie idee:** Se si ritiene che manchi qualcosa in LVGL, saremo lieti di riceverlo tramite un [GitHub Issue](#)
- **Sviluppo di Funzionalità:** Aiutare a progettare o sviluppare una funzionalità. Vedere di seguito.

Le problematiche di media e larga scala vengono discusse nelle problematiche della [Feature Planning](#).

Una issue [problematica] può essere sviluppata quando tutte le domande nel "issue template" hanno ricevuto risposta e non ci sono obiezioni da parte di alcun membro principale.

Utilizziamo le etichette GitHub per mostrare lo stato e gli attributi delle problematiche e delle Pull Request. Per cercare opportunità per contribuire, si può [Filtrare per queste etichette](#) :

- **Simple:** Ottima scelta per iniziare con un contributo LVGL
- **PR needed:** Abbiamo esaminato il problema, ma deve ancora essere implementato
- **Review needed:** È stata aperta una Pull Request che necessita di revisione/test

12.3 Pull Request

Il merging di nuovo codice in `lvgl/lvgl` e in altri repository avviene tramite *Pull Request* (in breve PR). Una PR è una notifica del tipo "Ehi, ho apportato degli aggiornamenti al progetto. Ecco le modifiche, volendo, si possono aggiungere". Per fare ciò, c'è bisogno di una copia (chiamata fork) del progetto originale nel proprio account, apportare alcune modifiche e notificare al repository originale gli aggiornamenti. Si può vedere come appare su GitHub per LVGL qui: <https://github.com/lvgl/lvgl/pulls>.

Per aggiungere le modifiche, si possono modificare i file online su GitHub e inviare una nuova richiesta Pull da lì (consigliato per piccole modifiche) oppure aggiungere gli aggiornamenti nel proprio editor/IDE e usare git per pubblicare le modifiche (consigliato per aggiornamenti più complessi).

12.3.1 Da GitHub

1. Accedere al file da modificare.
2. Cliccare sul pulsante Edit nell'angolo in alto a destra.
3. Aggiungere le modifiche al file.
4. Aggiungere un messaggio di commit in fondo alla pagina.
5. Cliccare sul pulsante *Propose changes*.

12.3.2 Dalla Workstation Locale

Queste istruzioni descrivono il repository principale `lvgl`, ma funzionano allo stesso modo di qualsiasi repository Git remoto.

1. Eseguire il fork del repository [lvgl repository](#). Per farlo, cliccare sul pulsante "Fork" nell'angolo in alto a destra. Il repository `lvgl` verrà "copiato" sul proprio account GitHub (https://github.com/<YOUR_NAME>?tab=repositories)
2. Clonare il repository "forkato".
3. Aggiungere le modifiche. Si può creare un *feature branch* dal branch `master` per gli aggiornamenti: `git checkout -b <new-feature-branch-name>`
4. Eseguire il commit e il push delle modifiche nel proprio repository `lvgl` forkato.
5. Creare una PR su GitHub dalla pagina del proprio repository `lvgl` forkato (https://github.com/<YOUR_NAME>/lvgl) cliccando sul pulsante "*New pull request*". Non dimenticare di selezionare il branch in cui sono state aggiunte le modifiche.
6. Impostare il branch di base in cui si desidera unire l'aggiornamento. Nel repository `lvgl`, sia le correzioni che le nuove funzionalità devono essere indirizzate al branch `master`.
7. Descrivere il contenuto dell'aggiornamento. Se applicabile, è gradito un codice di esempio.
8. Per apportare ulteriori modifiche, aggiornare il proprio repository `lvgl` forkato con nuovi commit. Appariranno automaticamente nella PR.

12.3.3 Formato dei Messaggi di Commit

Il formato dei messaggi di commit si ispira all'[Angular Commit Format](#).

È necessario utilizzare la seguente struttura:

```
<type>(<scope>): <subject>
  ---- blank line
<body>
  ---- blank line
<footer>
```

Possibili `<type>`:

- fix correzione di bug nel codice sorgente LVGL
- feat nuova funzionalità

- `arch` modifiche architetturali
- `perf` modifiche che influiscono sulle prestazioni
- `example` qualsiasi elemento relativo agli esempi (inclusi fix e nuovi esempi)
- `docs` qualsiasi elemento relativo alla documentazione (inclusi fix, formattazione e nuove pagine)
- `test` qualsiasi elemento relativo ai test (test nuovi e aggiornati o azioni di CI)
- `chore` qualsiasi modifica minore di formattazione o stile che renderebbe il changelog poco chiaro

`<scope>` è il nome del modulo, file o sottosistema interessato dal commit. Di solito è una sola parola e può essere scelta liberamente. Ad esempio `img`, `layout`, `txt`, `anim`. Lo scope può essere omesso.

`<subject>` contiene una breve descrizione della modifica seguendo queste linee guida:

- usare il modo imperativo: ad esempio, il presente "change", non "changed" né "changes";
- non usare la maiuscola per la prima lettera;
- nessun punto (.) alla fine;
- massimo 90 caratteri.

`<body>` è facoltativo e può essere utilizzato per descrivere i dettagli di questa modifica.

`<footer>` deve contenere:

- iniziare con "BREAKING CHANGE" se le modifiche 'rompono' l'API;
- riferimento al problema di GitHub o alla Pull Request, se applicabile. (Per i dettagli, vedere [Linking a pull request to an issue](#)).

Alcuni esempi:

```
fix(image): update size when a new source is set
```

```
fix(bar): fix memory leak
The animations weren't deleted in the destructor.
Fixes: #1234
```

```
feat: add span widget
The span widget allows mixing different font sizes, colors and styles.
It's similar to HTML <span>
```

```
docs(porting): fix typo
```

12.4 Developer Certification of Origin (DCO)

12.4.1 Panoramica

Per garantire che tutti i criteri di licenza siano soddisfatti per ogni repository del progetto LVGL, applichiamo un processo chiamato DCO (Developer's Certificate of Origin [Certificato di Origine dello Sviluppatore]).

Il testo del DCO è disponibile qui: <https://developercertificate.org/>.

Contribuendo a qualsiasi repository del progetto LVGL, si accetta che il contributo sia conforme al DCO.

Se il contributo soddisfa i requisiti del DCO, non sono necessarie ulteriori azioni. In caso di dubbi, non esitare a contattarci in un commento, ad esempio nella [Pull Request](#) inviata.

12.4.2 Licenze Accettate e Note di Copyright

Per rendere il DCO più facile da comprendere, ecco alcune guide pratiche su casi specifici:

Il proprio lavoro

Il caso più semplice è quando il contributo è esclusivamente frutto del proprio lavoro. In questo caso si può semplicemente inviare una Pull Request senza preoccuparsi di problemi di licenza.

Utilizzo di codice da un sorgente online

Se il codice da aggiungere è basato su un articolo, un post o un commento su un sito web (ad esempio StackOverflow), è necessario seguire la licenza e/o le regole di quel sito.

Ad esempio, nel caso di StackOverflow, è possibile utilizzare una nota come questa:

```
/* The original version of this code-snippet was published on StackOverflow.
 * Post: http://stackoverflow.com/questions/12345
 * Author: http://stackoverflow.com/users/12345/username
 * The following parts of the snippet were changed:
 * - Check this or that
 * - Optimize performance here and there
 */
... code snippet here ...
```

Utilizzo di codice con licenza MIT

Poiché LVGL è con licenza MIT, altro codice con licenza MIT può essere integrato senza problemi. La licenza MIT richiede l'aggiunta di una nota di copyright all'opera derivata. Qualsiasi opera derivata basata su codice con licenza MIT deve copiare il file di licenza o il testo dell'opera originale.

Utilizzare codice con licenza GPL

La licenza GPL non è compatibile con la licenza MIT. Pertanto, LVGL non può accettare codice con licenza GPL.

12.5 Stile di Codifica

12.5.1 File Template

Usare `misc/lv_tmpl.c` e `misc/lv_tmpl.h`

12.5.2 Convenzioni sui Nomi

- Le parole sono separate da `' '`
- Nei nomi di variabili e di funzioni utilizzare solo lettere minuscole (ad esempio `height_tmp`)
- Negli enum e nelle definizioni utilizzare solo lettere maiuscole (ad esempio `MAX_LINE_NUM`)
- Nomi globali (API):
 - iniziano con `lv`
 - seguiti dal nome del modulo: `button`, `label`, `style` ecc.
 - seguiti dall'azione (per le funzioni): `set`, `get`, ecc.
 - chiuso col soggetto: `name`, `size`, `state` ecc.
- Typedef
 - preferire `typedef struct` e `typedef enum` invece di `struct name` e `enum name`
 - `typedef struct` e `typedef enum` terminano sempre con `_t`
- Abbreviazioni:
 - Sono utilizzate e consentite le seguenti abbreviazioni:
 - * `dsc` descrittore
 - * `param` parametro
 - * `indev` device di input

- * anim animazione
- * buf buffer
- * str stringa
- * min/max minimo/massimo
- * alloc allocazione
- * ctrl control
- * pos posizione
- Evitare di aggiungere nuove abbreviazioni

12.5.3 Guida alla Codifica

- Editor:
 - Impostare l'editor in modo che utilizzi 4 spazi per le rientranze di tabulazione (invece dei caratteri di tabulazione).
 - Eccezione: il file **Kconfig** e tutti i file make richiedono caratteri di tabulazione iniziali sugli elementi figli.
- Funzioni:
 - Scrivere funzioni che utilizzano il principio di singola responsabilità.
 - Rendere le funzioni **static** quando non fanno parte dell'API pubblica di quell'oggetto (ove possibile).
- Variabili:
 - Una riga, una dichiarazione (DA NON FARE: char x, y;).
 - Usare <stdint.h> (*uint8_t*, *int32_t* ecc.).
 - Dichiarare le variabili dove necessario (non tutte all'inizio della funzione).
 - Utilizzare lo scope più piccolo richiesto.
 - Le variabili in un file (al di fuori delle funzioni) sono sempre *static*.
 - Non utilizzare variabili globali (utilizzare funzioni per impostare/ottenere variabili statiche).

12.5.4 Commenti

Prima di ogni prototipo di funzione nei file .h, includere un commento formattato in Doxygen come questo:

```
/**  
 * Brief description. Add a blank line + additional paragraphs when more detail is needed.  
 * @param parent brief description of argument. Additional detail can appear  
 * on subsequent lines. List of accepted values:  
 * - value one  
 * - value two  
 * - value three  
 * @return brief description of return value.  
 */  
type_name_t * lv_function_name(lv_obj_t * parent);
```

Il normale prefisso di commento `/**` fa sì che il commento documenti il pezzo di codice *dopo* il commento. Quando si documenta un pezzo di codice che si trova *prima* del commento, come un membro di una struttura, usare `/**<` in questo modo:

```
/**  
 * Brief description of struct  
 *  
 * When more detail is needed, add a blank line then the detail.  
 */  
typedef struct {  
    char     *text;  /**< Brief description of this member */  
    uint16_t length; /**< Brief description of this member */  
} lv_example_type_t;
```

- Quando si commenta il codice, usare commenti a blocchi come questo `/* Description */`, non commenti di fine riga come questo `// Description`.

- Includere uno spazio dopo /* o /**< e prima di */ per migliorare la leggibilità.
- Scrivere codice leggibile per evitare commenti descrittivi come: x++; /* Aggiunge 1 a x */.
- Il codice dovrebbe mostrare chiaramente cosa si sta facendo.
- Si dovrebbe scrivere il perché è stato fatto: x++; /* Point to closing '\0' of string */
- Sono accettati brevi "riassunti" di poche righe: /* Calculate new coordinates */
- Nei commenti, utilizzare le virgolette inverse (...) quando si fa riferimento a un elemento del codice, come una variabile, un tipo o il nome di una struttura: /* Update value of `x_act` */
- Quando si aggiungono o modificano commenti, le priorità sono (in ordine di importanza):
 1. chiarezza (la facilità con cui altri programmati possano comprendere le vostre intenzioni),
 2. leggibilità (la facilità con cui altri programmati possano leggere i vostri commenti),
 3. brevità (la qualità di usare poche parole quando si parla o si scrive).
- Le righe vuote all'interno dei commenti sono auspicabili quando migliorano la chiarezza e la leggibilità.
- Da ricordare, quando si scrive codice sorgente, non si sta solo insegnando al computer cosa fare, ma anche insegnando ad altri programmati come funziona, non solo agli utenti dell'API, ma anche ai futuri manutentori del codice sorgente. I commenti aggiungono informazioni su cosa si stava pensando quando è stato scritto il codice e perché sono state fatte le cose in quel modo, informazioni che non possono essere trasmesse dal solo codice sorgente.

Specifiche dei Commenti di Doxygen

Doxygen è il primo programma di una catena che genera la documentazione online dell'API LVGL dai file nel repository LVGL. Doxygen rileva i file a cui prestare attenzione tramite il comando @file all'interno di un commento Doxygen. I commenti Doxygen iniziano con /**. Ignora i commenti che non contengono esattamente due *.

Di seguito è riportata l'illustrazione di un prototipo di funzione API con documentazione che illustra la maggior parte dei comandi Doxygen utilizzati in LVGL.

```
/**  
 * Set alignment of Widgets placed in containers with LV_STYLE_FLEX_FLOW style.  
 *  
 * The values for the '..._place' arguments come from the 'lv_flex_align_t'  
 * enumeration and have the same meanings as they do for flex containers in CSS.  
 * @param obj  
 *          pointer to flex container. It must have  
 *          'LV_STYLE_FLEX_FLOW' style or nothing will happen.  
 * @param main_place  
 *          where to place items on main axis (in their track).  
 *          (Any value of 'lv_flex_align_t').  
 * @param cross_place  
 *          where to place item in track on cross axis.  
 *          - 'LV_FLEX_ALIGN_START'  
 *          - 'LV_FLEX_ALIGN_END'  
 *          - 'LV_FLEX_ALIGN_CENTER'  
 * @param track_cross_place  
 *          where to place tracks in cross direction.  
 *          (Any value of 'lv_flex_align_t').  
 * Example for a title bar layout:  
 * @code[c]  
 *     lv_obj_set_flex_align(title_bar, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);  
 * @endcode  
 * @see  
 *     - https://css-tricks.com/snippets/css/a-guide-to-flexbox/  
 *     - see 'lv_obj_set_flex_grow()' for additional information.  
 */  
void lv_obj_set_flex_align(lv_obj_t * widget, lv_flex_align_t main_place, lv_flex_align_t cross_place,  
                           lv_flex_align_t track_cross_place);
```

- Iniziare sempre un commento Doxygen con una breve descrizione dell'elemento di codice che documenta.
- Quando sono necessari maggiori dettagli, aggiungere una riga vuota sotto la breve descrizione e aggiungere ulteriori informazioni che potrebbero essere necessarie agli utenti dell'API LVGL, incluse le precondizioni per la chiamata della funzione. Doxygen necessita della riga vuota per separare le parti "brevi" da quelle di "dettaglio" della descrizione.
- Descrivere i parametri della funzione col comando @param. Quando una funzione scrive all'indirizzo contenuto da un parametro puntatore, se non è già ovvio (ad esempio, quando il nome del parametro contiene la parola "out"), includere la direzione nel comando per chiarezza:

@param[out] param_name descrizione.

- Descrivere i valori restituiti col comando @return.
- Aggiungere almeno 2 spazi dopo i comandi Doxygen per una migliore leggibilità.

- Utilizzare le virgolette inverse ('...') attorno agli elementi del codice (variabili, nomi di tipo, nomi di funzione). Per i nomi di tipo e di funzione, Doxygen genera un collegamento ipertestuale alla documentazione di quel membro del codice (quando esiste), con o senza le virgolette inverse singole.
- Aggiungere un "()" vuoto ai nomi di funzione. Doxygen non genererà un collegamento ipertestuale alla documentazione della funzione senza questo.
- Utilizzare una grammatica corretta per chiarezza. Le descrizioni dei parametri non necessitano di punti dopo di esse, a meno che non siano frasi complete.
- Allineare i bordi del testo attorno agli elenchi di parametri per facilitarne la lettura.
- Gli elenchi (ad esempio, dei valori dei parametri accettati) possono essere creati utilizzando il carattere '-'. Se l'elenco deve essere numerato, è possibile utilizzare anche i numeri.
- Inserire il codice di esempio in un blocco di codice circondandolo con i comandi @code{ . c} e @endcode. Doxygen non necessita della parte { . c}, ma il software a valle sì.
- Indirizzare il lettore a informazioni aggiuntive utilizzando il comando @see. Doxygen aggiunge un paragrafo "See also" [Vedere anche]. Il testo che segue il comando @see verrà indentato.
- Se si crea una nuova coppia di file .c e .h (ad esempio per un nuovo driver), assicurarsi che un commento Doxygen come questo sia presente all'inizio di ogni nuovo file. Doxygen non analizzerà il file senza il comando @file.

```
/**  
 * @file filename.c  
 *  
 */
```

Comandi Doxygen Supportati

- @file indica a Doxygen di analizzare questo file e, se applicabile, fornisce anche la documentazione sul file stesso (tutto ciò che lo segue nello stesso commento).
- @param name description documenta name come parametro di funzione, e description è il testo che lo segue finché Doxygen non incontra una riga vuota o un altro comando Doxygen.
- @return description documenta il valore di ritorno finché Doxygen non incontra una riga vuota o un altro comando Doxygen.
- @code{ . c}/@endcode racchiude il codice che deve essere inserito in un blocco di codice. Mentre Doxygen sa utilizzare la codifica a colori C per i blocchi di codice in un file .C, la parte a valle della sequenza di generazione della documentazione non lo fa, quindi l'aggiunta di { . c} al comando @code è necessaria.
- @note text inizia un paragrafo in cui è possibile inserire una nota. La nota termina con una riga vuota, la fine del commento o un altro comando Doxygen che inizia una nuova sezione. Se la nota contiene più di un paragrafo, è possibile aggiungerne altri utilizzando altri comandi @note. Al momento della stesura di questo documento, i comandi @par non aggiungono paragrafi aggiuntivi alle note, come indicato nella documentazione di Doxygen.
- @see text genera un paragrafo "See also" [Vedere anche] in una sezione evidenziata, utile quando si trovano altre prove delle informazioni aggiuntive su un argomento.

12.5.5 Convenzioni API

Per supportare la generazione automatica dei binding, l'API C LVGL deve seguire alcune convenzioni di codifica:

- Utilizzare enum invece delle macro. Se è inevitabile l'uso dei define esportarli con LV_EXPORT_CONST_INT(`defined_value`) subito dopo il define.
- Negli argomenti delle funzioni, utilizzare la dichiarazione type name[] per i parametri dell'array invece di type * name
- Utilizzare puntatori tipizzati invece di puntatori void*
- Il costruttore del Widget deve seguire il pattern lv_<widget_name>_create(lv_obj_t * parent).

- Le funzioni membro del Widget devono iniziare con `lv_<widget_name>` e devono ricevere `lv_obj_t*` come primo argomento, che è un puntatore all'oggetto Widget stesso.
- Le API `struct` devono seguire le convenzioni dei widget. Questo significa ricevere un puntatore alla `struct` come primo argomento, e il prefisso del nome della `struct` dovrebbe essere usato anche come prefisso del nome della funzione (ad esempio `lv_display_set_default(lv_display_t * disp)`)
- Le funzioni e le `struct` che non fanno parte dell'API pubblica devono iniziare con un carattere di sottolineatura per essere contrassegnate come "private".
- Anche l'argomento deve essere nominato nei file H.
- Non usare `malloc` in una variabile statica o globale. Invece, dichiarare la variabile nella struttura `lv_global_t` in `lv_global.h` e contrassegnarla con `(LV_GLOBAL_DEFAULT()->variable)` quando viene utilizzata.
- Per registrare e utilizzare le callback, è necessario seguire una delle seguenti procedure.
 - Passare un puntatore a una `struct` come primo argomento sia della funzione di registrazione che della callback. Tale `struct` deve contenere il campo `void * user_data`.
 - L'ultimo argomento della funzione di registrazione deve essere `void * user_data` e lo stesso `user_data` deve essere passato come ultimo argomento della callback.

Per ulteriori informazioni, consultare la documentazione di [MicroPython](#).

12.5.6 Formattazione

Ecco un esempio per mostrare il posizionamento delle parentesi e l'uso degli spazi:

```
/**  
 * Set new text for a label. Memory will be allocated by label to store text.  
 *  
 * @param label pointer to label Widget  
 * @param text '\0' terminated character string.  
 *           NULL to refresh with current text.  
 */  
void lv_label_set_text(lv_obj_t * label, const char * text)  
{ /* Main brackets of functions in new line */  
  
    if(label == NULL) return; /* No bracket only if command is inline with if statement */  
  
    lv_obj_inv(label);  
  
    lv_label_ext_t * ext = lv_obj_get_ext(label);  
  
    /* Comment before a section */  
    if(text == ext->txt || text == NULL) { /* Bracket of statements starts on same line */  
        lv_label_refr_text(label);  
        return;  
    }  
  
    ...  
}
```

È possibile utilizzare `astyle` per formattare il codice. Eseguire `code-format.py` dalla cartella `scripts`.

12.5.7 Include

Vari sottosistemi di LVGL possono essere abilitati o disabilitati impostando una macro in `lv_conf.h` rispettivamente su 1 o 0. I file di codice che contengono la logica per tali sottosistemi sono spesso organizzati in modo che vi sia una direttiva `#if <ENABLING_MACRO_NAME>` all'inizio del file e la sua corrispondente `#endif` alla fine. Se si aggiunge o si modifica un sottosistema di questo tipo in LVGL, ove possibile, l'unica `#include` che dovrebbe trovarsi sopra tali direttive condizionali dovrebbe essere sufficiente a includere la macro di abilitazione/disabilitazione. In particolare:

- nel file `.c`: l'`#include` che include l'header con la corrispondenza più vicina a quel file `.c`
- nel file `.h` corrispondente: `#include "lv_conf_internal.h"`

che, a sua volta, include `lv_conf.h`. Vedere gli esempi in `lv_freetype.c`, `lv_freetype_private.h` e `lv_freetype.h`.

12.5.8 pre-commit

`pre-commit` è un gestore di pacchetti multilinguaggio per gli hook di pre-commit. Consultare la [guida](#) per installare il pacchetto Python di pre-commit sul computer di sviluppo.

Una volta installato `pre-commit` si dovranno configurare gli script degli hook di git con:

```
pre-commit install
```

ora pre-commit verrà eseguito automaticamente su git commit!

12.5.9 Hook

L'hook locale `format-source` (vedere `.pre-commit-config.yaml`) esegue `astyle` su tutti i file sorgente e header in staging (che non sono esclusi, vedere la chiave `exclude` della configurazione dell'hook) prima di inserire il messaggio di commit. Se un file viene formattato da `astyle` sarà necessario aggiungere la modifica all'area di staging ed eseguire nuovamente `git commit`.

L'hook `trailing-whitespace` corregge gli spazi vuoti finali su tutti i file.

12.5.10 Saltare gli hook

Si può saltare un hook in particolare, con:

```
SKIP=name-of-the-hook git commit
```

12.5.11 Hook di test

Non è necessario eseguire un commit per testare gli hook, si possono testare aggiungendo i file nell'area di staging ed eseguendo:

```
pre-commit run name-of-the-hook
```

CAPITOLO 13

Log delle Modifiche

13.1 v9.4.0 16 Ottobre 2025

LVGL v9.4.0 offre significativi miglioramenti delle prestazioni, supporto GPU ampliato e nuove funzionalità per le moderne piattaforme embedded e MPU.

13.1.1 Caratteristiche Principali

- **Supporto glTF + 3D.** Carica i modelli 3D glTF direttamente nelle interfacce utente LVGL per interfacce ricche e immersive. [Per saperne di più](#).
- **Supporto XML per LVGL Pro.** Descrivere l'interfaccia utente in modo dichiarativo e cariarla in fase di esecuzione senza dover ricompilare il firmware. Da provarlo con il nuovissimo editor LVGL Pro. [Per Scoprire di Più <https://pro.lvgl.io/>](#)

13.1.2 Accelerazione GPU

- **Supporto per il rendering GPU EVE.** Scaricare il rendering su chip EVE esterni tramite SPI, liberando la CPU dell'MCU per altre attività. [Per Scoprire di Più](#).
- **Acceleratore hardware ESP PPA.** Si ottengono tempi di rendering più rapidi del 30% e un utilizzo della CPU ridotto del 30% sul nuovo ESP32-P4 con accelerazione hardware nativa. [Per Scoprire di Più](#).
- **NemaGFX Vector Graphic.** Rendering di SVG e grafica vettoriale in modo nativo su NeoChrom di ST e altre GPU compatibili con NemaGFX. [Learn More](#).
- **Unified VGLite Renderer.** Single driver supporting all platforms with a consistent feature set. [Learn More <https://docs.lvgl.io/master/details/integration/chip_vendors/nxp/vg_lite_gpu.html>](https://docs.lvgl.io/master/details/integration/chip_vendors/nxp/vg_lite_gpu.html)
- **Dave2D Renderer improvements.** Lower CPU usage on Renesas and Alif platforms with optimized Dave2D integration. [Learn More <https://docs.lvgl.io/master/details/integration/chip_vendors/renesas/built_in_drivers.html#dave2d>](https://docs.lvgl.io/master/details/integration/chip_vendors/renesas/built_in_drivers.html#dave2d)

13.1.3 MPU Features

- **GStreamer Support.** Play videos and multimedia content directly within LVGL UIs with full GStreamer codec support. [Learn more](#).
- **DRM + EGL Driver.** High-performance Linux display rendering with native DRM and OpenGL ES (EGL) support for embedded displays capable of rendering glTF models. [Learn more](#).

- **ARM NEON Optimization.** Up to 33% faster software rendering with on platforms that support NEON SIMD support. [Learn More](#).

13.1.4 Altre Funzionalità

- **Faster GIF Library.** No changes required, just upgrade LVGL and your GIFs will render faster. [Learn More](#).
- **FrogFS support.** Pack directory trees into a single "blob" and load them at runtime. [Learn More](#).

13.1.5 Documentazione

Alongside these features, we've thoroughly reorganized and proofread the entire documentation suite, and included numerous new driver guides and platform integrations to better serve our partners and the embedded community.

13.1.6 Nuove Funzionalità

- **feat(EVE): allow disabling write buffering** [9072](#)
- **feat(opengl): support glsl version 100 for default shader** [9029](#)
- **feat(subject): add back subject increment events** [9057](#)
- **feat(xml): add scrollbar modes** [9058](#)
- **feat(test): reintroduce lv_text_get_size without attributes** [9036](#)
- **feat(theme): re-initialize the theme on resolution change** [9048](#)
- **feat(g2d): add rgb565, png, tiled images and rotation support** [8848](#)
- **feat(gltf): replace default gltf env image with a more generic solution** [9045](#)
- **feat(drivers): NV3007 driver & docs** [9021](#)
- **feat(sysmon): add cpu proc usage when using monitor log mode** [9010](#)
- **feat(drm): add ability to select display mode** [8992](#)
- **feat(osal): enable linux specific features with os_none** [8989](#)
- **feat(egl/gltf): add gltf support with EGL** [8980](#)
- **feat(draw): support image colorkey** [8628](#)
- **feat(NemaGFX): add vector draw task support** [8938](#)
- **feat(chart): add cursor removal method** [8952](#)
- **feat(lv_global): move user_data to the top of lv_global_t** [8908](#)
- **feat(sdl): Add SDL_Window accessor function to retrieve window from lv_display_t** [8886](#)
- **feat(kconfig_verify): add space indentation check** [8962](#)
- **feat(xml): add spinbox support** [8957](#)
- **feat(EVE): support fonts with a stride alignment** [8937](#)
- **feat(xml): support <include_timeline>** [8902](#)
- **feat(display): deprecate lv_display_set_antialiasing** [8958](#)
- **feat(roller): function to get option string** [8730](#)
- **feat(xml): add subject_toggle_event and move the limits to subjects from increment event** [8930](#)
- **feat(qrcode): add new set_data helper function** [8950](#)
- **feat(draw/sw): add neon support for rgb888 and xrgb8888** [8933](#)
- **feat(XML/fsdrv): add frogfs and XML load from blob** [8870](#)
- **feat(xml): add new step tests click_on and set_language** [8931](#)

- **feat(label): implement api to bind a translation tag to a label** [8948](#)
- **feat(vglite): refactor driver config and remove NXP VGLite draw unit** [8911](#)
- **feat(pxp): add PXP support for tiled images** [8846](#)
- **feat(translation): add language changed event** [8882](#)
- **feat(profiler): add posix porting** [8924](#)
- **feat(EVE): support bitmaps larger than 511x511** [8732](#)
- **feat(XML): XML load from directory** [8774](#)
- **feat(wayland): remove deprecated WL_SHELL** [8877](#)
- **feat(api_mapping): add ability to disable api mapping** [8745](#)
- **feat(xml): add qrcode support in XML** [8904](#)
- **feat(draw/dave2d): remove todos** [8896](#)
- **feat(opengl): add support for creating an LVGL OpenGL texture display from an existing texture** [8837](#)
- **feat(grad): make complex gradients available regardless of whether draw_sw support is enabled** [8834](#)
- **feat(event): add ability to delete display and indev on event** [8893](#)
- **feat(draw): add event_cb in draw unit** [8891](#)
- **feat(sdl): add set_window_size** [8809](#)
- **feat(xml): add rotation in arc xml parser** [8899](#)
- **feat(style): add `lv_style_merge` and appropriate tests** [8680](#)
- **feat(wayland): Restore the touchscreen support** [8785](#)
- **feat(gstreamer): add gstreamer support** [8757](#)
- **feat(gltf): allow compiling gltf with EGL** [8833](#)
- **feat(tests): add lv_deinit test** [8789](#)
- **feat(draw_buf): add MVE & NEON support of alpha premultiply** [8712](#)
- **feat(OpenGL): add EGL support** [8677](#)
- **feat(osal): add lv_sleep_ms interface and implementation.** [8805](#)
- **feat(display): add ability to get current screen being loaded** [8746](#)
- **feat(display): initial loyan integration** [8630](#)
- **feat(draw): vg_lite: add initial support for the VGLite HAL** [8795](#)
- **feat(opengles): add performance measurement points** [8776](#)
- **feat(NemaGFX): Support complex gradients: linear, radial** [8696](#)
- **feat(tests): add lcd driver test** [8759](#)
- **feat(wayland/dmabuf): add window decorations support** [8406](#)
- **feat(draw/dave2d): improve performance using block processing** [8725](#)
- **feat(vglite): add vglite kernel driver** [8734](#)
- **feat(gltf): add gltf support** [8643](#)
- **feat(workflows): add gcover report** [8750](#)
- **feat(EVE): add asset pre-upload functions** [8710](#)
- **feat(draw_buf): make buf_copy overwritable for GPU** [8693](#)
- **feat(draw/sw): add neon 64bit support for rgb565 blending operations** [8590](#)

- **feat(font_manager): check resource leak before removing source** 8613
- **feat(evdev): allow adopting existing fd** 8556
- **feat(chart): add stacked chart support** 8573
- **feat(xml): add animation support** 8599
- **feat(nuttx_profiler): support trace to file** 8674
- **feat(vg_lite): added compatibility with NXP platforms** 8642
- **feat(EVE): Add EVE draw unit** 8211
- **feat(scripts): add display & drawbuf dump** 8645
- **feat(sysmon): add API to start or stop sysmon and dump FPS info** 8533
- **feat(obj): allow `LV_SIZE_CONTENT` for min and max width/height** 8448
- **feat(NemaGFX): encoded images** 8523
- **feat(cache): implement dynamic array with second chance algorithm class** 8433
- **feat(font): Remove SimSun Font** 8550
- **feat(svg): add SVG2 special path command 'A' support.** 8574
- **feat(label): support passing variadic list as param** 8322
- **feat(demo): separate the widgets demo into multiple files** 8507
- **feat(xml): add lv_xml_set_default_asset_path** 8549
- **feat(xml): add float support to subject change events** 8535
- **feat(nuttx) : add mouse capabilities to applications** 8425
- **feat(state): Add state trickle down** 8499
- **feat(events): add event trickle mechanism to propagate events to children** 8415
- **feat(chart): support custom LV_EVENT_REFR_EXT_DRAW_SIZE** 8268
- **feat(osal/linux): add function which provides cpu usage of the app itself** 8340
- **feat(dma2d): enable the DMA2D clock for STM32L4 series** 8429
- **feat(anim): call start callback when animation restarts** 8403
- **feat(DMA2D): encoded images** 8470
- **feat(translation): add translation support** 8255
- **feat(xml): add testing support for XMLs** 8327
- **feat(observer): add float subject type** 8388
- **feat(dma2d): improve DMA2D Compatibility** 8293
- **feat(assert): refactoring LV_ASSERT_HANDLER handling** 8434
- **feat(anim): add vsync mode** 8087
- **feat(drivers): add NXP ELCDIF initial display support** 8349
- **feat(draw/ppa): add initial Pixel Processing Accelerator infrastructure for ESP** 8270
- **feat(vg_lite): add tile image support** 8226
- **feat(draw): Optimize struct alignment** 8409
- **feat(vg_lite): support tiled image** 8355
- **feat(tiny_ttf): add kerning cache** 8320
- **feat(xml): make <style> and <bind> properties child tags instead of props** 02b079f

- **feat(observer): add subject set and increment functions** [3ecd37e](#)
- **feat(arc_label): add new widget arc label support** [5f28038](#)
- **feat(obj): add screen_create/load_event API and XML support** [a762e20](#)
- **feat(observer): add lv_obj_bind_style** [104ede5](#)
- **feat(xml): add remove_style and remove_style_all support** [3951ea4](#)
- **feat(arc_label): support recolor feature** [78e3375](#)
- **feat(scale): add lv_scale_bind_section_min/max_value** [7aaeb66](#)
- **feat(span): add lv_spangroup_bind_span_text and lv_spangroup_set_span_text_fmt** [03c4870](#)
- **feat(xml): add lv_switch** [250bc8b](#)
- **feat(scripts): add fully automatic test + environment installation verification script** [6fbdb7f](#)
- **feat(arc_label): support vertical align feature** [cb22319](#)
- **feat(examples): add arc label example** [5e8add3](#)
- **feat(draw): Use Espressif PPA driver in non-blocking mode, also tune the DMA2D burst size for performance on PPA side** [f3e0056](#)
- **feat(image): add data binding to image src** [d33320b](#)
- **feat(xml): add grid support** [92a51f0](#)
- **feat(bar): add data binding to the bar** [9bddc81](#)
- **feat(arc_label): support horizontal align feature** [bcfc2ab](#)
- **feat(qrcode): support qrcode quiet zone feature** [0b71f7d](#)
- **feat(tabview): support getting tab buttons by index** [fbc8dbe](#)
- **feat(xml): add support for scroll snapping** [40158e4](#)
- **feat(arc_label): support text offset** [fc83651](#)
- **feat(build.py): add documentation section to quickly orient new users on where data flows.** [e818acf](#)
- **feat(xml): qrcode support quiet_zone property** [98423e6](#)
- **feat(arc_label): support text letter space** [33bb46f](#)
- **feat(arc_label): support text direction of counter clockwise** [f55fee8](#)
- **feat(arc_label): support more calculate accuracy** [1dc268e](#)
- **feat(custom.css): add `lvgl-figure-caption` class back...** [ebcd244](#)
- **feat(arc_label): support radius percentage mode** [7cdc1ef](#)
- **feat(xml): review the types used for style properties** [96e4f8f](#)
- **feat: make enum type names consistent with value prefixes** [82cd8ad](#)
- **feat: remove line from `requirements.txt` that is not actually required to build docs** [452878f](#)

13.1.7 Performance

- **perf(EVE): buffered writes** [9051](#)
- **perf(egl): improve egl performance** [8966](#)
- **perf(GIF): Faster GIF library drop-in replacement** [8743](#)
- **perf(tests): remove duplicate vg-lite compilation** [8841](#)
- **perf(test_perf): increase the refresh rate to 60 FPS** [8751](#)
- **perf(vg_lite): optimize GPU fill rendering performance** [8609](#)

- **perf(vg_lite):** optimize the memory usage and search speed of grad item 8474
- **perf(snapshot):** render from the top object when taking snapshot. 8424
- **perf(text):** reduce the get glyph function call 8364
- **perf(draw):** only dispatch refreshing disp 8305
- **perf(arc_label):** optimization enhancement of arc label algorithm b49bba5

13.1.8 Fixes

- **fix(windows):** correctly calculate Windows window-size 9060
- **fix(g2d):** properly check and free an item from the map 9065
- **fix(DMA2D):** fix non-async mode build failure 9069
- **fix(vg_lite_driver):** general fixes at the vglite driver and vglite draw unit link 9063
- **fix(label):** ignore empty translation tags 9062
- **fix(examples):** fix format specifier warning 9066
- **fix(g2d):** fix compilation error with G2D version <2.3.0 9061
- **fix(chart):** fix division by zero if div. line count is 1 9064
- **fix(dma2d):** enable dma2d clock for STM32N6 9044
- **fix(indev):** fix off by one error resulting in elastic scroll when disabled 9047
- **fix(anim):** play animations from their current progress 9052
- **fix(draw_sw):** sw vector graphic partial mode fix y_offset 9006
- **fix(wayland):** correctly get driver data in lv_wayland_assign_physical_display 9017
- **fix(draw/neon):** disable function using an invalid intrinsic on arm32 9019
- **fix(OpenGL/DRM):** fix misc info logs 9027
- **fix(draw/vglite):** update the wait interrupt function to use the lv_sync_wait instead of the FreeRTOS Semaphore specfc API 9007
- **fix(draw):** fix render issue when triangle overhangs by sorting points 9002
- **fix(table):** remove free calls to user data 8881
- **fix(wayland):** bind driver data to display's driver data instead of user data 8991
- **fix(wayland):** multiple windows fix and wl_output support 8923
- **fix(assert):** revert asserting in `lv_assert_handler` function 8976
- **fix(file_explorer):** fix mem leak when sorting by file kind 8935
- **fix(themes):** add malloc assert checks 8977
- **fix(drm/egl):** correctly match egl config in config select cb 8978
- **fix(sdl):** support UTF-8 characters in the keyboard 8906
- **fix(xml):** add the missing parsing of min/max subject limits 8971
- **fix(wayland):** bind draw buffers to the window instead of the shm_context 8878
- **fix(draw_buf):** check range for draw buffer when accessing pixel 8907
- **fix(opengles):** add missing variable declaration 8969
- **fix(menu):** larger bitfield for enum value needed on Windows 8965
- **fix(gdb/draw_buf):** validate and correct expected data size for pixel buffer 8944
- **fix(EVE):** align the dummy buf to 4 bytes 8953

- fix(vg_lite): change log level from info to warning for unsupported gradient features 8947
- fix(fbdev): fix stride, bounds checking, and partial flushes in direct render mode in flush callback 8914
- fix(vg_lite): replace FLT_MIN to -FLT_MAX 8940
- fix(matrix): fix matrix transformation area coordinate error 8928
- fix(vector): rename and enhance rectangle path function with xywh parameters 8934
- fix(msgbox): remove superfluous dependency check 8916
- fix(sysmon): division by zero error 8918
- fix(flex): fix alignment issue when using rtl in flex layout 8929
- fix(vector): adjust image stride handling in fill pattern and add test case 8815
- fix(vg_lite): fix build break after LV_USE_VECTOR_GRAPHIC is turned off 8912
- fix(keyboard): add missing dependency checks 8917
- fix(wayland): fix stride calculation when iterating over drawbuf 8915
- fix(kconfig): add TinyTTF cache count setting support 8919
- fix(memfs): use string only in lv_fs_path_ex_t for better compatibility 8571
- fix(os): make lv_lock/unlock public as they should be used by the users 8889
- fix(menu): allow using menu section as parent of menu container 8884
- fix(xml): fix grid support 8888
- fix(font): fix stride calculation of fonts 8887
- fix(label): make long dot mode compatible with static text 8823
- fix(opengles): fix memory leak when shader compilation fails 8854
- fix(sysmon): handle LV_SYSMON_PROC_IDLE_AVAILABLE disabled in lv_linux 8871
- fix(profiler): remove macro for external usage 8894
- fix(span): opt span ellipsis more precise in omitting text 8864
- fix(file_explorer): ensure current_path is null-terminated after appending '/' 8797
- fix(gdb): use stride to get actual buffer width for display dump 8853
- fix(draw/dave2d): prevent deadlock by avoid to flush the GPU buffer when empty. 8869
- fix(fbdev): fix segfault when using display rotation 8850
- fix(dmabuf): flicker when the buffers are swapped for new frame 8563
- fix(label): issue when setting fmt text with long mode dots 8781
- fix(vglite): arc_path calculation overflow on VGLite 8847
- fix(draw/dave2d): redefine fixed point macros 8859
- fix(xml): fix multiple delimiter handling in lv_xml_split_str 8858
- fix(svg): fix stroke dash style not restore bug. 8844
- fix(draw/sw): check that malloc doesn't return a NULL pointer 8756
- fix(wayland): throw error when using dmabuf without 2 draw buffers 8744
- fix(span): fix text width calculations and improve ellipsis handling 8821
- fix(tjpgd): Fixed memory leak if opening JPEG image failed. 8707
- fix(sdl): The default size is fixed 8808
- fix(opengl): fix multiple issues with shader manipulation 8832

- fix(draw/convert): fix duplicate c declaration symbol during docs build 8827
- fix(indev_gesture): fix memory leak when deleting indev 8814
- fix(cache): make the cache module private 8793
- fix(gltf): add missing initialization call for viewer attribute 8807
- fix(label): preserve recolor state across line wrapping 8755
- fix(draw/dma2d): fix freezing in async mode. 8765
- fix(draw/pxp): fix alignment issue on cache flush. 8764
- fix(profiler): fix crash during lv_deinit 8788
- fix(freetype): fix crash during lv_deinit 8787
- fix(lodepng): check that src is not null before calling memcpy 8791
- fix(jpeg): fix to use source stride when copy jpeg 8778
- fix(style): format error in log call 8754
- fix(test_span): remove the reference before deleting the font 8753
- fix(event): don't try to trickle to deleted widgets 8777
- fix(vg_lite): fix rotated vector font clipping error 8739
- fix(EVE): rounded arcs with transparency 8767
- fix(image): reset img when src_type is unknow or info is wrong 8716
- fix(image_decoder): fix memory leaks caused by multi-core rendering 8735
- fix(image): fix unused var warning when LV_LOG_WARN isn't available 8591
- fix(tests): fix gcovr report generation failure 8727
- fix(canvas): add the missing D-Cache flush 8736
- fix(test_cache): fix access to freed stack variables 8717
- fix(draw_span): add missing line_space attribute to dsc and remaining_len check 8718
- fix(drivers): fix compile error for st ltdc 8704
- fix(scale): initialize label size if text is NULL 8699
- fix(evdev): add support for multi-touch 8173
- fix(SVG): fix SVG decoder memory leak. close decoder 8658
- fix(eve): fix image double ended clipping in EVE 8675
- fix(xmls): fixed division lines in Chart XML API 8681
- fix(EVE): fix EVE build break 8672
- fix(vg_lite): remove the 16px width limit for image tile drawing mode 8661
- fix(vg_lite_tvg): fix matrix calculation inconsistency with hardware behavior 8666
- fix(vg_lite): always use phy_clip_area as the scissor area 8669
- fix(indev): always assign data->point to the primary point instead of... 8596
- fix(draw):fix lv_matrix_transform_point 8667
- fix(demos/widgets): add missing aligned font resource declarations 8647
- fix(vg_lite): fix incorrect handling of ARGB8888_PREMULTIPLIED 8610
- fix(vg_lite_tvg): fix incorrect stride handling 8648
- fix(textarea): fill letter space with selection bg color. 8238

- fix(msgbox): fix memory leak and correct LV_ASSERT_MALLOC parameters 8624
- fix(textarea): label scroll to the cursor pos when textarea size change 8548
- fix(scripts): Union syntax compatible with Python 3.8 8641
- fix(font loader): overflow error when calculating ids size 8617
- fix(svg): cast char to unsigned preventing char subscription errors 8619
- fix(example/label): make animation repeat indefinitely 8621
- fix(tabview): emit LV_EVENT_VALUE_CHANGED on header-button taps 8537
- fix.dropdown): support aligning the dropdown button's text 8601
- fix(mem): change variable attribute location 8593
- fix(grid): fix uninitialized var warnings 8592
- fix.dropdown): check for NULL in dropdown label before dereferencing 8543
- fix(xml): set correct param type for subject_set_string_event 8595
- fix(xml): add missing type for lv_arc start/end angles 8580
- fix(xml): specify element type="void" for event/screen events 8577
- fix(xml): fix incorrect prop names in lv_arc 8562
- fix(label): fix label clip overflow 8557
- fix(NemaGFX): non-grad triangle coords 8524
- fix(draw_vector): ensure vector_graphic can be used if it's enabled 8522
- fix(svg): add log for unsupport path command. 8545
- fix(xml): add missing subject_set_float_event to the schema 8538
- fix(lv_example_bar_7): make bar fit in 320x240 box for webpage 8525
- fix(xml): fix scale section props 8528
- fix(draw/g2d): use image stride if it's set 8503
- fix(snapshot): not all children rendered during snapshot 8479
- fix(indev): set the timestamp only if it was not set in the read_cb 8501
- fix(wayland): call display_flush_ready on intermediate frames 8471
- fix(examples): wrap logging code with LV_USE_LOG to avoid unused variable warnings 8377
- fix(g2d): calculate stride because it may not be set in the image descriptor 8483
- fix: include v9.2 api map to avoid breaking changes 8490
- fix(vg_lite): fix grad item not fully initialized 8486
- fix(draw/sw_mask): fix potential buffer overflow 8478
- fix(lv_deinit): only call xml test call when both LV_USE_XML and LV_USE_TEST are enabled 8482
- fix(indev): decouple scroll momentum decay from indev read loop (#6832) 7794
- fix(draw_sw): fix ARGB8888PM case where the color of dest and src is the same but opa is different. 8454
- fix(wayland): Avoid premultiplication at the driver level if LVGL is already rendering in pre-multiplied alpha. 8456
- fix.dropdown): handle the case where options are null lv_dropdown_add_option 8338
- fix(xml): handle LV_STATE_USER_1-4 8451
- fix(assert): fix lv_assert_handler function static analyze complain 8447

- fix(blend/helium): eliminate duplicate function declarations 8402
- fix(draw): Fix unused function warning for blending 8414
- fix(bin_decoder): don't flush cache in case of `use_directly` 8411
- fix(svg): fix some svg issues. 8378
- fix(span): fix correct text length calculation 8368
- fix(uefi): include configuration header instead of full LVGL header 8380
- fix(sdl): add malloc check after reallocating memory 8369
- fix(draw_vector): fix the unprocessed layer opa in vector rendering 8294
- fix(kconfig): Fix depends clauses for LV_USE_OS 8392
- fix(thorvg): fix compilation on thread-less platforms 8373
- fix(thorvg): include alloca on zephyr 8381
- fix(xml): fix screen names and extended component names b441ec3
- fix(custom.js): fix to correctly deal with when... 45e4041
- fix(xml): make all obj elements type=void 8de083e
- fix(xml): minor fixes in parsers dea37bd
- fix(custom.js): fix examples code-folding issue... 706715f
- fix(xml): find constants in styles in the global scope 84acb67
- fix(arc_label): fix the pointer overflow issue in 32-bit mode 059eb4a
- fix(grid): ignore grids without row or column descriptors 38000e8
- fix(xml): process 'align' style property ec0e3cf
- fix(xml): fix prop order of subject_set_* f7dc114
- fix(xml): fix wrong tag names for subject events 8599e9b
- fix(xml_animation): use lv_obj_find_child_by_name instead of lv_obj_get_child_by_name 876dacb
- fix(scripts/prerequisites-apt): corrected freetype version number for verification dccab75
- fix(properties.py): fix to comply with adjustments 3a4cf38
- fix: add `brotli` module required for `fonttools` to... 4c6624b
- fix(obj xml): fix typo in style_shadow_width 42a391d
- fix(textarea xml): rename placeholder to placeholder_text 3f41172
- fix(xml): fix event_cb parser typo (trigger_str -> trigger) 244f9cc
- fix(scripts/prerequisites-apt): add missing libsdl2-dev:i386 installation c0d08d8
- fix(keyboard xml): add number mode 248192f
- fix(xml): add missing lv_xml_widget.h include 6ff3cee
- fix(xml): fix better handling broken components 592df6b
- fix(docs): fix docs build for arclabel example building f40e0e5

13.1.9 Esempi

- examples(scale): Compass style rotating scale 8445

13.1.10 Docs

- docs(xml): fix the descripton of how to initialize the UI from C 9070
- docs(README): remove outdated content and add new features 9023
- docs(xml): Fix array type description count to 3 (#9013) 9028
- docs: update the esp docs indentation at the FS section. 9031
- docs(editor): improve installation and figma docs 8998
- docs(drivers): fix broken links 9022
- docs(espressif): fix indentation at file system reconfigure project. 9024
- docs(xml/events): rename page to avoid confusion when searching documentation 8995
- docs(flags): add chapter explaining how to add/remove multiple flags 9000
- docs(events): add explanation about lv_event_get_target_obj warning and tips 8996
- docs(xml): add missing licence files 8984
- docs(editor): add missing intro pages and minor tweaks 8983
- docs: split indev, style, and widget basics to multiple pages 8943
- docs(editor): add XML and C integration guides and minor refactoring 8954
- docs(xml): add help texts in XMLs 8961
- docs(pc): add the missing PC simulator docs 8824
- docs(editor): add cli docs 8970
- docs(snapshot): complementary color format 8780
- docs(xml): add online share docs 8973
- docs(editor): add docs specific to editor 8972
- docs(style): update font sizes and weights for improved readability 8968
- docs(events): fix grammatical issue 8782
- docs (frameworks): PlatformIO 8942
- docs(xml): add placeholder for editor docs 8910
- docs(misc): fix doc issues described in issue #8839 8857
- docs(timer/indev): add example section about indev interrupts resuming the indev timer 8843
- docs(opengl): add opengl overview page 8838
- docs(xml): reorganize and add better descriptions 8818
- docs(espressif): update esp32 documentation 8830
- docs(Kconfig): clarify comments about driver-identifier letter 8851
- docs(misc): fix multiple formatting errors and typos 8738
- docs(misc): fix formatting errors, typos and missing reference 8748
- docs(style): fix arrows in code blocks 8819
- docs: fix indev.rst typo 8721
- docs(xml): next set of XML-doc refinements 8740
- docs(STM32): simplify and rewrite 8768
- docs(mipi): align return type of callbacks in documentation with code 8803
- docs(integration): reorganize the files 8701

- **docs(wayland): use correct window create function name** 8731
- **docs: miscellaneous improvements** 8552
- **docs(image.rst): remove reference to True Color which was removed in v9.0** 8700
- **docs(misc): fix several issues:** 8714
- **docs(keyboard): fix typos and unify comments** 8391
- **docs(misc): fix several issues** 8698
- **docs(api_doc_builder.py): fix frail assumptions in API-page generation** 8694
- **docs(xml): results of minor amount of proofreading** 8604
- **docs(drivers): Update st7796.rst URL** 8687
- **docs(image): document the fact that indexed data is stored as argb8888** 8620
- **docs: update docs styling** 8585
- **docs(custom.css): fix text color on view-on-github example buttons** 8652
- **docs(custom.css): Fix field list items being jammed together** 8639
- **docs: remove outdated references to LV_ZOOM_IMAGE_NONE** 8632
- **docs(LTDC): Clarify LTDC framebuffer address meaning** 8615
- **docs(list formatting): fix erroneous list formatting** 8602
- **docs(readme): fix broken links** 8605
- **docs(style): fix definition-list indentation** 8616
- **docs(intro): improve initial presentation of LVGL...** 8597
- **docs(renesas): mention relevant information about FSP 6.0** 8572
- **docs: fix wording in display event, viewe and renesas** 8546
- **docs(home): fix links** 8560
- **docs(intro): reorganize and simplify** 8389
- **docs: remove outdated doc material** 8520
- **docs(xml): restructure and extend the XML docs** 8495
- **docs(Renesas): Add Renesas RA8P1 docs** 8509
- **docs(readme): add Hebrew language section to documentation** 8345
- **docs(wayland): add missing documentation about its config options** 8476
- **docs: fix spacing between same-level sections and h4 and link colors** 8443
- **docs(lv_conf): fix links** 8462
- **docs(1-level TOC styling): multiple fixes...** 8431
- **docs: Fix several typos and punctuation** 8427
- **docs(calendar): fix typo** 8413
- **docs(boards): Add Viewe Knob Display links** 8461
- **docs(doc-build warnings): fix warnings caused by LV_FORMAT_ATTRIBUTE** 8384
- **docs(xml): add license for the XML format** 8438
- **docs(observer): fix typo in parameter description** 8372
- **docs(custom.css): partially address code-highlighting problem...** 8374
- **docs(CHANGELOG.rst): fix link to SVG support page** 8356

- **docs(CHANGELOG.rst)** replace `kisvegabor/lvgl_upstream` with... [9cf1e38](#)
- **docs:** generate changelog [05d8b9e](#)
- **docs:** Lot of smaller improvements based on feedback [2b9592e](#)
- **docs(custom.js):** The custom.js file now contains the version selector controller JavaScript and other methods that were previously directly added to the main layout. [8541a11](#)
- **docs(custom.css):** enhance styling and add custom CSS variables [4f843e8](#)
- **docs(data_binding):** move the data binding docs to to the given widget's page [f6f6570](#)
- **docs:** add dynamic announcements from remote .JSON [77cfa87](#)
- **docs(furo):** convert to furo theme... [2129d37](#)
- **docs:** Enhance sidebar, remove jQuery dependency [5b3d6d8](#)
- **docs(misc):** eliminate several Sphinx warnings [b74ac71](#)
- **docs(Sphinx Warnings):** add name to anonymous enums to... [55ab01e](#)
- **docs(Icons.html):** Update SVG icons for improved accessibility and consistency [a13113b](#)
- **docs(arc_label):** add arc label docs [8487b4c](#)
- **docs:** fixed several problems... [4533d29](#)
- **docs(misc):** minor improvements to: [450a390](#)
- **docs(doxygen_xml.py):** exclude files that... [e92ce1f](#)
- **docs(enum-naming):** adjustments per conversation with Gábor [5bc21a4](#)
- **docs(logo):** update dark and light logos [44a369a](#)
- **docs(custom.css):** format field lists [45575c0](#)
- **docs(observer):** fix minor typos [0594775](#)
- **docs(API/index.rst):** eliminate Sphinx warnings when... [0829150](#)
- **docs(data_binding):** fix typos [db71f48](#)
- **docs(CHANGELOG.rst):** fix parsing error on 8 boldface strings [82ff3c2](#)
- **docs(custom.css):** Refactor link color variables for improved theming consistency [f725034](#)
- **docs(observer):** fix subject name in the example [8003908](#)
- **docs(properties.py):** fix several problems... [f3370d0](#)
- **docs(index.rst):** Add `hide-toc` furo theme prop to hide sidebar on the homepage and remove main header image. [af2c420](#)
- **docs(style):** add example values to lv_style_selector_t [6bcdff3c](#)
- **docs(observer):** minor grammar corrections [989ac6c](#)
- **docs:** (custom.css): add caption feature and... [6f26b71](#)
- **docs(build.py):** make Sphinx exit with status 1 when new warnings show up [ef4aa46](#)
- **docs(os/index.rst):** put TOC back in alpha sequence [c994e5c](#)
- **docs(custom.css):** Fix theme toggle button label color [bc4b55c](#)
- **docs(Makefile):** fix typo for Linux sphinx-build after intermediate directory created [79e55d4](#)
- **docs(lv_obj.h):** make enum name consistent with enumerators [3d1b533](#)
- **docs(lv_obj.h):** make enum name consistent with enumerators [ffb29b0](#)
- **docs(lv_svg.h):** fix build of `lv_svg.c` [cb76aaa](#)
- **docs(image.rst):** fix doc-gen warning [70f8c52](#)

- **docs(unneeded): eliminate unneeded docs....** [d2d2acc](#)

13.1.11 CI and tests

- **ci(release_updater): some release_updater fixes and conf.defaults deprecated configs are non-fatal** [9076](#)
- **ci(release_updater): fix 2 port updater URLs** [9074](#)
- **ci(pio): fix the PlatformIO publisher action** [9073](#)
- **ci(docs): fix docs ci by fetching lvgl in default gh workspace folder** [9053](#)
- **ci(docs): fix docs CI. use the action's checked-out repo for emscripten** [9020](#)
- **test(docker): add compatibility to tests' dockerfile for arm platforms** [9015](#)
- **ci(docs): fix examples build error when not rebased fully** [8880](#)
- **ci(test): generate the I1 test images with pngquant 2.17** [8892](#)
- **test(bin_decoder): add lv_bin_decoder test case** [8867](#)
- **ci(stalebot): increase stale and close times to 60 and 14 days** [8840](#)
- **ci(hw/comment): gracefully exit workflow if triggered by another label** [8828](#)
- **test(images): update test images according to LVGLImage script** [8790](#)
- **ci(install): add cmake installation workflow** [8794](#)
- **tests: add LV_DRAW_BUF_STRIDE_ALIGN = 1 test** [8804](#)
- **test: fix syntax warning by using invalid escape sequence '.'** [8792](#)
- **ci(deps): bump actions/checkout from 4 to 5** [8796](#)
- **ci:(port_updater): add more CI port updater urls** [8686](#)
- **tests: rename makefile directory to makefiles.** [8646](#)
- **ci: add hardware performance test workflow** [8567](#)
- **ci: find and edit existing comment for benchmark results** [8531](#)
- **ci: disable benchmark results comment if there's no diff** [8515](#)
- **ci: emulated perf workflow** [7949](#)
- **ci(lv_tmpl): fix filename template mismatches and add CI** [8326](#)
- **test(perf): add perf test framework** [8064](#)
- **test(arc_label): add arc_label test cases** [9cf663e](#)
- **test(qrcode): add qrcode quiet zone test** [bdcb291](#)
- **ci: enable LV_USE_OBJ_NAME as it's required for some XML features** [297aeae](#)
- **ci(hw): add missing fontconfig lib dependency for ejlv** [f5fe4f2](#)
- **ci: correctly pull the latest results in case it's a PR** [ac36b29](#)
- **ci(perf/build/hw): remove checkout step from workflow** [7081b45](#)
- **ci(hw): increase job timeout** [826d167](#)
- **ci(build/hw): disable reluctant hw builds on pushes to master** [f6198e6](#)
- **ci: update lv_perf image version to a more stable version** [29eac60](#)
- **ci(perf/hw): correctly pass github token to command** [0b6cab8](#)
- **ci: generate a release body for the benchmark results pre-release** [ea86afd](#)

13.1.12 Others

- **chore(cmsis-pack) update cmsis-pack for the last PRs of v9.4.0** 9077
- **chore(cmsis-pack): Update for v.9.4.0** 8945
- **chore(conf): remove redundant thorvg definition** 9067
- **chore: fix examples not showing on docs by fixing path to examplelist.c** 9059
- **refactor(style): remove dead code** 9040
- **build(cmake): fix MSVC exports for shared/static libraries on Windows** 9026
- **build(kconfig): add missing argb8888 premultiplied setting** 8997
- **chore(deps): bump actions/setup-python from 5 to 6** 8985
- **chore(deps): bump actions/stale from 9 to 10** 8986
- **chore(deps): bump actions/checkout from 4 to 5** 8987
- **chore(deps): bump actions/github-script from 7 to 8** 8988
- **chore(font): add missing special characters to aligned fonts** 8975
- **chore(xml): add API docs to lv_xml_update_from_data** 8967
- **chore: fix lv_font_montserrat_14_aligned.c include** 8951
- **refactor(opengles): mainline usage of shader manager in opengl driver** 8836
- **refactor(gdb): lv_global supports lazy loading and refactors some structures and formats.** 8939
- **refactor(vector): clean up the API of vector drawing** 8922
- **chore(xml): add guards around widget parsers** 8932
- **refactor(gdb): re-structure lvlgdb module** 8920
- **chore(tests): fix typo in qrcode test** 8900
- **chore(lvconf_gen): generate empty lv_conf.h if it doesn't exist** 8885
- **chore(repo_updater): add missing boards to the repo updater** 8856
- **refactor(sysmon): polish the UI with LV_SYSMON_PROC_IDLE_AVAILABLE** 8519
- **chore(kconfig): add missing LV_OS SDL2 option to LV_USE_OS** 8831
- **refactor(opengl): move opengl shader manager to opengles driver** 8829
- **chore(font): replace arial with liberationsans** 8784
- **scripts: add the frdm-mcxn-947 to the branch updater script** 8752
- **chore: remove multiple semicolons** 8723
- Fixes bug where multiple events with same callback couldn't be removed in single call. 8706
- **chore(deps): bump dawidd6/action-download-artifact from 9 to 11** 8654
- **build(zephyr): remove zephyr env cmake** 8695
- **chore(stale): ping maintainers on stale bot comments** 8683
- **chore(indev): explain why the timer is resumed on press in event driven mode** 8660
- **chore(configopts): add support for LV_*_USE_* defines** 8575
- **refactor(text): use struct to pass params to lv_text_* and lv_font_* functions** 8500
- **chore(examples/scroll): remove unnecessary chinese comments that cause VC compilation failure** 8668
- **Revert "feat(vg_lite): adapt premultiply src over blend mode (#6062)"** 8649
- **chore(lv_tree): Code reorganization for lv_tree.h** 8583

- refactor(`lv_part_t`/`lv_state_t`): refactor to align types with prototypes [8511](#)
- refactor(`draw`): use better names in `lv_draw_task_state_t` and handle queued vs in_progress [8512](#)
- Replace legacy v9.0 call [8633](#)
- build(`cmake`): only set project name if lvgl's cmakelists is root [8588](#)
- chore(`test/perf`): add ability to debug perf tests [8569](#)
- build(`preprocess`): replace `which` with `shutil.which` to provide a cross-platform compatible solution to find `pcpp` [8530](#)
- build(`esp`): fix lvgl components requirements [8527](#)
- chore(`deps`): bump dawidd6/action-download-artifact from 9 to 11 [8513](#)
- chore: use filename to better match mpk and json files [8518](#)
- chore: fix typos again [8295](#)
- chore(`version`): update version numbers to v9.4.0-dev [8498](#)
- refactor(`themes`): Improved consistency between styles [8346](#)
- chore(`xml`): fix mismatches between `lv_obj` elements and their XML schema [8485](#)
- chore(`api_doc_builder`): fix TOC ordering on API pages. [8421](#)
- chore(`NemaGFX`): add NemaPVG lib binaries [8430](#)
- chore(`refr`): advance send `LV_EVENT_RENDER_START` [8457](#)
- chore(`lvglimage`): fix indexed image premultiply error [8367](#)
- chore(`api_doc_builder`): fix regex for unix systems [8342](#)
- chore(`image`): reduce warning level for src being NULL [8394](#)
- refactor(`drm`): rename config allowing us to link with GBM automatically [8203](#)
- refactor(`xml`): use single properties and introduce the prop-paramX syntax [3e61613](#)
- refactor(`observer`): move bindings to the related widgets [26fd3e7](#)
- chore(`arc_label`): rename `arc_label` to `arclabel` [9489759](#)
- initial commit [1906242](#)
- chore(`doc-build`): adjustments after discussion with André. [6d546a8](#)
- chore: minor fixed in XML and observer related codes [b4c9d2e](#)
- chore(`xml`): memzero mallocoed memoried for consitent behavior [584bac6](#)
- chore: minor formatting and optimizations [b946eb6](#)
- refactor(`arc_label`): improve angle calculations and type safety [6cbf36a](#)
- chore(`observer`): add _event suffix to `add_subject_set/increment` [bd9e9d2](#)
- chore(`arc_label`): documentation, formatting, redundant code fixes [cb593b3](#)
- chore(`xml`): make element access="bind" > "custom" for more flexibility [a134a56](#)
- revert: revert examples [272da82](#)
- chore(`doc-build warnings`): struct adjustments to work around Breathe bug [5d7b2e7](#)
- chore: update version [07271d5](#)
- chore(`lv_obj.h`): fix per discussion [6639c31](#)

14.1 lv_api_map_v8.h

Define

```
LV_DISP_ROTATION_0  
LV_DISP_ROTATION_90  
LV_DISP_ROTATION_180  
LV_DISP_ROTATION_270  
LV_DISP_RENDER_MODE_PARTIAL  
LV_DISP_RENDER_MODE_DIRECT  
LV_DISP_RENDER_MODE_FULL  
LV_BTNMATRIX_BTN_NONE  
LV_BTNMATRIX_CTRL_HIDDEN  
LV_BTNMATRIX_CTRL_NO_REPEAT  
LV_BTNMATRIX_CTRL_DISABLED  
LV_BTNMATRIX_CTRL_CHECKABLE
```

`LV_BTNMATRIX_CTRL_CHECKED`

`LV_BTNMATRIX_CTRL_CLICK_TRIG`

`LV_BTNMATRIX_CTRL_POPOVER`

`LV_BTNMATRIX_CTRL_CUSTOM_1`

`LV_BTNMATRIX_CTRL_CUSTOM_2`

`LV_IMGBTN_STATE_RELEASED`

`LV_IMGBTN_STATE_PRESSED`

`LV_IMGBTN_STATE_DISABLED`

`LV_IMGBTN_STATE_CHECKED_RELEASED`

`LV_IMGBTN_STATE_CHECKED_PRESSED`

`LV_IMGBTN_STATE_CHECKED_DISABLED`

`LV_RES_OK`

`LV_RES_INV`

`LV_INDEV_STATE_PR`

`LV_INDEV_STATE_REL`

`lv_obj_del`

`lv_obj_del_async`

`lv_obj_clear_flag`

`lv_obj_clear_state`

`lv_indev_set_disp`

`lv_indev_get_act`

`lv_scr_act`

`lv_disp_remove`

```
lv_disp_set_default  
  
lv_disp_get_default  
  
lv_disp_get_next  
  
lv_disp_set_rotation  
  
lv_disp_get_hor_res  
  
lv_disp_get_ver_res  
  
lv_disp_get_physical_hor_res  
  
lv_disp_get_physical_ver_res  
  
lv_disp_get_offset_x  
  
lv_disp_get_offset_y  
  
lv_disp_get_rotation  
  
lv_disp_get_dpi  
  
lv_disp_get_antialiasing  
  
lv_disp_flush_ready  
  
lv_disp_flush_is_last  
  
lv_disp_get_scr_act  
  
lv_disp_get_scr_prev  
  
lv_disp_load_scr  
  
lv_scr_load  
  
lv_scr_load_anim  
  
lv_disp_get_layer_top  
  
lv_disp_get_layer_sys  
  
lv_disp_send_event
```

```
lv_disp_set_theme  
  
lv_disp_get_theme  
  
lv_disp_get_inactive_time  
  
lv_disp_trig_activity  
  
lv_disp_enable_invalidation  
  
lv_disp_is_invalidation_enabled  
  
lv_disp_refr_timer  
  
lv_disp_get_refr_timer  
  
lv_timer_del  
  
lv_anim_del  
  
lv_anim_del_all  
  
lv_anim_set_ready_cb  
  
lv_group_del  
  
lv_txt_get_size  
  
lv_txt_get_width  
  
lv_img_create  
  
lv_img_set_src  
  
lv_img_set_offset_x  
  
lv_img_set_offset_y  
  
lv_img_set_angle  
  
lv_img_set_pivot  
  
lv_img_set_zoom  
  
lv_img_set_antialias
```

```
lv_img_get_src  
  
lv_img_get_offset_x  
  
lv_img_get_offset_y  
  
lv_img_get_angle  
  
lv_img_get_pivot  
  
lv_img_get_zoom  
  
lv_img_get_antialias  
  
lv_imgbtn_create  
  
lv_imgbtn_set_src  
  
lv_imgbtn_set_state  
  
lv_imgbtn_get_src_left  
  
lv_imgbtn_get_src_middle  
  
lv_imgbtn_get_src_right  
  
lv_list_set_btn_text  
  
lv_list_get_btn_text  
  
lv_list_add_btn  
  
lv_btn_create  
  
lv_btndmatrix_create  
  
lv_btndmatrix_set_map  
  
lv_btndmatrix_set_ctrl_map  
  
lv_btndmatrix_set_selected_btn  
  
lv_btndmatrix_set_btn_ctrl  
  
lv_btndmatrix_clear_btn_ctrl
```

```
lv_btnmatrix_set_btn_ctrl_all  
  
lv_btnmatrix_clear_btn_ctrl_all  
  
lv_btnmatrix_set_btn_width  
  
lv_btnmatrix_set_one_checked  
  
lv_btnmatrix_get_map  
  
lv_btnmatrix_get_selected_btn  
  
lv_btnmatrix_get_btn_text  
  
lv_btnmatrix_has_button_ctrl  
  
lv_btnmatrix_get_one_checked  
  
lv_tabview_get_tab_btns  
  
lv_tabview_get_tab_act  
  
lv_tabview_set_act  
  
lv_tileview_get_tile_act  
  
lv_obj_set_tile_id  
  
lv_obj_set_tile  
  
lv_roller_set_visible_row_cnt  
  
lv_roller_get_option_cnt  
  
lv_table_set_col_cnt  
  
lv_table_set_row_cnt  
  
lv_table_get_col_cnt  
  
lv_table_get_row_cnt  
  
lv_table_set_col_width  
  
lv_table_get_col_width
```

`lv_dropdown_get_option_cnt`

`lv_obj_get_child_cnt`

`lv_obj_get_disp`

`lv_obj_delete_anim_ready_cb`

`LV_STYLE_ANIM_TIME`

`LV_STYLE_IMG_OPA`

`LV_STYLE_IMG_RECOLOR`

`LV_STYLE_IMG_RECOLOR_OPA`

`LV_STYLE_SHADOW_OFS_X`

`LV_STYLE_SHADOW_OFS_Y`

`LV_STYLE_TRANSFORM_ANGLE`

`lv_obj_get_style_anim_time`

`lv_obj_get_style_img_opa`

`lv_obj_get_style_img_recolor`

`lv_obj_get_style_img_recolor_filtered`

`lv_obj_get_style_img_recolor_opa`

`lv_obj_get_style_shadow_ofs_x`

`lv_obj_get_style_shadow_ofs_y`

`lv_obj_get_style_transform_angle`

`lv_obj_get_style_bg_img_src`

`lv_obj_get_style_bg_img_recolor`

`lv_obj_get_style_bg_img_recolor_opa`

`lv_obj_set_style_anim_time`

```
lv_obj_set_style_img_opa  
  
lv_obj_set_style_img_recolor  
  
lv_obj_set_style_img_recolor_opa  
  
lv_obj_set_style_shadow_ofs_x  
  
lv_obj_set_style_shadow_ofs_y  
  
lv_obj_set_style_transform_zoom  
  
lv_obj_set_style_transform_angle  
  
lv_obj_set_style_bg_img_src  
  
lv_obj_set_style_bg_img_recolor  
  
lv_obj_set_style_bg_img_recolor_opa  
  
lv_style_set_anim_time  
  
lv_style_set_img_opa  
  
lv_style_set_img_recolor  
  
lv_style_set_img_recolor_opa  
  
lv_style_set_shadow_ofs_x  
  
lv_style_set_shadow_ofs_y  
  
lv_style_set_transform_angle  
  
lv_style_set_transform_zoom  
  
lv_style_set_bg_img_src  
  
lv_style_set_bg_img_recolor  
  
lv_style_set_bg_img_recolor_opa  
  
lv_keyboard_get_selected_btn  
  
lv_keyboard_get_btn_text
```

LV_ZOOM_NONE

lv_image_decoder_built_in_open

lv_image_decoder_built_in_close

LV_IMG_DECLARE(var_name)

Use this macro to declare an image in a C file

Typedef

typedef int32_t **lv_coord_t**

typedef lv_result_t **lv_res_t**

typedef *lv_image_dsc_t* **lv_img_dsc_t**

typedef lv_display_t **lv_disp_t**

typedef *lv_display_rotation_t* **lv_disp_rotation_t**

typedef *lv_display_render_mode_t* **lv_disp_render_t**

typedef *lv_anim_completed_cb_t* **lv_anim_ready_cb_t**

typedef *lv_screen_load_anim_t* **lv_scr_load_anim_t**

typedef *lv_buttonmatrix_ctrl_t* **lv_btnmatrix_ctrl_t**

Funzioni

static inline uint32_t **lv_task_handler**(void)

static inline void **lv_obj_move_foreground**(lv_obj_t *obj)

Move the object to the foreground. It will look like if it was created as the last child of its parent. It also means it can cover any of the siblings.

Parametri

obj -- pointer to an object

static inline void **lv_obj_move_background**(lv_obj_t *obj)

Move the object to the background. It will look like if it was created as the first child of its parent. It also means any of the siblings can cover the object.

Parametri

obj -- pointer to an object

14.2 lv_api_map_v9_0.h

Define

lv_image_set_align

```
lv_image_get_align  
  
lv_button_bind_checked  
  
LV_DRAW_BUF_DEFINE  
  
_lv_utils_bsearch  
  
lv_draw_buf_align_user  
  
lv_draw_buf_create_user  
  
lv_draw_buf_width_to_stride_user  
  
lv_draw_buf_dup_user  
  
lv_draw_buf_invalidate_cache_user(handlers, drawbuf, area)  
lv_draw_buf_flush_cache_user(handlers, drawbuf, area)  
lv_draw_buf_destroy_user(handlers, drawbuf)
```

14.3 lv_api_map_v9_1.h

Define

```
_LV_EVENT_LAST  
  
_lv_event_code_t  
  
_lv_event_mark_deleted  
  
lv_obj_add_event  
  
_LV_STYLE_LAST_BUILT_IN_PROP  
  
_LV_FLEX_REVERSE  
  
_LV_FLEX_WRAP  
  
_LV_FLEX_COLUMN  
  
_lv_area_is_equal  
  
_lv_area_is_in  
  
_lv_area_intersect
```

```
_lv_area_is_point_on  
  
_lv_area_join  
  
_lv_image_buf_get_transformed_area  
  
_lv_ll_init  
  
_lv_ll_ins_head  
  
_lv_ll_ins_prev  
  
_lv_ll_ins_tail  
  
_lv_ll_get_head  
  
_lv_ll_get_tail  
  
_lv_ll_get_next  
  
_lv_ll_get_prev  
  
_lv_ll_get_len  
  
_lv_ll_move_before  
  
_lv_ll_is_empty  
  
_lv_ll_clear  
  
_lv_ll_remove  
  
_lv_ll_chg_list  
  
_LV_LL_READ  
  
_LV_LL_READ_BACK  
  
_lv_obj_scroll_by_raw  
  
_lv_obj_get_ext_draw_size  
  
_lv_indev_scroll_handler  
  
_lv_display_refr_timer
```

```
_lv_disp_refr_timer  
  
_lv_disp_get_refr_timer  
  
lv_obj_get_child_by_id  
  
lv_obj_update_flag  
  
_lv_inv_area  
  
lv_chart_set_all_value  
  
lv_calendar_set_showed_date  
  
lv_chart_set_range  
  
lv_chart_set_value_by_id  
  
lv_chart_get_x_array  
  
lv_chart_get_y_array  
  
lv_chart_set_ext_x_array  
  
lv_chart_set_ext_y_array  
  
LV_LABEL_LONG_WRAP  
  
LV_LABEL_LONG_DOT  
  
LV_LABEL_LONG_SCROLL  
  
LV_LABEL_LONG_SCROLL_CIRCULAR  
  
LV_LABEL_LONG_CLIP  
  
lv_anim_set_time  
  
lv_anim_set_playback_time  
  
lv_anim_set_playback_delay  
  
lv_anim_set_playback_duration  
  
lv_gradient_init_stops
```

```
lv_gradient_stop_t  
  
lv_spangroup_new_span  
  
lv_spangroup_refr_mode  
  
lv_slider_set_left_value  
  
lv_calendar_header_arrow_create  
  
lv_calendar_header_dropdown_create
```

14.4 lv_api_map_v9_2.h

Define

```
LV_SCR_LOAD_ANIM_NONE  
  
LV_SCR_LOAD_ANIM_OVER_LEFT  
  
LV_SCR_LOAD_ANIM_OVER_RIGHT  
  
LV_SCR_LOAD_ANIM_OVER_TOP  
  
LV_SCR_LOAD_ANIM_OVER_BOTTOM  
  
LV_SCR_LOAD_ANIM_MOVE_LEFT  
  
LV_SCR_LOAD_ANIM_MOVE_RIGHT  
  
LV_SCR_LOAD_ANIM_MOVE_TOP  
  
LV_SCR_LOAD_ANIM_MOVE_BOTTOM  
  
LV_SCR_LOAD_ANIM_FADE_IN  
  
LV_SCR_LOAD_ANIM_FADE_ON  
  
LV_SCR_LOAD_ANIM_FADE_OUT  
  
LV_SCR_LOAD_ANIM_OUT_LEFT  
  
LV_SCR_LOAD_ANIM_OUT_RIGHT  
  
LV_SCR_LOAD_ANIM_OUT_TOP
```

`LV_SCR_LOAD_ANIM_OUT_BOTTOM`

`lv_ft81x_spi_operation`

14.5 lv_api_map_v9_3.h

Define

`lv_draw_buf_malloc_cb`

`lv_draw_buf_free_cb`

`lv_draw_buf_copy_cb`

`lv_draw_buf_align_cb`

`lv_draw_buf_cache_operation_cb`

`lv_draw_buf_width_to_stride_cb`

`lv_glfw_window_t`

`lv_glfw_texture_t`

`lv_glfw_window_create`

`lv_glfw_window_create_ex`

`lv_glfw_window_set_title`

`lv_glfw_window_delete`

`lv_glfw_window_set_flip`

`lv_glfw_window_add_texture`

`lv_glfw_texture_remove`

`lv_glfw_texture_set_x`

`lv_glfw_texture_set_y`

`lv_glfw_texture_set_opa`

`lv_glfw_texture_get_mouse_indev`

```
lv_glfw_window_get_glfw_window

lv_vector_dsc_create

lv_vector_dsc_delete

lv_vector_dsc_set_transform

lv_vector_dsc_set_blend_mode

lv_vector_dsc_set_fill_color32

lv_vector_dsc_set_fill_color

lv_vector_dsc_set_fill_opa

lv_vector_dsc_set_fill_rule

lv_vector_dsc_set_fill_units

lv_vector_dsc_set_fill_image

lv_vector_dsc_set_fill_linear_gradient

lv_vector_dsc_set_fill_radial_gradient

lv_vector_dsc_set_fill_gradient_spread

lv_vector_dsc_set_fill_gradient_color_stops

lv_vector_dsc_set_fill_transform

lv_vector_dsc_set_stroke_color32

lv_vector_dsc_set_stroke_color

lv_vector_dsc_set_stroke_opa

lv_vector_dsc_set_stroke_width

lv_vector_dsc_set_stroke_dash

lv_vector_dsc_set_stroke_cap

lv_vector_dsc_set_stroke_join
```

```
lv_vector_dsc_set_stroke_miter_limit  
lv_vector_dsc_set_stroke_linear_gradient  
lv_vector_dsc_set_stroke_radial_gradient  
lv_vector_dsc_set_stroke_gradient_spread  
lv_vector_dsc_set_stroke_gradient_color_stops  
lv_vector_dsc_set_stroke_transform  
lv_vector_dsc_identity  
lv_vector_dsc_scale  
lv_vector_dsc_rotate  
lv_vector_dsc_translate  
lv_vector_dsc_skew  
lv_vector_dsc_add_path  
lv_vector_dsc_clear_area
```

14.6 lv_api_map_v9_4.h

Define

```
lv_tabview_rename_tab
```

14.7 lv_conf_internal.h

GENERATED FILE, DO NOT EDIT IT!

This file ensures all defines of lv_conf.h have a default value.

Define

```
LV_OS_NONE
```

```
LV_OS_PTHREAD
```

```
LV_OS_FREERTOS
```

LV_OS_CMSIS_RTOS2

LV_OS_RTTHREAD

LV_OS_WINDOWS

LV_OS_MQX

LV_OS_SDL2

LV_OS_CUSTOM

LV_STDLIB_BUILTIN

LV_STDLIB_CLIB

LV_STDLIB_MICROPYTHON

LV_STDLIB_RTTHREAD

LV_STDLIB_CUSTOM

LV_DRAW_SW_ASM_NONE

LV_DRAW_SW_ASM_NEON

LV_DRAW_SW_ASM_HELIUM

LV_DRAW_SW_ASM_CUSTOM

LV_NEMA_HAL_CUSTOM

LV_NEMA_HAL_STM32

LV_COLOR_DEPTH

Handle special Kconfig options. Color depth: 1 (I1), 8 (L8), 16 (RGB565), 24 (RGB888), 32 (XRGB8888)

LV_USE_STDLIB_MALLOC

Possible values

- **LV_STDLIB_BUILTIN**: LVGL's built in implementation
- **LV_STDLIB_CLIB**: Standard C functions, like malloc, strlen, etc
- **LV_STDLIB_MICROPYTHON**: MicroPython implementation
- **LV_STDLIB_RTTHREAD**: RT-Thread implementation
- **LV_STDLIB_CUSTOM**: Implement the functions externally

LV_USE_STDLIB_STRING

Possible values

- LV_STDLIB_BUILTIN: LVGL's built in implementation
- LV_STDLIB_CLIB: Standard C functions, like malloc, strlen, etc
- LV_STDLIB_MICROPYTHON: MicroPython implementation
- LV_STDLIB_RTTHREAD: RT-Thread implementation
- LV_STDLIB_CUSTOM: Implement the functions externally

LV_USE_STDLIB_SPRINTF

Possible values

- LV_STDLIB_BUILTIN: LVGL's built in implementation
- LV_STDLIB_CLIB: Standard C functions, like malloc, strlen, etc
- LV_STDLIB_MICROPYTHON: MicroPython implementation
- LV_STDLIB_RTTHREAD: RT-Thread implementation
- LV_STDLIB_CUSTOM: Implement the functions externally

LV_STDINT_INCLUDE**LV_STDDEF_INCLUDE****LV_STDBOOL_INCLUDE****LV_INTTYPES_INCLUDE****LV_LIMITS_INCLUDE****LV_STDARG_INCLUDE****LV_MEM_SIZE**Size of memory available for `lv_malloc()` in bytes (>= 2kB) [bytes]**LV_MEM_POOL_EXPAND_SIZE**Size of the memory expand for `lv_malloc()` in bytes**LV_MEM_ADDR**Set an address for the memory pool instead of allocating it as a normal array. Can be in external SRAM too.
0: unused**LV_DEF_REFR_PERIOD**

Default display refresh, input device read and animation step period. [ms]

LV_DPI_DEF

Default Dots Per Inch. Used to initialize default sizes such as widgets sized, style paddings. (Not so important, you can adjust it to modify default sizes and spaces.) [px/inch]

LV_USE_OS

Select operating system to use. Possible options:

- LV_OS_NONE
- LV_OS_PTHREAD
- LV_OS_FREERTOS

- LV_OS_CMSIS_RTOS2
- LV_OS_RTTHREAD
- LV_OS_WINDOWS
- LV_OS_MQX
- LV_OS SDL2
- LV_OS_CUSTOM

LV_DRAW_BUF_STRIDE_ALIGN

Align stride of all layers and images to this bytes

LV_DRAW_BUF_ALIGN

Align start address of draw_buf addresses to this bytes

LV_DRAW_TRANSFORM_USE_MATRIX

Using matrix for transformations. Requirements:

- LV_USE_MATRIX = 1.
- Rendering engine needs to support 3x3 matrix transformations.

LV_DRAW_LAYER_SIMPLE_BUF_SIZE

The target buffer size for simple layer chunks. [bytes]

LV_DRAW_LAYER_MAX_MEMORY

No limit by default [bytes]

LV_DRAW_THREAD_STACK_SIZE

Stack size of drawing thread. NOTE: If FreeType or ThorVG is enabled, it is recommended to set it to 32KB or more. [bytes]

LV_DRAW_THREAD_PRIO

Thread priority of the drawing task. Higher values mean higher priority. Can use values from lv_thread_prio_t enum in lv_os.h: LV_THREAD_PRIO_LOWEST, LV_THREAD_PRIO_LOW, LV_THREAD_PRIO_MID, LV_THREAD_PRIO_HIGH, LV_THREAD_PRIO_HIGHEST Make sure the priority value aligns with the OS-specific priority levels. On systems with limited priority levels (e.g., FreeRTOS), a higher value can improve rendering performance but might cause other tasks to starve.

LV_USE_DRAW_SW**LV_DRAW_SW_SUPPORT_RGB565****LV_DRAW_SW_SUPPORT_RGB565_SWAPPED****LV_DRAW_SW_SUPPORT_RGB565A8****LV_DRAW_SW_SUPPORT_RGB888****LV_DRAW_SW_SUPPORT_XRGB8888****LV_DRAW_SW_SUPPORT_ARGB8888****LV_DRAW_SW_SUPPORT_ARGB8888_PREMULTIPLIED**

LV_DRAW_SW_SUPPORT_L8

LV_DRAW_SW_SUPPORT_AL88

LV_DRAW_SW_SUPPORT_A8

LV_DRAW_SW_SUPPORT_I1

LV_DRAW_SW_I1_LUM_THRESHOLD

LV_DRAW_SW_DRAW_UNIT_CNT

Set number of draw units.

- > 1 requires operating system to be enabled in `LV_USE_OS`.
- > 1 means multiple threads will render the screen in parallel.

LV_USE_DRAW_ARM2D_SYNC

Use Arm-2D to accelerate software (sw) rendering.

LV_USE_NATIVE_HELIUM_ASM

Enable native helium assembly to be compiled.

LV_DRAW_SW_COMPLEX

- 0: Use a simple renderer capable of drawing only simple rectangles with gradient, images, text, and straight lines only.
- 1: Use a complex renderer capable of drawing rounded corners, shadow, skew lines, and arcs too.

LV_DRAW_SW_SHADOW_CACHE_SIZE

Allow buffering some shadow calculation. `LV_DRAW_SW_SHADOW_CACHE_SIZE` is the maximum shadow size to buffer, where shadow size is `shadow_width + radius`. Caching has `LV_DRAW_SW_SHADOW_CACHE_SIZE^2` RAM cost.

LV_DRAW_SW_CIRCLE_CACHE_SIZE

Set number of maximally-cached circle data. The circumference of 1/4 circle are saved for anti-aliasing. `radius * 4` bytes are used per circle (the most often used radii are saved).

- 0: disables caching

LV_USE_DRAW_SW_ASM

LV_USE_DRAW_SW_COMPLEX_GRADIENTS

Enable drawing complex gradients in software: linear at an angle, radial or conical

LV_USE_NEMA_GFX

LV_USE_PXP

Use NXP's PXP on iMX RTxxx platforms.

LV_USE_G2D

Use NXP's G2D on MPU platforms.

LV_USE_DRAW_DAVE2D

Use Renesas Dave2D on RA platforms.

LV_USE_DRAW SDL

Draw using cached SDL textures

LV_USE_DRAW_VG_LITE

Use VG-Lite GPU.

LV_USE_DRAW_DMA2D

Accelerate blends, fills, etc. with STM32 DMA2D

LV_USE_DRAW_OPENGL_ES

Draw using cached OpenGL ES textures. Requires LV_USE_OPENGL_ES

LV_USE_PPA

Draw using espressif PPA accelerator

LV_USE_DRAW_EVE**LV_USE_LOG**

Enable log module

LV_USE_ASSERT_NULL

Check if the parameter is NULL. (Very fast, recommended)

LV_USE_ASSERT_MALLOC

Checks if the memory is successfully allocated or not. (Very fast, recommended)

LV_USE_ASSERT_STYLE

Check if the styles are properly initialized. (Very fast, recommended)

LV_USE_ASSERT_MEM_INTEGRITY

Check the integrity of lv_mem after critical operations. (Slow)

LV_USE_ASSERT_OBJ

Check the object's type and existence (e.g. not deleted). (Slow)

LV_ASSERT_HANDLER_INCLUDE

Add a custom handler when assert happens e.g. to restart MCU.

LV_ASSERT_HANDLER

Halt by default

LV_USE_REFR_DEBUG

1: Draw random colored rectangles over the redrawn areas.

LV_USE_LAYER_DEBUG

1: Draw a red overlay for ARGB layers and a green overlay for RGB layers

LV_USE_PARALLEL_DRAW_DEBUG

1: Adds the following behaviors for debugging:

- Draw overlays with different colors for each draw_unit's tasks.
- Draw index number of draw unit on white background.
- For layers, draws index number of draw unit on black background.

LV_ENABLE_GLOBAL_CUSTOM**LV_CACHE_DEF_SIZE**

Default cache size in bytes. Used by image decoders such as lv_lodepng to keep the decoded image in memory. If size is not set to 0, the decoder will fail to decode when the cache is full. If size is 0, the cache function is not enabled and the decoded memory will be released immediately after use.

LV_IMAGE_HEADER_CACHE_DEF_CNT

Default number of image header cache entries. The cache is used to store the headers of images. The main logic is like LV_CACHE_DEF_SIZE but for image headers.

LV_GRADIENT_MAX_STOPS

Number of stops allowed per gradient. Increase this to allow more stops. This adds (sizeof(lv_color_t) + 1) bytes per additional stop.

LV_COLOR_MIX_ROUND_OFS

Adjust color mix functions rounding. GPUs might calculate color mix (blending) differently.

- 0: round down,
- 64: round up from x.75,
- 128: round up from half,
- 192: round up from x.25,
- 254: round up

LV_OBJ_STYLE_CACHE

Add 2 x 32-bit variables to each lv_obj_t to speed up getting style properties

LV_USE_OBJ_ID

Add id field to lv_obj_t

LV_USE_OBJ_NAME

Enable support widget names

LV_OBJ_ID_AUTO_ASSIGN

Automatically assign an ID when obj is created

LV_USE_OBJ_ID_BUILTIN

Use builtin obj ID handler functions:

- lv_obj_assign_id: Called when a widget is created. Use a separate counter for each widget class as an ID.
- lv_obj_id_compare: Compare the ID to decide if it matches with a requested value.
- lv_obj_stringify_id: Return string-ified identifier, e.g. "button3".
- lv_obj_free_id: Does nothing, as there is no memory allocation for the ID. When disabled these functions needs to be implemented by the user.

LV_USE_OBJ_PROPERTY

Use obj property set/get API.

LV_USE_OBJ_PROPERTY_NAME

Enable property name support.

LV_USE_GESTURE_RECOGNITION**LV_BIG_ENDIAN_SYSTEM**

For big endian systems set to 1

LV_ATTRIBUTE_MEM_ALIGN_SIZE

Define a custom attribute for `lv_tick_inc` function
 Define a custom attribute for `lv_timer_handler` function
 Define a custom attribute for `lv_display_flush_ready` function
 Align VG_LITE buffers on this number of bytes.

Nota

`vglite_src_buf_aligned()` uses this value to validate alignment of passed buffer pointers.

LV_EXPORT_CONST_INT(int_value)

Will be added where memory needs to be aligned (with -Os data might not be aligned to boundary by default).
 E.g. `attribute((aligned(4)))` Attribute to mark large constant arrays, for example for font bitmaps Compiler prefix for a large array declaration in RAM Place performance critical functions into a faster memory (e.g RAM) Export integer constant to binding. This macro is used with constants in the form of `LV_<CONST>` that should also appear on LVGL binding API such as MicroPython. The default value just prevents GCC warning

LV_USE_FLOAT

Prefix all global extern data with this Use float as `lv_value_precise_t`

LV_USE_MATRIX

Enable matrix support

- Requires `LV_USE_FLOAT = 1`

LV_USE_PRIVATE_API

Include `lvgl_private.h` in `lvgl.h` to access internal data and functions by default

LV_FONT_MONTserrat_8**LV_FONT_MONTserrat_10****LV_FONT_MONTserrat_12****LV_FONT_MONTserrat_14****LV_FONT_MONTserrat_16****LV_FONT_MONTserrat_18****LV_FONT_MONTserrat_20**

LV_FONT_MONTserrat_22

LV_FONT_MONTserrat_24

LV_FONT_MONTserrat_26

LV_FONT_MONTserrat_28

LV_FONT_MONTserrat_30

LV_FONT_MONTserrat_32

LV_FONT_MONTserrat_34

LV_FONT_MONTserrat_36

LV_FONT_MONTserrat_38

LV_FONT_MONTserrat_40

LV_FONT_MONTserrat_42

LV_FONT_MONTserrat_44

LV_FONT_MONTserrat_46

LV_FONT_MONTserrat_48

LV_FONT_MONTserrat_28_COMPRESSED

 bpp = 3

LV_FONT_DEJAVU_16_PERSIAN_HEBREW

 Hebrew, Arabic, Persian letters and all their forms

LV_FONT_SOURCE_HAN_SANS_SC_14_CJK

 1338 most common CJK radicals

LV_FONT_SOURCE_HAN_SANS_SC_16_CJK

 1338 most common CJK radicals

LV_FONT_UNSCII_8

 Pixel perfect monospaced fonts

LV_FONT_UNSCII_16

LV_FONT_CUSTOM_DECLARE

Optionally declare custom fonts here.

You can use any of these fonts as the default font too and they will be available globally. Example:

```
#define LV_FONT_CUSTOM_DECLARE LV_FONT_DECLARE(my_font_1) LV_FONT_DECLARE(my_font_2)
```

LV_FONT_DEFAULT

Always set a default font

LV_FONT_FMT_TXT_LARGE

Enable handling large font and/or fonts with a lot of characters. The limit depends on the font size, font face and bpp. A compiler error will be triggered if a font needs it.

LV_USE_FONT_COMPRESSED

Enables/disables support for compressed fonts.

LV_USE_FONT_PLACEHOLDER

Enable drawing placeholders when glyph dsc is not found.

LV_TXT_ENC

Select a character encoding for strings. Your IDE or editor should have the same character encoding.

- LV_TXT_ENC_UTF8
- LV_TXT_ENC_ASCII

LV_TXT_BREAK_CHARS

While rendering text strings, break (wrap) text on these chars.

LV_TXT_LINE_BREAK_LONG_LEN

If a word is at least this long, will break wherever "prettiest". To disable, set to a value <= 0.

LV_TXT_LINE_BREAK_LONG_PRE_MIN_LEN

Minimum number of characters in a long word to put on a line before a break. Depends on LV_TXT_LINE_BREAK_LONG_LEN.

LV_TXT_LINE_BREAK_LONG_POST_MIN_LEN

Minimum number of characters in a long word to put on a line after a break. Depends on LV_TXT_LINE_BREAK_LONG_LEN.

LV_USE_BIDI

Support bidirectional text. Allows mixing Left-to-Right and Right-to-Left text. The direction will be processed according to the Unicode Bidirectional Algorithm: <https://www.w3.org/International/articles/inline-bidi-markup/uba-basics>

LV_USE_ARABIC_PERSIAN_CHARS

Enable Arabic/Persian processing In these languages characters should be replaced with another form based on their position in the text

LV_TXT_COLOR_CMD

LV_WIDGETS_HAS_DEFAULT_VALUE

- 1: Causes these widgets to be given default values at creation time.
- lv_buttonmatrix_t: Get default maps: {"Btn1", "Btn2", "Btn3", "\n", "Btn4", "Btn5", ""}, else map not set.
 - lv_checkbox_t : String label set to "Check box", else set to empty string.
 - lv_dropdown_t : Options set to "Option 1", "Option 2", "Option 3", else no values are set.
 - lv_roller_t : Options set to "Option 1", "Option 2", "Option 3", "Option 4", "Option 5", else no values are set.
 - lv_label_t : Text set to "Text", else empty string.
 - lv_arclabel_t : Text set to "Arced Text", else empty string.

LV_USE_ANIMIMG**LV_USE_ARC****LV_USE_ARCLABEL****LV_USE_BAR****LV_USE_BUTTON****LV_USE_BUTTONMATRIX****LV_USE_CALENDAR****LV_CALENDAR_WEEK_STARTS_MONDAY****LV_CALENDAR_DEFAULT_DAY_NAMES****LV_CALENDAR_DEFAULT_MONTH_NAMES****LV_USE_CALENDAR_HEADER_ARROW****LV_USE_CALENDAR_HEADER_DROPDOWN****LV_USE_CALENDAR_CHINESE****LV_USE_CANVAS****LV_USE_CHART****LV_USE_CHECKBOX****LV_USE_DROPDOWN**

Requires: lv_label

LV_USE_IMAGE

Requires: lv_label

LV_USE_IMAGEBUTTON

LV_USE_KEYBOARD

LV_USE_LABEL

LV_LABEL_TEXT_SELECTION

Enable selecting text of the label

LV_LABEL_LONG_TXT_HINT

Store some extra info in labels to speed up drawing of very long text

LV_LABEL_WAIT_CHAR_COUNT

The count of wait chart

LV_USE_LED

LV_USE_LINE

LV_USE_LIST

LV_USE_LOTTIE

Requires: lv_canvas, thorvg

LV_USE_MENU

LV_USE_MSGBOX

LV_USE_ROLLER

Requires: lv_label

LV_USE_SCALE

LV_USE_SLIDER

Requires: lv_bar

LV_USE_SPAN

LV_SPAN_SNIPPET_STACK_SIZE

A line of text can contain this maximum number of span descriptors.

LV_USE_SPINBOX

LV_USE_SPINNER

LV_USE_SWITCH

LV_USE_TABLE

LV_USE_TABVIEW

LV_USE_TEXTAREA

Requires: lv_label

LV_TEXTAREA_DEF_PWD_SHOW_TIME

[ms]

LV_USE_TILEVIEW

LV_USE_WIN

LV_USE_3DTEXTURE

LV_USE_THEME_DEFAULT

A simple, impressive and very complete theme

LV_THEME_DEFAULT_DARK

0: Light mode; 1: Dark mode

LV_THEME_DEFAULT_GROW

1: Enable grow on press

LV_THEME_DEFAULT_TRANSITION_TIME

Default transition time in ms.

LV_USE_THEME_SIMPLE

A very simple theme that is a good starting point for a custom theme

LV_USE_THEME_MONO

A theme designed for monochrome displays

LV_USE_FLEX

A layout similar to Flexbox in CSS.

LV_USE_GRID

A layout similar to Grid in CSS.

LV_FS_DEFAULT_DRIVER LETTER

Setting a default driver letter allows skipping the driver prefix in filepaths. Documentation about how to use the below driver-identifier letters can be found at <https://docs.lvgl.io/master/details/main-modules/fs.html#lv-fs-identifier-letters>.

LV_USE_FS_STDIO

API for fopen, fread, etc.

LV_USE_FS_POSIX

API for open, read, etc.

LV_USE_FS_WIN32

API for CreateFile, ReadFile, etc.

LV_USE_FS_FATFS

API for FATFS (needs to be added separately). Uses f_open, f_read, etc.

LV_USE_FS_MEMFS

API for memory-mapped file access.

LV_USE_FS_LITTLEFS

API for LittleFs.

LV_USE_FS_ARDUINO_ESP_LITTLEFS

API for Arduino LittleFs.

LV_USE_FS_ARDUINO_SD

API for Arduino Sd.

LV_USE_FS_UEFI

API for UEFI

LV_USE_FS_FROGFS**LV_USE_LODEPNG**

LODEPNG decoder library

LV_USE_LIBPNG

PNG decoder(libpng) library

LV_USE_BMP

BMP decoder library

LV_USE_TJPGD

JPG + split JPG decoder library. Split JPG is a custom format optimized for embedded systems.

LV_USE_LIBJPEG_TURBO

libjpeg-turbo decoder library.

- Supports complete JPEG specifications and high-performance JPEG decoding.

LV_USE_LIBWEBP

WebP decoder library

LV_USE_GIF

GIF decoder library

LV_USE_GSTREAMER

GStreamer library

LV_BIN_DECODER_RAM_LOAD

Decode bin images to RAM

LV_USE_RLE

RLE decompress library

LV_USE_QRCODE

QR code library

LV_USE_BARCODE

Barcode code library

LV_USE_FREETYPE

FreeType library

LV_USE_TINY_TTF

Built-in TTF decoder

LV_USE_RLOTTIE

Rlottie library

LV_USE_GLTF

Requires LV_USE_3DTEXTURE = 1

LV_USE_VECTOR_GRAPHIC

Enable Vector Graphic APIs Requires LV_USE_MATRIX = 1

LV_USE_THORVG_INTERNAL

Enable ThorVG (vector graphics library) from the src/libs folder. Requires LV_USE_VECTOR_GRAPHIC

LV_USE_THORVG_EXTERNAL

Enable ThorVG by assuming that its installed and linked to the project Requires LV_USE_VECTOR_GRAPHIC

LV_USE_LZ4_INTERNAL

Use lvgl built-in LZ4 lib

LV_USE_LZ4_EXTERNAL

Use external LZ4 library

LV_USE_SVG**LV_USE_SVG_ANIMATION****LV_USE_SVG_DEBUG**

LV_USE_FFMPEG

FFmpeg library for image decoding and playing videos. Supports all major image formats so do not enable other image decoder with it.

LV_USE_SNAPSHOT

1: Enable API to take snapshot for object

LV_USE_SYSMON

1: Enable system monitor component

LV_USE_PROFILER

1: Enable runtime performance profiler

LV_USE_MONKEY

1: Enable Monkey test

LV_USE_GRIDNAV

1: Enable grid navigation

LV_USE_FRAGMENT

1: Enable lv_obj fragment logic

LV_USE_IMGFONT

1: Support using images as font in label or span widgets

LV_USE_OBSERVER

1: Enable an observer pattern implementation

LV_USE_IME_PINYIN

1: Enable Pinyin input method
• Requires: lv_keyboard

LV_USE_FILE_EXPLORER

1: Enable file explorer.
• Requires: lv_table

LV_USE_FONT_MANAGER

1: Enable Font manager

LV_USE_TEST

Enable emulated input devices, time emulation, and screenshot compares.

LV_USE_XML

Enable loading XML UIs runtime

LV_USE_TRANSLATION

1: Enable text translation support

LV_USE_COLOR_FILTER

LV_USE SDL

Use SDL to open window on PC and handle mouse and keyboard.

LV_USE_X11

Use X11 to open window on Linux desktop and handle mouse and keyboard

LV_USE WAYLAND

Use Wayland to open a window and handle input on Linux or BSD desktops

LV_USE_LINUX_FBDEV

Driver for /dev/fb

LV_USE_NUTTX

Use Nuttx to open window and handle touchscreen

LV_USE_LINUX_DRM

Driver for /dev/dri/card

LV_USE_TFT_ESPI

Interface for TFT_eSPI

LV_USE_LOVYAN_GFX

Interface for Lovyan_GFX

LV_USE_EVDEV

Driver for evdev input devices

LV_USE_LIBINPUT

Driver for libinput input devices

LV_USE_ST7735

LV_USE_ST7789

LV_USE_ST7796

LV_USE_ILI9341

LV_USE_FT81X

LV_USE_NV3007

LV_USE_GENERIC_MIPI

LV_USE_RENESAS_GLCD

Driver for Renesas GLCD

LV_USE_ST_LTDC

Driver for ST LTDC

LV_USE_NXP_ELCDIF

Driver for NXP ELCDIF

LV_USE_WINDOWS

LVGL Windows backend

LV_USE_UEFI

LVGL UEFI backend

LV_USE_OPENGLS

Use a generic OpenGL driver that can be used to embed in other applications or used with GLFW/EGL

LV_USE_GLFW

Use GLFW to open window on PC and handle mouse and keyboard. Requires

LV_USE_QNX

QNX Screen display and input drivers

LV_BUILD_EXAMPLES

Enable examples to be built with the library.

LV_BUILD_DEMOS

Build the demos

LV_USE_DEMO_WIDGETS

Show some widgets. This might be required to increase `LV_MEM_SIZE`.

LV_USE_DEMO_KEYPAD_AND_ENCODER

Demonstrate usage of encoder and keyboard.

LV_USE_DEMO_BENCHMARK

Benchmark your system

LV_USE_DEMO_RENDER

Render test for each primitive.

- Requires at least 480x272 display.

LV_USE_DEMO_STRESS

Stress test for LVGL

LV_USE_DEMO_MUSIC

Music player demo

LV_USE_DEMO_VECTOR_GRAPHIC

Vector graphic demo

LV_USE_DEMO_GLTF

GLTF demo

LV_USE_DEMO_FLEX_LAYOUT

Flex layout demo

LV_USE_DEMO_MULTILANG

Smart-phone like multi-language demo

LV_USE_DEMO_TRANSFORM

Widget transformation demo

LV_USE_DEMO_EBIKE

LV_USE_DEMO_HIGH_RES

High-resolution demo

LV_USE_DEMO_SMARTWATCH

LV_USE_ANIMIMAGE

LV_USE_VG_LITE_DRIVER

LV_USE_VG_LITE_THORVG

LV_LOG_LEVEL

LV_LOG_TRACE_MEM

LV_LOG_TRACE_TIMER

LV_LOG_TRACE_INDEV

LV_LOG_TRACE_DISP_REFR

LV_LOG_TRACE_EVENT

LV_LOG_TRACE_OBJ_CREATE

LV_LOG_TRACE_LAYOUT

LV_LOG_TRACE_ANIM

LV_WAYLAND_USE_DMABUF

LV_WAYLAND_WINDOW_DECORATIONS

LV_LINUX_DRM_USE_EGL**LV_USE_PERF_MONITOR****LV_USE_MEM_MONITOR****LV_SYSMON_PROC_IDLE_AVAILABLE****LV_USE_PERF_MONITOR_LOG_MODE****LV_USE_LZ4****LV_USE_THORVG****LV_USE_EGL****LV_FS_IS_VALID_LETTER(l)**

14.8 lv_conf_kconfig.h

- * Configs that need special handling when LVGL is used with Kconfig

14.9 lv_init.h

Funzioni

void lv_init(void)

Initialize LVGL library. Should be called before any other LVGL related function.

void lv_deinit(void)

Deinit the 'lv' library

bool lv_is_initialized(void)

Returns whether the 'lv' library is currently initialized

14.10 lvgl.h

Questo file esiste solo per essere compatibile con la struttura della libreria di Arduino

14.11 lvgl_private.h

Questo file esiste solo per essere compatibile con la struttura della libreria di Arduino

14.12 core

14.12.1 lv_global.h

Define

ZERO_MEM_SENTINEL

LV_GLOBAL_DEFAULT()

Typedef

typedef struct *lv_global_t* **lv_global_t**

Variabili

lv_global_t **lv_global**

struct **lv_global_t**

Membri Pubblici

void *user_data

User data for the LVGL library. Move from the bottom of the struct to avoid breaking the ABI. E.g., if the user data is used by a closed-source library, this can help to avoid re-compiling the library when the lvgl-related configs are changed.

bool inited

bool deinit_in_progress

Can be used e.g. in the LV_EVENT_DELETE to deinit the drivers too

lv_ll_t **disp_ll**

lv_display_t *disp_refresh

lv_display_t *disp_default

lv_ll_t **style_trans_ll**

bool style_refresh

uint32_t style_custom_table_size

uint32_t style_last_custom_prop_id

uint8_t *style_custom_prop_flag_lookup_table

lv_ll_t **group_ll**

lv_group_t *group_default

lv_ll_t **indev_ll**

```
lv_indev_t *indev_active  
  
lv_obj_t *indev_obj_active  
  
uint32_t layout_count  
  
lv_layout_dsc_t *layout_list  
  
bool layout_update_mutex  
  
uint32_t memory_zero  
  
uint32_t math_rand_seed  
  
lv_event_t *event_header  
  
uint32_t event_last_register_id  
  
lv_timer_state_t timer_state  
  
lv_anim_state_t anim_state  
  
lv_tick_state_t tick_state  
  
lv_draw_buf_handlers_t draw_buf_handlers  
  
lv_draw_buf_handlers_t font_draw_buf_handlers  
  
lv_draw_buf_handlers_t image_cache_draw_buf_handlers  
    Ensure that all assigned draw buffers can be managed by image cache.  
  
lv_ll_t img_decoder_ll  
  
lv_cache_t *img_cache  
  
lv_cache_t *img_header_cache  
  
lv_draw_global_info_t draw_info  
  
lv_ll_t draw_sw_blend_handler_ll  
  
lv_draw_sw_mask_radius_circle_dsc_arr_t sw_circle_cache  
  
lv_log_print_g_cb_t custom_log_print_cb
```

```
uint32_t log_last_log_time

void *theme_simple

void *theme_default

void *theme_mono

lv_tlsf_state_t tlsf_state

lv_ll_t fsdrv_ll

lv_fs_drv_t stdio_fs_drv

lv_fs_drv_t posix_fs_drv

lv_fs_drv_t fatfs_fs_drv

lv_fs_drv_t win32_fs_drv

lv_fs_drv_t uefi_fs_drv

lv_fs_drv_t littlefs_fs_drv

lv_fs_drv_t arduino_esp_littlefs_fs_drv

lv_fs_drv_t arduino_sd_fs_drv

lv_fs_drv_t frogfs_fs_drv

struct _lv_freetype_context_t *ft_context

lv_font_fmt_rle_t font_fmt_rle

struct _snippet_stack *span_snippet_stack

lv_style_t file_explorer_quick_access_style

size_t file_explorer_count

lv_sysmon_backend_data_t sysmon_mem

size_t imcand_len

void *objid_array
```

```

uint32_t objid_count

lv_test_state_t test_state

lv_ll_t translation_packs_ll

const char *translation_selected_lang

struct _lv_nuttx_ctx_t *nuttx_ctx

lv_evdev_discovery_t *evdev_discovery

const char *xml_path_prefix

uint32_t lv_event_xml_store_timeline

lv_ll_t xml_loads

lv_draw_eve_unit_t *draw_eve_unit

```

14.12.2 lv_group.h

Typedef

`typedef void (*lv_group_focus_cb_t)(lv_group_t*)`

`typedef void (*lv_group_edge_cb_t)(lv_group_t*, bool)`

Enum

`enum lv_key_t`

Predefined keys to control which Widget has focus via `lv_group_send(group, c)`

Values:

enumerator `LV_KEY_UP`

enumerator `LV_KEY_DOWN`

enumerator `LV_KEY_RIGHT`

enumerator `LV_KEY_LEFT`

enumerator `LV_KEY_ESC`

enumerator `LV_KEY_DEL`

enumerator **LV_KEY_BACKSPACE**

enumerator **LV_KEY_ENTER**

enumerator **LV_KEY_NEXT**

enumerator **LV_KEY_PREV**

enumerator **LV_KEY_HOME**

enumerator **LV_KEY_END**

enum **lv_group_refocus_policy_t**

Values:

enumerator **LV_GROUP_REFOCUS_POLICY_NEXT**

enumerator **LV_GROUP_REFOCUS_POLICY_PREV**

Funzioni

lv_group_t *lv_group_create(void)

Create new Widget group.

Ritorna

pointer to the new Widget group

void lv_group_delete(lv_group_t *group)

Delete group object.

Parametri

group -- pointer to a group

void lv_group_set_default(lv_group_t *group)

Set default group. New Widgets will be added to this group if it's enabled in their class with `add_to_def_group = true`.

Parametri

group -- pointer to a group (can be NULL)

lv_group_t *lv_group_get_default(void)

Get default group.

Ritorna

pointer to the default group

void lv_group_add_obj(lv_group_t *group, lv_obj_t *obj)

Add an Widget to group.

Parametri

- **group** -- pointer to a group
- **obj** -- pointer to a Widget to add

void lv_group_swap_obj(lv_obj_t *obj1, lv_obj_t *obj2)

Swap 2 Widgets in group. Widgets must be in the same group.

Parametri

- **obj1** -- pointer to a Widget
- **obj2** -- pointer to another Widget

`void lv_group_remove_obj(lv_obj_t *obj)`

Remove a Widget from its group.

Parametri

- **obj** -- pointer to Widget to remove

`void lv_group_remove_all_objs(lv_group_t *group)`

Remove all Widgets from a group.

Parametri

- **group** -- pointer to a group

`void lv_group_focus_obj(lv_obj_t *obj)`

Focus on a Widget (defocus the current).

Parametri

- **obj** -- pointer to Widget to focus on

`void lv_group_focus_next(lv_group_t *group)`

Focus on next Widget in a group (defocus the current).

Parametri

- **group** -- pointer to a group

`void lv_group_focus_prev(lv_group_t *group)`

Focus on previous Widget in a group (defocus the current).

Parametri

- **group** -- pointer to a group

`void lv_group_focus_freeze(lv_group_t *group, bool en)`

Do not allow changing focus from current Widget.

Parametri

- **group** -- pointer to a group
- **en** -- true: freeze, false: release freezing (normal mode)

`lv_result_t lv_group_send_data(lv_group_t *group, uint32_t c)`

Send a control character to Widget that has focus in a group.

Parametri

- **group** -- pointer to a group
- **c** -- a character (use LV_KEY_... to navigate)

Ritorna

result of Widget with focus in group.

`void lv_group_set_focus_cb(lv_group_t *group, lv_group_focus_cb_t focus_cb)`

Set a function for a group which will be called when a new Widget has focus.

Parametri

- **group** -- pointer to a group
- **focus_cb** -- the call back function or NULL if unused

`void lv_group_set_edge_cb(lv_group_t *group, lv_group_edge_cb_t edge_cb)`

Set a function for a group which will be called when a focus edge is reached

Parametri

- **group** -- pointer to a group
- **edge_cb** -- the call back function or NULL if unused

`void lv_group_set_refocus_policy(lv_group_t *group, lv_group_refocus_policy_t policy)`

Set whether the next or previous Widget in a group gets focus when Widget that has focus is deleted.

Parametri

- **group** -- pointer to a group
- **policy** -- new refocus policy enum

`void lv_group_set_editing(lv_group_t *group, bool edit)`

Manually set the current mode (edit or navigate).

Parametri

- **group** -- pointer to group

- **edit** -- true: edit mode; false: navigate mode

`void lv_group_set_wrap(lv_group_t *group, bool en)`

Set whether moving focus to next/previous Widget will allow wrapping from first->last or last->first Widget.

Parametri

- **group** -- pointer to group
- **en** -- true: wrapping enabled; false: wrapping disabled

`lv_obj_t *lv_group_get_focused(const lv_group_t *group)`

Get Widget that has focus, or NULL if there isn't one.

Parametri

group -- pointer to a group

Ritorna

pointer to Widget with focus

`lv_group_focus_cb_t lv_group_get_focus_cb(const lv_group_t *group)`

Get focus callback function of a group.

Parametri

group -- pointer to a group

Ritorna

the call back function or NULL if not set

`lv_group_edge_cb_t lv_group_get_edge_cb(const lv_group_t *group)`

Get edge callback function of a group.

Parametri

group -- pointer to a group

Ritorna

the call back function or NULL if not set

`bool lv_group_get_editing(const lv_group_t *group)`

Get current mode (edit or navigate).

Parametri

group -- pointer to group

Ritorna

true: edit mode; false: navigate mode

`bool lv_group_get_wrap(lv_group_t *group)`

Get whether moving focus to next/previous Widget will allow wrapping from first->last or last->first Widget.

Parametri

group -- pointer to group

`uint32_t lv_group_get_obj_count(lv_group_t *group)`

Get number of Widgets in group.

Parametri

group -- pointer to a group

Ritorna

number of Widgets in the group

`lv_obj_t *lv_group_get_obj_by_index(lv_group_t *group, uint32_t index)`

Get nth Widget within group.

Parametri

- **group** -- pointer to a group
- **index** -- index of Widget within the group

Ritorna

pointer to Widget

`uint32_t lv_group_get_count(void)`

Get the number of groups.

Ritorna

number of groups

`lv_group_t *lv_group_by_index(uint32_t index)`

Get a group by its index.

Parametri

`index` -- index of the group

Ritorna

pointer to the group

14.12.3 lv_group_private.h

Funzioni

`void lv_group_init(void)`

Init the group module

Remark

Internal function, do not call directly.

`void lv_group_deinit(void)`

Deinit the group module

Remark

Internal function, do not call directly.

`struct _lv_group_t`

#include <lv_group_private.h> Groups can be used to logically hold objects so that they can be individually focused. They are NOT for laying out objects on a screen (try layouts for that).

Membri Pubblici

`lv_ll_t obj_ll`

Linked list to store the objects in the group

`lv_obj_t **obj_focus`

The object in focus

`lv_group_focus_cb_t focus_cb`

A function to call when a new object is focused (optional)

`lv_group_edge_cb_t edge_cb`

A function to call when an edge is reached, no more focus targets are available in this direction (to allow edge feedback like a sound or a scroll bounce)

`void *user_data`

`uint8_t frozen`

1: can't focus to new object

`uint8_t editing`

1: Edit mode, 0: Navigate mode

uint8_t refocus_policy

1: Focus prev if focused on deletion. 0: Focus next if focused on deletion.

uint8_t wrap

1: Focus next/prev can wrap at end of list. 0: Focus next/prev stops at end of list.

14.12.4 lv_obj.h

Define

LV_ASSERT_OBJ(obj_p, obj_class)

LV_TRACE_OBJ_CREATE(...)

Enum**enum lv_obj_flag_t**

On/Off features controlling the object's behavior. OR-ed values are possible

Note: update obj flags corresponding properties below whenever add/remove flags or change bit definition of flags.

Values:

enumerator LV_OBJ_FLAG_HIDDEN

Make the object hidden. (Like it wasn't there at all)

enumerator LV_OBJ_FLAG_CLICKABLE

Make the object clickable by the input devices

enumerator LV_OBJ_FLAG_CLICK_FOCUSABLE

Add focused state to the object when clicked

enumerator LV_OBJ_FLAG_CHECKABLE

Toggle checked state when the object is clicked

enumerator LV_OBJ_FLAG_SCROLLABLE

Make the object scrollable

enumerator LV_OBJ_FLAG_SCROLL_ELASTIC

Allow scrolling inside but with slower speed

enumerator LV_OBJ_FLAG_SCROLL_MOMENTUM

Make the object scroll further when "thrown"

enumerator LV_OBJ_FLAG_SCROLL_ONE

Allow scrolling only one snappable children

enumerator LV_OBJ_FLAG_SCROLL_CHAIN_HOR

Allow propagating the horizontal scroll to a parent

enumerator **LV_OBJ_FLAG_SCROLL_CHAIN_VER**

Allow propagating the vertical scroll to a parent

enumerator **LV_OBJ_FLAG_SCROLL_CHAIN**

enumerator **LV_OBJ_FLAG_SCROLL_ON_FOCUS**

Automatically scroll object to make it visible when focused

enumerator **LV_OBJ_FLAG_SCROLL_WITH_ARROW**

Allow scrolling the focused object with arrow keys

enumerator **LV_OBJ_FLAG_SNAPPABLE**

If scroll snap is enabled on the parent it can snap to this object

enumerator **LV_OBJ_FLAG_PRESS_LOCK**

Keep the object pressed even if the press slid from the object

enumerator **LV_OBJ_FLAG_EVENT_BUBBLE**

Propagate the events to the parent too

enumerator **LV_OBJ_FLAG_GESTURE_BUBBLE**

Propagate the gestures to the parent

enumerator **LV_OBJ_FLAG_ADV_HITTEST**

Allow performing more accurate hit (click) test. E.g. consider rounded corners.

enumerator **LV_OBJ_FLAG_IGNORE_LAYOUT**

Make the object not positioned by the layouts

enumerator **LV_OBJ_FLAG_FLOATING**

Do not scroll the object when the parent scrolls and ignore layout

enumerator **LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS**

Send LV_EVENT_DRAW_TASK_ADDED events

enumerator **LV_OBJ_FLAG_OVERFLOW_VISIBLE**

Do not clip the children to the parent's ext draw size

enumerator **LV_OBJ_FLAG_EVENT_TRICKLE**

Propagate the events to the children too

enumerator **LV_OBJ_FLAG_STATE_TRICKLE**

Propagate the states to the children too

enumerator **LV_OBJ_FLAG_LAYOUT_1**

Custom flag, free to use by layouts

enumerator **LV_OBJ_FLAG_LAYOUT_2**

Custom flag, free to use by layouts

enumerator **LV_OBJ_FLAG_FLEX_IN_NEW_TRACK**

Start a new flex track on this item

enumerator **LV_OBJ_FLAG_WIDGET_1**

Custom flag, free to use by widget

enumerator **LV_OBJ_FLAG_WIDGET_2**

Custom flag, free to use by widget

enumerator **LV_OBJ_FLAG_USER_1**

Custom flag, free to use by user

enumerator **LV_OBJ_FLAG_USER_2**

Custom flag, free to use by user

enumerator **LV_OBJ_FLAG_USER_3**

Custom flag, free to use by user

enumerator **LV_OBJ_FLAG_USER_4**

Custom flag, free to use by user

enum **_lv_signed_prop_id_t**

Values:

enumerator **LV_PROPERTY_OBJ_FLAG_START**

enumerator **LV_PROPERTY_OBJ_FLAG_HIDDEN**

enumerator **LV_PROPERTY_OBJ_FLAG_CLICKABLE**

enumerator **LV_PROPERTY_OBJ_FLAG_CLICK_FOCUSABLE**

enumerator **LV_PROPERTY_OBJ_FLAG_CHECKABLE**

enumerator **LV_PROPERTY_OBJ_FLAG_SCROLLABLE**

enumerator **LV_PROPERTY_OBJ_FLAG_SCROLL_ELASTIC**

enumerator **LV_PROPERTY_OBJ_FLAG_SCROLL_MOMENTUM**

enumerator **LV_PROPERTY_OBJ_FLAG_SCROLL_ONE**

enumerator **LV_PROPERTY_OBJ_FLAG_SCROLL_CHAIN_HOR**

enumerator **LV_PROPERTY_OBJ_FLAG_SCROLL_CHAIN_VER**

enumerator **LV_PROPERTY_OBJ_FLAG_SCROLL_ON_FOCUS**

enumerator **LV_PROPERTY_OBJ_FLAG_SCROLL_WITH_ARROW**

enumerator **LV_PROPERTY_OBJ_FLAG_SNAPPABLE**

enumerator **LV_PROPERTY_OBJ_FLAG_PRESS_LOCK**

enumerator **LV_PROPERTY_OBJ_FLAG_EVENT_BUBBLE**

enumerator **LV_PROPERTY_OBJ_FLAG_GESTURE_BUBBLE**

enumerator **LV_PROPERTY_OBJ_FLAG_ADV_HITTEST**

enumerator **LV_PROPERTY_OBJ_FLAG_IGNORE_LAYOUT**

enumerator **LV_PROPERTY_OBJ_FLAG_FLOATING**

enumerator **LV_PROPERTY_OBJ_FLAG_SEND_DRAW_TASK_EVENTS**

enumerator **LV_PROPERTY_OBJ_FLAG_OVERFLOW_VISIBLE**

enumerator **LV_PROPERTY_OBJ_FLAG_EVENT_TRICKLE**

enumerator **LV_PROPERTY_OBJ_FLAG_STATE_TRICKLE**

enumerator **LV_PROPERTY_OBJ_FLAG_LAYOUT_1**

enumerator **LV_PROPERTY_OBJ_FLAG_LAYOUT_2**

enumerator **LV_PROPERTY_OBJ_FLAG_FLEX_IN_NEW_TRACK**

enumerator **LV_PROPERTY_OBJ_FLAG_WIDGET_1**

enumerator **LV_PROPERTY_OBJ_FLAG_WIDGET_2**

enumerator **LV_PROPERTY_OBJ_FLAG_USER_1**

enumerator **LV_PROPERTY_OBJ_FLAG_USER_2**

enumerator **LV_PROPERTY_OBJ_FLAG_USER_3**

enumerator **LV_PROPERTY_OBJ_FLAG_USER_4**

enumerator **LV_PROPERTY_OBJ_FLAG_END**

enumerator **LV_PROPERTY_OBJ_STATE_START**

enumerator **LV_PROPERTY_OBJ_STATE_CHECKED**

enumerator **LV_PROPERTY_OBJ_STATE_FOCUSED**

enumerator **LV_PROPERTY_OBJ_STATE_FOCUS_KEY**

enumerator **LV_PROPERTY_OBJ_STATE_EDITED**

enumerator **LV_PROPERTY_OBJ_STATE_HOVERED**

enumerator **LV_PROPERTY_OBJ_STATE_PRESSED**

enumerator **LV_PROPERTY_OBJ_STATE_SCROLLED**

enumerator **LV_PROPERTY_OBJ_STATE_DISABLED**

enumerator **LV_PROPERTY_OBJ_STATE_USER_1**

enumerator **LV_PROPERTY_OBJ_STATE_USER_2**

enumerator **LV_PROPERTY_OBJ_STATE_USER_3**

enumerator **LV_PROPERTY_OBJ_STATE_USER_4**

enumerator **LV_PROPERTY_OBJ_STATE_ANY**

enumerator **LV_PROPERTY_OBJ_STATE_END**

enumerator **LV_PROPERTY_OBJ_PARENT**

enumerator **LV_PROPERTY_OBJ_X**

enumerator **LV_PROPERTY_OBJ_Y**

enumerator **LV_PROPERTY_OBJ_W**

enumerator **LV_PROPERTY_OBJ_H**

enumerator **LV_PROPERTY_OBJ_CONTENT_WIDTH**

enumerator **LV_PROPERTY_OBJ_CONTENT_HEIGHT**

enumerator **LV_PROPERTY_OBJ_LAYOUT**

enumerator **LV_PROPERTY_OBJ_ALIGN**

enumerator **LV_PROPERTY_OBJ_SCROLLBAR_MODE**

enumerator **LV_PROPERTY_OBJ_SCROLL_DIR**

enumerator **LV_PROPERTY_OBJ_SCROLL_SNAP_X**

enumerator **LV_PROPERTY_OBJ_SCROLL_SNAP_Y**

enumerator **LV_PROPERTY_OBJ_SCROLL_X**

enumerator **LV_PROPERTY_OBJ_SCROLL_Y**

enumerator **LV_PROPERTY_OBJ_SCROLL_TOP**

enumerator **LV_PROPERTY_OBJ_SCROLL_BOTTOM**

enumerator **LV_PROPERTY_OBJ_SCROLL_LEFT**

enumerator **LV_PROPERTY_OBJ_SCROLL_RIGHT**

enumerator **LV_PROPERTY_OBJ_SCROLL_END**

enumerator **LV_PROPERTY_OBJ_EXT_DRAW_SIZE**

enumerator **LV_PROPERTY_OBJ_EVENT_COUNT**

enumerator **LV_PROPERTY_OBJ_SCREEN**

enumerator **LV_PROPERTY_OBJ_DISPLAY**

enumerator **LV_PROPERTY_OBJ_CHILD_COUNT**

enumerator **LV_PROPERTY_OBJ_INDEX**

enumerator **LV_PROPERTY_OBJ_END**

Funzioni

`lv_obj_t *lv_obj_create(lv_obj_t *parent)`

Create a base object (a rectangle)

Parametri

`parent` -- pointer to a parent object. If NULL then a screen will be created.

Ritorna

pointer to the new object

`void lv_obj_add_flag(lv_obj_t *obj, lv_obj_flag_t f)`

Set one or more flags

Parametri

- **obj** -- pointer to an object
- **f** -- OR-ed values from `lv_obj_flag_t` to set.

`void lv_obj_remove_flag(lv_obj_t *obj, lv_obj_flag_t f)`

Remove one or more flags

Parametri

- **obj** -- pointer to an object
- **f** -- OR-ed values from `lv_obj_flag_t` to clear.

`void lv_obj_set_flag(lv_obj_t *obj, lv_obj_flag_t f, bool v)`

Set add or remove one or more flags.

Parametri

- **obj** -- pointer to an object
- **f** -- OR-ed values from `lv_obj_flag_t` to update.
- **v** -- true: add the flags; false: remove the flags

`void lv_obj_add_state(lv_obj_t *obj, lv_state_t state)`

Add one or more states to the object. The other state bits will remain unchanged. If specified in the styles, transition animation will be started from the previous state to the current.

Parametri

- **obj** -- pointer to an object
- **state** -- the states to add. E.g `LV_STATE_PRESSED | LV_STATE_FOCUSED`

`void lv_obj_remove_state(lv_obj_t *obj, lv_state_t state)`

Remove one or more states to the object. The other state bits will remain unchanged. If specified in the styles, transition animation will be started from the previous state to the current.

Parametri

- **obj** -- pointer to an object
- **state** -- the states to add. E.g `LV_STATE_PRESSED | LV_STATE_FOCUSED`

`void lv_obj_set_state(lv_obj_t *obj, lv_state_t state, bool v)`

Add or remove one or more states to the object. The other state bits will remain unchanged.

Parametri

- **obj** -- pointer to an object
- **state** -- the states to add. E.g `LV_STATE_PRESSED | LV_STATE_FOCUSED`
- **v** -- true: add the states; false: remove the states

`void lv_obj_set_user_data(lv_obj_t *obj, void *user_data)`

Set the user_data field of the object

Parametri

- **obj** -- pointer to an object
- **user_data** -- pointer to the new user_data.

`bool lv_obj_has_flag(const lv_obj_t *obj, lv_obj_flag_t f)`

Check if a given flag or all the given flags are set on an object.

Parametri

- **obj** -- pointer to an object
- **f** -- the flag(s) to check (OR-ed values can be used)

Ritorna

true: all flags are set; false: not all flags are set

`bool lv_obj_has_flag_any(const lv_obj_t *obj, lv_obj_flag_t f)`

Check if a given flag or any of the flags are set on an object.

Parametri

- **obj** -- pointer to an object
- **f** -- the flag(s) to check (OR-ed values can be used)

Ritorna

true: at least one flag is set; false: none of the flags are set

`lv_state_t lv_obj_get_state(const lv_obj_t *obj)`

Get the state of an object

Parametri

- **obj** -- pointer to an object

Ritorna

the state (OR-ed values from `lv_state_t`)

`bool lv_obj_has_state(const lv_obj_t *obj, lv_state_t state)`

Check if the object is in a given state or not.

Parametri

- **obj** -- pointer to an object
- **state** -- a state or combination of states to check

Ritorna

true: obj is in state; false: obj is not in state

`lv_group_t *lv_obj_get_group(const lv_obj_t *obj)`

Get the group of the object

Parametri

- **obj** -- pointer to an object

Ritorna

the pointer to group of the object

`void *lv_obj_get_user_data(lv_obj_t *obj)`

Get the user_data field of the object

Parametri

- **obj** -- pointer to an object

Ritorna

the pointer to the user_data of the object

`void lv_obj_allocate_spec_attr(lv_obj_t *obj)`

Allocate special data for an object if not allocated yet.

Parametri

- **obj** -- pointer to an object

`bool lv_obj_check_type(const lv_obj_t *obj, const lv_obj_class_t *class_p)`

Check the type of obj.

Parametri

- **obj** -- pointer to an object
- **class_p** -- a class to check (e.g. `lv_slider_class`)

Ritorna

true: class_p is the obj class.

`bool lv_obj_has_class(const lv_obj_t *obj, const lv_obj_class_t *class_p)`

Check if any object has a given class (type). It checks the ancestor classes too.

Parametri

- **obj** -- pointer to an object
- **class_p** -- a class to check (e.g. `lv_slider_class`)

Ritorna

true: obj has the given class

`const lv_obj_class_t *lv_obj_get_class(const lv_obj_t *obj)`

Get the class (type) of the object

Parametri

- **obj** -- pointer to an object

Ritorna

the class (type) of the object

`bool lv_obj_is_valid(const lv_obj_t *obj)`

Check if any object is still "alive".

Parametri

- **obj** -- pointer to an object

Ritorna

true: valid

`void lv_obj_null_on_delete(lv_obj_t **obj_ptr)`

Utility to set an object reference to NULL when it gets deleted. The reference should be in a location that will not become invalid during the object's lifetime, i.e. static or allocated.

Parametri

- **obj_ptr** -- a pointer to a pointer to an object

`void lv_obj_add_screen_load_event(lv_obj_t *obj, lv_event_code_t trigger, lv_obj_t *screen,`

`lv_screen_load_anim_t anim_type, uint32_t duration, uint32_t delay)`

Add an event handler to a widget that will load a screen on a trigger.

Parametri

- **obj** -- pointer to widget which should load the screen
- **trigger** -- an event code, e.g. LV_EVENT_CLICKED
- **screen** -- the screen to load (must be a valid widget)
- **anim_type** -- element of `lv_screen_load_anim_t` the screen load animation
- **duration** -- duration of the animation in milliseconds
- **delay** -- delay before the screen load in milliseconds

`void lv_obj_add_screen_create_event(lv_obj_t *obj, lv_event_code_t trigger, lv_screen_create_cb_t`

`screen_create_cb, lv_screen_load_anim_t anim_type, uint32_t duration,`
`uint32_t delay)`

Add an event handler to a widget that will create a screen on a trigger. The created screen will be deleted when it's unloaded

Parametri

- **obj** -- pointer to widget which should load the screen
- **trigger** -- an event code, e.g. LV_EVENT_CLICKED
- **screen_create_cb** -- a callback to create the screen, e.g. `lv_obj_t *myscreen_create(void)`
- **anim_type** -- element of `lv_screen_load_anim_t` the screen load animation
- **duration** -- duration of the animation in milliseconds
- **delay** -- delay before the screen load in milliseconds

`void lv_obj_add_play_timeline_event(lv_obj_t *obj, lv_event_code_t trigger, lv_anim_timeline_t *at, uint32_t`
`delay, bool reverse)`

Play a timeline animation on a trigger

Parametri

- **obj** -- pointer to widget which should trigger playing the animation
- **trigger** -- an event code, e.g. LV_EVENT_CLICKED
- **at** -- pointer to an animation timeline
- **delay** -- wait time before starting the animation
- **reverse** -- true: play in reverse

`void lv_obj_set_id(lv_obj_t *obj, void *id)`

Set an id for an object.

Parametri

- **obj** -- pointer to an object
- **id** -- the id of the object

`void *lv_obj_get_id(const lv_obj_t *obj)`

Get the id of an object.

Parametri

- **obj** -- pointer to an object

Ritorna

the id of the object

```
lv_obj_t *lv_obj_find_by_id(const lv_obj_t *obj, const void *id)
```

DEPRECATED IDs are used only to print the widget trees. To find a widget use `lv_obj_find_by_name`

Get the child object by its id. It will check children and grandchildren recursively. Function `lv_obj_id_compare` is used to matched obj id with given id.

Parametri

- **obj** -- pointer to an object
- **id** -- the id of the child object

Ritorna

pointer to the child object or NULL if not found

```
void lv_obj_assign_id(const lv_obj_class_t *class_p, lv_obj_t *obj)
```

Assign id to object if not previously assigned. This function gets called automatically when `LV_OBJ_ID_AUTO_ASSIGN` is enabled.

Set `LV_USE_OBJ_ID_BUILTIN` to use the builtin method to generate object ID. Otherwise, these functions including `lv_obj_[set|assign|free|stringify]_id` and `lv_obj_id_compare` should be implemented externally.

Parametri

- **class_p** -- the class this obj belongs to. Note `obj->class_p` is the class currently being constructed.
- **obj** -- pointer to an object

```
void lv_obj_free_id(lv_obj_t *obj)
```

Free resources allocated by `lv_obj_assign_id` or `lv_obj_set_id`. This function is also called automatically when object is deleted.

Parametri

- **obj** -- pointer to an object

```
int lv_obj_id_compare(const void *id1, const void *id2)
```

Compare two obj id, return 0 if they are equal.

Set `LV_USE_OBJ_ID_BUILTIN` to use the builtin method for compare. Otherwise, it must be implemented externally.

Parametri

- **id1** -- the first id
- **id2** -- the second id

Ritorna

0 if they are equal, non-zero otherwise.

```
const char *lv_obj_stringify_id(lv_obj_t *obj, char *buf, uint32_t len)
```

Format an object's id into a string.

Parametri

- **obj** -- pointer to an object
- **buf** -- buffer to write the string into
- **len** -- length of the buffer

```
void lv_objid_builtin_destroy(void)
```

Free resources used by builtin ID generator.

Variabili

```
const lv_obj_class_t lv_obj_class
```

Make the base object's class publicly available.

14.12.5 lv_obj_class.h

Typedef

```
typedef void (*lv_obj_class_event_cb_t)(lv_obj_class_t *class_p, lv_event_t *e)
```

Enum

enum **lv_obj_class_editable_t**

Values:

enumerator **LV_OBJ_CLASS_EDITABLE_INHERIT**

Check the base class. Must have 0 value to let zero initialized class inherit

enumerator **LV_OBJ_CLASS_EDITABLE_TRUE**

enumerator **LV_OBJ_CLASS_EDITABLE_FALSE**

enum **lv_obj_class_group_def_t**

Values:

enumerator **LV_OBJ_CLASS_GROUP_DEF_INHERIT**

Check the base class. Must have 0 value to let zero initialized class inherit

enumerator **LV_OBJ_CLASS_GROUP_DEF_TRUE**

enumerator **LV_OBJ_CLASS_GROUP_DEF_FALSE**

enum **lv_obj_class_theme_inheritable_t**

Values:

enumerator **LV_OBJ_CLASS_THEME_INHERITABLE_FALSE**

Do not inherit theme from base class.

enumerator **LV_OBJ_CLASS_THEME_INHERITABLE_TRUE**

Funzioni

lv_obj_t *lv_obj_class_create_obj(const lv_obj_class_t *class_p, lv_obj_t *parent)

Create an object form a class descriptor

Parametri

- **class_p** -- pointer to a class
- **parent** -- pointer to an object where the new object should be created

Ritorna

pointer to the created object

void lv_obj_class_init_obj(lv_obj_t *obj)

bool lv_obj_is_editable(lv_obj_t *obj)

bool lv_obj_is_group_def(lv_obj_t *obj)

14.12.6 lv_obj_class_private.h

Funzioni

void lv_obj_destruct(lv_obj_t *obj)

struct **_lv_obj_class_t**

#include <lv_obj_class_private.h> Describe the common methods of every object. Similar to a C++ class.

Membri Pubblici

`const lv_obj_class_t *base_class`

`void (*constructor_cb)(const lv_obj_class_t *class_p, lv_obj_t *obj)`

class_p is the final class while obj->class_p is the class currently being [de]constructed.

`void (*destructor_cb)(const lv_obj_class_t *class_p, lv_obj_t *obj)`

`void (*event_cb)(const lv_obj_class_t *class_p, lv_event_t *e)`

class_p is the class in which event is being processed. Widget type specific event function

`uint32_t prop_index_start`

`uint32_t prop_index_end`

`const lv_property_ops_t *properties`

`uint32_t properties_count`

`const lv_property_name_t *property_names`

`uint32_t names_count`

`void *user_data`

`const char *name`

`int32_t width_def`

`int32_t height_def`

`uint32_t editable`

Value from `lv_obj_class_editable_t`

`uint32_t group_def`

Value from `lv_obj_class_group_def_t`

`uint32_t instance_size`

`uint32_t theme_inheritable`

Value from `lv_obj_class_theme_inheritable_t`

14.12.7 lv_obj_draw.h

Enum

enum lv_layer_type_t

Store the type of layer required to render a widget.

Values:

enumerator LV_LAYER_TYPE_NONE

No layer is needed.

enumerator LV_LAYER_TYPE_SIMPLE

Simple layer means that the layer can be rendered in chunks. For example with `opa_layered = 140` it's possible to render only 10 lines from the layer. When it's ready go to the next 10 lines. It avoids large memory allocations for the layer buffer. The buffer size for a chunk can be set by `LV_DRAW_LAYER_SIMPLE_BUF_SIZE` in `lv_conf.h`.

enumerator LV_LAYER_TYPE_TRANSFORM

The widget is transformed and cannot be rendered in chunks. It's because - due to the transformations - pixel outside of a given area will also contribute to the final image. In this case there is no limitation on the buffer size. LVGL will allocate as large buffer as needed to render the transformed area.

Funzioni

void lv_obj_init_draw_rect_dsc(lv_obj_t *obj, lv_part_t part, lv_draw_rect_dsc_t *draw_dsc)

Initialize a rectangle draw descriptor from an object's styles in its current state

Nota

Only the relevant fields will be set. E.g. if `border width == 0` the other border properties won't be evaluated.

Parametri

- **obj** -- pointer to an object
- **part** -- part of the object, e.g. `LV_PART_MAIN`, `LV_PART_SCROLLBAR`, `LV_PART_KNOB`, etc
- **draw_dsc** -- the descriptor to initialize. If an `..._opa` field is set to `LV_OPA_TRANSP` the related properties won't be initialized. Should be initialized with `lv_draw_rect_dsc_init(draw_dsc)`.

void lv_obj_init_draw_label_dsc(lv_obj_t *obj, lv_part_t part, lv_draw_label_dsc_t *draw_dsc)

Initialize a label draw descriptor from an object's styles in its current state

Parametri

- **obj** -- pointer to an object
- **part** -- part of the object, e.g. `LV_PART_MAIN`, `LV_PART_SCROLLBAR`, `LV_PART_KNOB`, etc
- **draw_dsc** -- the descriptor to initialize. If the `opa` field is set to or the property is equal to `LV_OPA_TRANSP` the rest won't be initialized. Should be initialized with `lv_draw_label_dsc_init(draw_dsc)`.

void lv_obj_init_draw_image_dsc(lv_obj_t *obj, lv_part_t part, lv_draw_image_dsc_t *draw_dsc)

Initialize an image draw descriptor from an object's styles in its current state

Parametri

- **obj** -- pointer to an object
- **part** -- part of the object, e.g. `LV_PART_MAIN`, `LV_PART_SCROLLBAR`, `LV_PART_KNOB`, etc
- **draw_dsc** -- the descriptor to initialize. Should be initialized with `lv_draw_image_dsc_init(draw_dsc)`.

`void lv_obj_init_draw_line_dsc(lv_obj_t *obj, lv_part_t part, lv_draw_line_dsc_t *draw_dsc)`

Initialize a line draw descriptor from an object's styles in its current state

Parametri

- **obj** -- pointer to an object
- **part** -- part of the object, e.g. LV_PART_MAIN, LV_PART_SCROLLBAR, LV_PART_KNOB, etc
- **draw_dsc** -- the descriptor to initialize. Should be initialized with `lv_draw_line_dsc_init(draw_dsc)`.

`void lv_obj_init_draw_arc_dsc(lv_obj_t *obj, lv_part_t part, lv_draw_arc_dsc_t *draw_dsc)`

Initialize an arc draw descriptor from an object's styles in its current state

Parametri

- **obj** -- pointer to an object
- **part** -- part of the object, e.g. LV_PART_MAIN, LV_PART_SCROLLBAR, LV_PART_KNOB, etc
- **draw_dsc** -- the descriptor to initialize. Should be initialized with `lv_draw_arc_dsc_init(draw_dsc)`.

`int32_t lv_obj_calculate_ext_draw_size(lv_obj_t *obj, lv_part_t part)`

Get the required extra size (around the object's part) to draw shadow, outline, value etc.

Parametri

- **obj** -- pointer to an object
- **part** -- part of the object

Ritorna

the extra size required around the object

`void lv_obj_refresh_ext_draw_size(lv_obj_t *obj)`

Send a 'LV_EVENT_REFR_EXT_DRAW_SIZE' Call the ancestor's event handler to the object to refresh the value of the extended draw size. The result will be saved in obj.

Parametri

- **obj** -- pointer to an object

14.12.8 lv_obj_draw_private.h

Funzioni

`int32_t lv_obj_get_ext_draw_size(const lv_obj_t *obj)`

Get the extended draw area of an object.

Parametri

- **obj** -- pointer to an object

Ritorna

the size extended draw area around the real coordinates

`lv_layer_type_t lv_obj_get_layer_type(const lv_obj_t *obj)`

14.12.9 lv_obj_event.h

Enum

enum `lv_cover_res_t`

Cover check results.

Values:

enumerator `LV_COVER_RES_COVER`

enumerator `LV_COVER_RES_NOT_COVER`

enumerator **LV_COVER_RES_MASKED**

Funzioni

lv_result_t lv_obj_send_event(lv_obj_t *obj, lv_event_code_t event_code, void *param)

Send an event to the object

Parametri

- **obj** -- pointer to an object
- **event_code** -- the type of the event from `lv_event_t`
- **param** -- arbitrary data depending on the widget type and the event. (Usually NULL)

Ritorna

LV_RESULT_OK: obj was not deleted in the event; LV_RESULT_INVALID: obj was deleted in the event_code

lv_result_t lv_obj_event_base(const lv_obj_class_t *class_p, lv_event_t *e)

Used by the widgets internally to call the ancestor widget types's event handler

Parametri

- **class_p** -- pointer to the class of the widget (NOT the ancestor class)
- **e** -- pointer to the event descriptor

Ritorna

LV_RESULT_OK: the target object was not deleted in the event; LV_RESULT_INVALID: it was deleted in the event_code

lv_obj_t *lv_event_get_current_target_obj(lv_event_t *e)

Get the current target of the event. It's the object which event handler being called. If the event is not bubbled it's the same as "original" target.

Parametri

- **e** -- pointer to the event descriptor

Ritorna

the target of the event_code

lv_obj_t *lv_event_get_target_obj(lv_event_t *e)

Get the object originally targeted by the event. It's the same even if the event is bubbled.

Parametri

- **e** -- pointer to the event descriptor

Ritorna

pointer to the original target of the event_code

lv_event_dsc_t *lv_obj_add_event_cb(lv_obj_t *obj, lv_event_cb_t event_cb, lv_event_code_t filter, void *user_data)

Add an event handler function for an object. Used by the user to react on event which happens with the object. An object can have multiple event handler. They will be called in the same order as they were added.

Parametri

- **obj** -- pointer to an object
- **filter** -- an event code (e.g. `LV_EVENT_CLICKED`) on which the event should be called. `LV_EVENT_ALL` can be used to receive all the events.
- **event_cb** -- the new event function
- **user_data** -- custom data will be available in event_cb

Ritorna

handler to the event. It can be used in `lv_obj_remove_event_dsc`.

uint32_t lv_obj_get_event_count(lv_obj_t *obj)

lv_event_dsc_t *lv_obj_get_event_dsc(lv_obj_t *obj, uint32_t index)

bool lv_obj_remove_event(lv_obj_t *obj, uint32_t index)

bool lv_obj_remove_event_dsc(lv_obj_t *obj, lv_event_dsc_t *dsc)

`uint32_t lv_obj_remove_event_cb(lv_obj_t *obj, lv_event_cb_t event_cb)`

Remove an event_cb from an object

Parametri

- **obj** -- pointer to a obj
- **event_cb** -- the event_cb of the event to remove

Ritorna

the count of the event removed

`uint32_t lv_obj_remove_event_cb_with_user_data(lv_obj_t *obj, lv_event_cb_t event_cb, void *user_data)`

Remove an event_cb with user_data

Parametri

- **obj** -- pointer to a obj
- **event_cb** -- the event_cb of the event to remove
- **user_data** -- user_data

Ritorna

the count of the event removed

`lv_indev_t *lv_event_get_indev(lv_event_t *e)`

Get the input device passed as parameter to indev related events.

Parametri

- **e** -- pointer to an event

Ritorna

the indev that triggered the event or NULL if called on a not indev related event

`lv_layer_t *lv_event_get_layer(lv_event_t *e)`

Get the draw context which should be the first parameter of the draw functions. Namely:
`LV_EVENT_DRAW_MAIN/POST, LV_EVENT_DRAW_MAIN/POST_BEGIN, LV_EVENT_DRAW_MAIN/POST_END`

Parametri

- **e** -- pointer to an event

Ritorna

pointer to a draw context or NULL if called on an unrelated event

`const lv_area_t *lv_event_get_old_size(lv_event_t *e)`

Get the old area of the object before its size was changed. Can be used in `LV_EVENT_SIZE_CHANGED`

Parametri

- **e** -- pointer to an event

Ritorna

the old absolute area of the object or NULL if called on an unrelated event

`uint32_t lv_event_get_key(lv_event_t *e)`

Get the key passed as parameter to an event. Can be used in `LV_EVENT_KEY`

Parametri

- **e** -- pointer to an event

Ritorna

the triggering key or NULL if called on an unrelated event

`int32_t lv_event_get_rotary_diff(lv_event_t *e)`

Get the signed rotary encoder diff. passed as parameter to an event. Can be used in `LV_EVENT_ROTARY`

Parametri

- **e** -- pointer to an event

Ritorna

the triggering key or NULL if called on an unrelated event

`lv_anim_t *lv_event_get_scroll_anim(lv_event_t *e)`

Get the animation descriptor of a scrolling. Can be used in `LV_EVENT_SCROLL_BEGIN`

Parametri

- **e** -- pointer to an event

Ritorna

the animation that will scroll the object. (can be modified as required)

```
void lv_event_set_ext_draw_size(lv_event_t *e, int32_t size)
```

Set the new extra draw size. Can be used in `LV_EVENT_REFR_EXT_DRAW_SIZE`

Parametri

- `e` -- pointer to an event
- `size` -- The new extra draw size

```
lv_point_t *lv_event_get_self_size_info(lv_event_t *e)
```

Get a pointer to an `lv_point_t` variable in which the self size should be saved (width in `point->x` and height `point->y`). Can be used in `LV_EVENT_GET_SELF_SIZE`

Parametri

- `e` -- pointer to an event

Ritorna

pointer to `lv_point_t` or NULL if called on an unrelated event

```
lv_hit_test_info_t *lv_event_get_hit_test_info(lv_event_t *e)
```

Get a pointer to an `lv_hit_test_info_t` variable in which the hit test result should be saved. Can be used in `LV_EVENT_HIT_TEST`

Parametri

- `e` -- pointer to an event

Ritorna

pointer to `lv_hit_test_info_t` or NULL if called on an unrelated event

```
const lv_area_t *lv_event_get_cover_area(lv_event_t *e)
```

Get a pointer to an area which should be examined whether the object fully covers it or not. Can be used in `LV_EVENT_HIT_TEST`

Parametri

- `e` -- pointer to an event

Ritorna

an area with absolute coordinates to check

```
void lv_event_set_cover_res(lv_event_t *e, lv_cover_res_t res)
```

Set the result of cover checking. Can be used in `LV_EVENT_COVER_CHECK`

Parametri

- `e` -- pointer to an event
- `res` -- an element of `::lv_cover_check_info_t`

```
lv_draw_task_t *lv_event_get_draw_task(lv_event_t *e)
```

Get the draw task which was just added. Can be used in `LV_EVENT_DRAW_TASK_ADDED` event

Parametri

- `e` -- pointer to an event

Ritorna

the added draw task

14.12.10 lv_obj_event_private.h

```
struct _lv_hit_test_info_t
```

#include <lv_obj_event_private.h> Used as the event parameter of `LV_EVENT_HIT_TEST` to check if an point can click the object or not. `res` should be set like this:

- If already set to `false` another event wants that point non clickable. If you want to respect it leave it as `false` or set `true` to overwrite it.
- If already set `true` and point shouldn't be clickable set to `false`
- If already set to `true` you agree that point can click the object leave it as `true`

Membri Pubblici

```
const lv_point_t *point
```

A point relative to screen to check if it can click the object or not

```
bool res
    true: point can click the object; false: it cannot

struct _lv_cover_check_info_t
#include <lv_obj_event_private.h> Used as the event parameter of LV\_EVENT\_COVER\_CHECK to
check if an area is covered by the object or not. In the event use const lv\_area\_t * area =
lv\_event\_get\_cover\_area\(e\) to get the area to check and lv\_event\_set\_cover\_res\(e, res\) to set the
result.
```

Membri Pubblici

[lv_cover_res_t](#) res

const [lv_area_t](#) *area

14.12.11 lv_obj_pos.h

Enum

enum [lv_obj_point_transform_flag_t](#)

Values:

enumerator **LV_OBJ_POINT_TRANSFORM_FLAG_NONE**

No flags

enumerator **LV_OBJ_POINT_TRANSFORM_FLAG_RECURSIVE**

Consider the transformation properties of the parents too

enumerator **LV_OBJ_POINT_TRANSFORM_FLAG_INVERSE**

Execute the inverse of the transformation (-angle and 1/zoom)

enumerator **LV_OBJ_POINT_TRANSFORM_FLAG_INVERSE_RECURSIVE**

Both inverse and recursive

Funzioni

void [lv_obj_set_pos\(lv_obj_t *obj, int32_t x, int32_t y\)](#)

Set the position of an object relative to the set alignment.

Nota

With default alignment it's the distance from the top left corner

Nota

E.g. [LV_ALIGN_CENTER](#) alignment it's the offset from the center of the parent

i Nota

The position is interpreted on the content area of the parent

i Nota

The values can be set in pixel or in percentage of parent size with `lv_pct(v)`

Parametri

- **obj** -- pointer to an object
- **x** -- new x coordinate
- **y** -- new y coordinate

`void lv_obj_set_x(lv_obj_t *obj, int32_t x)`

Set the x coordinate of an object

i Nota

With default alignment it's the distance from the top left corner

i Nota

E.g. `LV_ALIGN_CENTER` alignment it's the offset from the center of the parent

i Nota

The position is interpreted on the content area of the parent

i Nota

The values can be set in pixel or in percentage of parent size with `lv_pct(v)`

Parametri

- **obj** -- pointer to an object
- **x** -- new x coordinate

`void lv_obj_set_y(lv_obj_t *obj, int32_t y)`

Set the y coordinate of an object

i Nota

With default alignment it's the distance from the top left corner

i Nota

E.g. `LV_ALIGN_CENTER` alignment it's the offset from the center of the parent

i Nota

The position is interpreted on the content area of the parent

i Nota

The values can be set in pixel or in percentage of parent size with `lv_pct(v)`

Parametri

- **obj** -- pointer to an object
- **y** -- new y coordinate

`void lv_obj_set_size(lv_obj_t *obj, int32_t w, int32_t h)`

Set the size of an object.

i Nota

possible values are: pixel simple set the size accordingly LV_SIZE_CONTENT set the size to involve all children in the given direction `lv_pct(x)` to set size in percentage of the parent's content area size (the size without paddings). x should be in [0..1000)% range

Parametri

- **obj** -- pointer to an object
- **w** -- the new width
- **h** -- the new height

`bool lv_obj_refr_size(lv_obj_t *obj)`

Recalculate the size of the object

Parametri

- **obj** -- pointer to an object

Ritorna

true: the size has been changed

`void lv_obj_set_width(lv_obj_t *obj, int32_t w)`

Set the width of an object

i Nota

possible values are: pixel simple set the size accordingly LV_SIZE_CONTENT set the size to involve all children in the given direction `lv_pct(x)` to set size in percentage of the parent's content area size (the size without paddings). x should be in [0..1000)% range

Parametri

- **obj** -- pointer to an object
- **w** -- the new width

`void lv_obj_set_height(lv_obj_t *obj, int32_t h)`

Set the height of an object

i Nota

possible values are: pixel simple set the size accordingly LV_SIZE_CONTENT set the size to involve all children in the given direction `lv_pct(x)` to set size in percentage of the parent's content area size (the size without paddings). x should be in [0..1000)% range

Parametri

- **obj** -- pointer to an object
- **h** -- the new height

void lv_obj_set_content_width(lv_obj_t *obj, int32_t w)

Set the width reduced by the left and right padding and the border width.

Parametri

- **obj** -- pointer to an object
- **w** -- the width without paddings in pixels

void lv_obj_set_content_height(lv_obj_t *obj, int32_t h)

Set the height reduced by the top and bottom padding and the border width.

Parametri

- **obj** -- pointer to an object
- **h** -- the height without paddings in pixels

void lv_obj_set_layout(lv_obj_t *obj, uint32_t layout)

Set a layout for an object

Parametri

- **obj** -- pointer to an object
- **layout** -- pointer to a layout descriptor to set

bool lv_obj_is_layout_positioned(const lv_obj_t *obj)

Test whether the object is positioned by a layout or not

Parametri

- **obj** -- pointer to an object to test

Ritorna

true: positioned by a layout; false: not positioned by a layout

void lv_obj_mark_layout_as_dirty(lv_obj_t *obj)

Mark the object for layout update.

Parametri

- **obj** -- pointer to an object whose children need to be updated

void lv_obj_request_layout_complete_event(lv_obj_t *obj)

Mark screen to send layout completed event after update.

Parametri

- **obj** -- Any object on the target screen

void lv_obj_update_layout(const lv_obj_t *obj)

Update the layout of an object.

Parametri

- **obj** -- pointer to an object whose position and size needs to be updated

void lv_obj_set_align(lv_obj_t *obj, lv_align_t align)

Change the alignment of an object.

Parametri

- **obj** -- pointer to an object to align
- **align** -- type of alignment (see '*lv_align_t*' enum) LV_ALIGN_OUT_... can't be used.

void lv_obj_align(lv_obj_t *obj, lv_align_t align, int32_t x_ofs, int32_t y_ofs)

Change the alignment of an object and set new coordinates. Equivalent to: `lv_obj_set_align(obj, align); lv_obj_set_pos(obj, x_ofs, y_ofs);`

Parametri

- **obj** -- pointer to an object to align
- **align** -- type of alignment (see '*lv_align_t*' enum) LV_ALIGN_OUT_... can't be used.
- **x_ofs** -- x coordinate offset after alignment
- **y_ofs** -- y coordinate offset after alignment

`void lv_obj_align_to(lv_obj_t *obj, const lv_obj_t *base, lv_align_t align, int32_t x_ofs, int32_t y_ofs)`
 Align an object to another object.

Nota

if the position or size of `base` changes `obj` needs to be aligned manually again

Parametri

- `obj` -- pointer to an object to align
- `base` -- pointer to another object (if NULL `obj`'s parent is used). '`obj`' will be aligned to it.
- `align` -- type of alignment (see '`lv_align_t`' enum)
- `x_ofs` -- x coordinate offset after alignment
- `y_ofs` -- y coordinate offset after alignment

`void lv_obj_center(lv_obj_t *obj)`

Align an object to the center on its parent.

Nota

if the parent size changes `obj` needs to be aligned manually again

Parametri

`obj` -- pointer to an object to align

`void lv_obj_set_transform(lv_obj_t *obj, const lv_matrix_t *matrix)`

Set the transform matrix of an object

Nota

`LV_DRAW_TRANSFORM_USE_MATRIX` needs to be enabled.

Parametri

- `obj` -- pointer to an object
- `matrix` -- pointer to a matrix to set

`void lv_obj_reset_transform(lv_obj_t *obj)`

Reset the transform matrix of an object to identity matrix

Nota

`LV_DRAW_TRANSFORM_USE_MATRIX` needs to be enabled.

Parametri

`obj` -- pointer to an object

`void lv_obj_get_coords(const lv_obj_t *obj, lv_area_t *coords)`

Copy the coordinates of an object to an area

Parametri

- `obj` -- pointer to an object
- `coords` -- pointer to an area to store the coordinates

`int32_t lv_obj_get_x(const lv_obj_t *obj)`

Get the x coordinate of object.

Nota

The position of the object is recalculated only on the next redraw. To force coordinate recalculation call `lv_obj_update_layout(obj)`.

Nota

Zero return value means the object is on the left padding of the parent, and not on the left edge.

Nota

Scrolling of the parent doesn't change the returned value.

Nota

The returned value is always the distance from the parent even if `obj` is positioned by a layout.

Parametri

`obj` -- pointer to an object

Ritorna

distance of `obj` from the left side of its parent plus the parent's left padding

```
int32_t lv_obj_get_x2(const lv_obj_t *obj)
```

Get the x2 coordinate of object.

Nota

The position of the object is recalculated only on the next redraw. To force coordinate recalculation call `lv_obj_update_layout(obj)`.

Nota

Zero return value means the object is on the right padding of the parent, and not on the right edge.

Nota

Scrolling of the parent doesn't change the returned value.

Nota

The returned value is always the distance from the parent even if `obj` is positioned by a layout.

Parametri

`obj` -- pointer to an object

Ritorna

distance of `obj` from the right side of its parent plus the parent's right padding

```
int32_t lv_obj_get_y(const lv_obj_t *obj)
```

Get the y coordinate of object.

Nota

The position of the object is recalculated only on the next redraw. To force coordinate recalculation call `lv_obj_update_layout(obj)`.

Nota

Zero return value means the object is on the top padding of the parent, and not on the top edge.

Nota

Scrolling of the parent doesn't change the returned value.

Nota

The returned value is always the distance from the parent even if `obj` is positioned by a layout.

Parametri

`obj` -- pointer to an object

Ritorna

distance of `obj` from the top side of its parent plus the parent's top padding

```
int32_t lv_obj_get_y2(const lv_obj_t *obj)
```

Get the y2 coordinate of object.

Nota

The position of the object is recalculated only on the next redraw. To force coordinate recalculation call `lv_obj_update_layout(obj)`.

Nota

Zero return value means the object is on the bottom padding of the parent, and not on the bottom edge.

Nota

Scrolling of the parent doesn't change the returned value.

Nota

The returned value is always the distance from the parent even if `obj` is positioned by a layout.

Parametri

`obj` -- pointer to an object

Ritorna

distance of `obj` from the bottom side of its parent plus the parent's bottom padding

```
int32_t lv_obj_get_x_aligned(const lv_obj_t *obj)
```

Get the actually set x coordinate of object, i.e. the offset from the set alignment

Parametri

`obj` -- pointer to an object

Ritorna

the set x coordinate

```
int32_t lv_obj_get_y_aligned(const lv_obj_t *obj)
```

Get the actually set y coordinate of object, i.e. the offset from the set alignment

Parametri

`obj` -- pointer to an object

Ritorna

the set y coordinate

```
int32_t lv_obj_get_width(const lv_obj_t *obj)
```

Get the width of an object

Nota

The position of the object is recalculated only on the next redraw. To force coordinate recalculation call `lv_obj_update_layout(obj)`.

Parametri

`obj` -- pointer to an object

Ritorna

the width in pixels

```
int32_t lv_obj_get_height(const lv_obj_t *obj)
```

Get the height of an object

Nota

The position of the object is recalculated only on the next redraw. To force coordinate recalculation call `lv_obj_update_layout(obj)`.

Parametri

`obj` -- pointer to an object

Ritorna

the height in pixels

```
int32_t lv_obj_get_content_width(const lv_obj_t *obj)
```

Get the width reduced by the left and right padding and the border width.

Nota

The position of the object is recalculated only on the next redraw. To force coordinate recalculation call `lv_obj_update_layout(obj)`.

Parametri

`obj` -- pointer to an object

Ritorna

the width which still fits into its parent without causing overflow (making the parent scrollable)

```
int32_t lv_obj_get_content_height(const lv_obj_t *obj)
```

Get the height reduced by the top and bottom padding and the border width.

Nota

The position of the object is recalculated only on the next redraw. To force coordinate recalculation call `lv_obj_update_layout(obj)`.

Parametri

- **obj** -- pointer to an object

Ritorna

the height which still fits into the parent without causing overflow (making the parent scrollable)

```
void lv_obj_get_content_coords(const lv_obj_t *obj, lv_area_t *area)
```

Get the area reduced by the paddings and the border width.

Nota

The position of the object is recalculated only on the next redraw. To force coordinate recalculation call `lv_obj_update_layout(obj)`.

Parametri

- **obj** -- pointer to an object
- **area** -- the area which still fits into the parent without causing overflow (making the parent scrollable)

```
int32_t lv_obj_get_self_width(const lv_obj_t *obj)
```

Get the width occupied by the "parts" of the widget. E.g. the width of all columns of a table.

Nota

This size independent from the real size of the widget. It just tells how large the internal ("virtual") content is.

Parametri

- **obj** -- pointer to an objects

Ritorna

the width of the virtually drawn content

```
int32_t lv_obj_get_self_height(const lv_obj_t *obj)
```

Get the height occupied by the "parts" of the widget. E.g. the height of all rows of a table.

Nota

This size independent from the real size of the widget. It just tells how large the internal ("virtual") content is.

Parametri

- **obj** -- pointer to an objects

Ritorna

the width of the virtually drawn content

```
bool lv_obj_refresh_self_size(lv_obj_t *obj)
```

Handle if the size of the internal ("virtual") content of an object has changed.

Parametri

- **obj** -- pointer to an object

Ritorna
false: nothing happened; true: refresh happened

```
void lv_obj_refr_pos(lv_obj_t *obj)
```

```
void lv_obj_move_to(lv_obj_t *obj, int32_t x, int32_t y)
```

```
void lv_obj_move_children_by(lv_obj_t *obj, int32_t x_diff, int32_t y_diff, bool ignore_floating)
```

```
const lv_matrix_t *lv_obj_get_transform(const lv_obj_t *obj)
```

Get the transform matrix of an object

Parametri

- obj** -- pointer to an object

Ritorna
pointer to the transform matrix or NULL if not set

```
void lv_obj_transform_point(const lv_obj_t *obj, lv_point_t *p, lv_obj_point_transform_flag_t flags)
```

Transform a point using the angle and zoom style properties of an object

Parametri

- **obj** -- pointer to an object whose style properties should be used
- **p** -- a point to transform, the result will be written back here too
- **flags** -- OR-ed valued of :cpp:enum:*lv_obj_point_transform_flag_t*

```
void lv_obj_transform_point_array(const lv_obj_t *obj, lv_point_t points[], size_t count, lv_obj_point_transform_flag_t flags)
```

Transform an array of points using the angle and zoom style properties of an object

Parametri

- **obj** -- pointer to an object whose style properties should be used
- **points** -- the array of points to transform, the result will be written back here too
- **count** -- number of points in the array
- **flags** -- OR-ed valued of :cpp:enum:*lv_obj_point_transform_flag_t*

```
void lv_obj_get_transformed_area(const lv_obj_t *obj, lv_area_t *area, lv_obj_point_transform_flag_t flags)
```

Transform an area using the angle and zoom style properties of an object

Parametri

- **obj** -- pointer to an object whose style properties should be used
- **area** -- an area to transform, the result will be written back here too
- **flags** -- OR-ed valued of :cpp:enum:*lv_obj_point_transform_flag_t*

```
void lv_obj_invalidate_area(const lv_obj_t *obj, const lv_area_t *area)
```

Mark an area of an object as invalid. The area will be truncated to the object's area and marked for redraw.

Parametri

- **obj** -- pointer to an object
- **area** -- the area to redraw

```
void lv_obj_invalidate(const lv_obj_t *obj)
```

Mark the object as invalid to redrawn its area

Parametri

- obj** -- pointer to an object

```
bool lv_obj_area_is_visible(const lv_obj_t *obj, lv_area_t *area)
```

Tell whether an area of an object is visible (even partially) now or not

Parametri

- **obj** -- pointer to an object
- **area** -- the are to check. The visible part of the area will be written back here.

Ritorna
true visible; false not visible (hidden, out of parent, on other screen, etc)

```
bool lv_obj_is_visible(const lv_obj_t *obj)
```

Tell whether an object is visible (even partially) now or not

Parametri

- obj** -- pointer to an object

Ritorna

true: visible; false not visible (hidden, out of parent, on other screen, etc)

```
void lv_obj_set_ext_click_area(lv_obj_t *obj, int32_t size)
```

Set the size of an extended clickable area

Parametri

- **obj** -- pointer to an object
- **size** -- extended clickable area in all 4 directions [px]

```
void lv_obj_get_click_area(const lv_obj_t *obj, lv_area_t *area)
```

Get the an area where to object can be clicked. It's the object's normal area plus the extended click area.

Parametri

- **obj** -- pointer to an object
- **area** -- store the result area here

```
bool lv_obj_hit_test(lv_obj_t *obj, const lv_point_t *point)
```

Hit-test an object given a particular point in screen space.

Parametri

- **obj** -- object to hit-test
- **point** -- screen-space point (absolute coordinate)

Ritorna

true: if the object is considered under the point

```
int32_t lv_clamp_width(int32_t width, int32_t min_width, int32_t max_width, int32_t ref_width)
```

Clamp a width between min and max width. If the min/max width is in percentage value use the ref_width

Parametri

- **width** -- width to clamp
- **min_width** -- the minimal width
- **max_width** -- the maximal width
- **ref_width** -- the reference width used when min/max width is in percentage

Ritorna

the clamped width

```
int32_t lv_clamp_height(int32_t height, int32_t min_height, int32_t max_height, int32_t ref_height)
```

Clamp a height between min and max height. If the min/max height is in percentage value use the ref_height

Parametri

- **height** -- height to clamp
- **min_height** -- the minimal height
- **max_height** -- the maximal height
- **ref_height** -- the reference height used when min/max height is in percentage

Ritorna

the clamped height

14.12.12 lv_obj_private.h

```
struct _lv_obj_spec_attr_t
```

#include <lv_obj_private.h> Special, rarely used attributes. They are allocated automatically if any elements is set.

Membri Pubblici

lv_obj_t ****children**

Store the pointer of the children in an array.

lv_group_t ***group_p**

lv_event_list_t **event_list**

const char *name

Pointer to the name

lv_point_t **scroll**

The current X/Y scroll offset

int32_t ext_click_pad

Extra click padding in all direction

int32_t ext_draw_size

EXTend the size in every direction for drawing.

uint16_t child_cnt

Number of children

uint16_t scrollbar_mode

How to display scrollbars, see *lv_scrollbar_mode_t*

uint16_t scroll_snap_x

Where to align the snappable children horizontally, see *lv_scroll_snap_t*

uint16_t scroll_snap_y

Where to align the snappable children vertically

uint16_t scroll_dir

The allowed scroll direction(s), see *lv_dir_t*

uint16_t layer_type

Cache the layer type here. Element of *lv_intermediate_layer_type_t*

uint16_t name_static

1: name was not dynamically allocated

struct _lv_obj_t**Membri Pubblici**

const lv_obj_class_t *class_p

lv_obj_t *parent

lv_obj_spec_attr_t *spec_attr

lv_obj_style_t *styles

void *user_data

```

void *id

lv_area_t coords

lv_obj_flag_t flags

uint16_t state

uint16_t layout_inv

uint16_t readjust_scroll_after_layout

uint16_t scr_layout_inv

uint16_t scr_layout_complete_pending

uint16_t skip_trans

uint16_t style_cnt

uint16_t h_layout

uint16_t w_layout

uint16_t is_deleting

```

14.12.13 lv_obj_scroll.h

Enum

enum **lv_scrollbar_mode_t**

Scrollbar modes: shows when should the scrollbars be visible

Values:

enumerator **LV_SCROLLBAR_MODE_OFF**

Never show scrollbars

enumerator **LV_SCROLLBAR_MODE_ON**

Always show scrollbars

enumerator **LV_SCROLLBAR_MODE_ACTIVE**

Show scroll bars when Widget is being scrolled

enumerator **LV_SCROLLBAR_MODE_AUTO**

Show scroll bars when the content is large enough to be scrolled

enum lv_scroll_snap_t

Scroll span align options. Tells where to align the snappable children when scroll stops.

Values:

enumerator LV_SCROLL_SNAP_NONE

Do not align, leave where it is

enumerator LV_SCROLL_SNAP_START

Align to the left/top

enumerator LV_SCROLL_SNAP_END

Align to the right/bottom

enumerator LV_SCROLL_SNAP_CENTER

Align to the center

Funzioni**void lv_obj_set_scrollbar_mode(lv_obj_t *obj, lv_scrollbar_mode_t mode)**

Set how the scrollbars should behave.

Parametri

- **obj** -- pointer to Widget
- **mode** -- LV_SCROLL_MODE_ON/OFF/AUTO/ACTIVE

void lv_obj_set_scroll_dir(lv_obj_t *obj, lv_dir_t dir)

Set direction Widget can be scrolled

Parametri

- **obj** -- pointer to Widget
- **dir** -- one or more bit-wise OR-ed values of **lv_dir_t** enumeration

void lv_obj_set_scroll_snap_x(lv_obj_t *obj, lv_scroll_snap_t align)

Set where to snap the children when scrolling ends horizontally

Parametri

- **obj** -- pointer to Widget
- **align** -- value from **lv_scroll_snap_t** enumeration

void lv_obj_set_scroll_snap_y(lv_obj_t *obj, lv_scroll_snap_t align)

Set where to snap the children when scrolling ends vertically

Parametri

- **obj** -- pointer to Widget
- **align** -- value from **lv_scroll_snap_t** enumeration

lv_scrollbar_mode_t lv_obj_get_scrollbar_mode(const lv_obj_t *obj)

Get the current scroll mode (when to hide the scrollbars)

Parametri

obj -- pointer to Widget

Ritorna

the current scroll mode from **lv_scrollbar_mode_t**

lv_dir_t lv_obj_get_scroll_dir(const lv_obj_t *obj)

Get directions Widget can be scrolled (set with **lv_obj_set_scroll_dir()**)

Parametri

obj -- pointer to Widget

Ritorna

current scroll direction bit(s)

`lv_scroll_snap_t lv_obj_get_scroll_snap_x(const lv_obj_t *obj)`

Get where to snap child Widgets when horizontal scrolling ends.

Parametri

`obj` -- pointer to Widget

Ritorna

current snap value from `lv_scroll_snap_t`

`lv_scroll_snap_t lv_obj_get_scroll_snap_y(const lv_obj_t *obj)`

Get where to snap child Widgets when vertical scrolling ends.

Parametri

`obj` -- pointer to Widget

Ritorna

current snap value from `lv_scroll_snap_t`

`int32_t lv_obj_get_scroll_x(const lv_obj_t *obj)`

Get current X scroll position. Identical to `lv_obj_get_scroll_left()`.

Parametri

`obj` -- pointer to scrollable container Widget

Ritorna

current scroll position from left edge

- If Widget is not scrolled return 0.
- If scrolled return > 0.
- If scrolled inside (elastic scroll) return < 0.

`int32_t lv_obj_get_scroll_y(const lv_obj_t *obj)`

Get current Y scroll position. Identical to `lv_obj_get_scroll_top()`.

Parametri

`obj` -- pointer to scrollable container Widget

Ritorna

current scroll position from top edge

- If Widget is not scrolled return 0.
- If scrolled return > 0.
- If scrolled inside (elastic scroll) return < 0.

`int32_t lv_obj_get_scroll_top(const lv_obj_t *obj)`

Number of pixels a scrollable container Widget can be scrolled down before its top edge appears. When `LV_OBJ_FLAG_SCROLL_ELASTIC` flag is set in Widget, this value can go negative while Widget is being dragged below its normal top-edge boundary.

Parametri

`obj` -- pointer to scrollable container Widget

Ritorna

pixels Widget can be scrolled down before its top edge appears

`int32_t lv_obj_get_scroll_bottom(const lv_obj_t *obj)`

Number of pixels a scrollable container Widget can be scrolled up before its bottom edge appears. When `LV_OBJ_FLAG_SCROLL_ELASTIC` flag is set in Widget, this value can go negative while Widget is being dragged above its normal bottom-edge boundary.

Parametri

`obj` -- pointer to scrollable container Widget

Ritorna

pixels Widget can be scrolled up before its bottom edge appears

`int32_t lv_obj_get_scroll_left(const lv_obj_t *obj)`

Number of pixels a scrollable container Widget can be scrolled right before its left edge appears. When `LV_OBJ_FLAG_SCROLL_ELASTIC` flag is set in Widget, this value can go negative while Widget is being dragged farther right than its normal left-edge boundary.

Parametri

`obj` -- pointer to scrollable container Widget

Ritorna

pixels Widget can be scrolled right before its left edge appears

```
int32_t lv_obj_get_scroll_right(const lv_obj_t *obj)
```

Number of pixels a scrollable container Widget can be scrolled left before its right edge appears. When LV_OBJ_FLAG_SCROLL_ELASTIC flag is set in Widget, this value can go negative while Widget is being dragged farther left than its normal right-edge boundary.

Parametri

- **obj** -- pointer to scrollable container Widget

Ritorna

pixels Widget can be scrolled left before its right edge appears

```
void lv_obj_get_scroll_end(lv_obj_t *obj, lv_point_t *end)
```

Get the X and Y coordinates where the scrolling will end for Widget if a scrolling animation is in progress. If no scrolling animation, give the current x or y scroll position.

Parametri

- **obj** -- pointer to scrollable Widget
- **end** -- pointer to *lv_point_t* object in which to store result

```
void lv_obj_scroll_by(lv_obj_t *obj, int32_t dx, int32_t dy, lv_anim_enable_t anim_en)
```

Scroll by given amount of pixels.

Nota

> 0 value means scroll right/bottom (show the more content on the right/bottom)

Nota

e.g. dy = -20 means scroll down 20 px

Parametri

- **obj** -- pointer to scrollable Widget to scroll
- **dx** -- pixels to scroll horizontally
- **dy** -- pixels to scroll vertically
- **anim_en** -- LV_ANIM_ON: scroll with animation; LV_ANIM_OFF: scroll immediately

```
void lv_obj_scroll_by_bounded(lv_obj_t *obj, int32_t dx, int32_t dy, lv_anim_enable_t anim_en)
```

Scroll by given amount of pixels. dx and dy will be limited internally to allow scrolling only on the content area.

Nota

e.g. dy = -20 means scroll down 20 px

Parametri

- **obj** -- pointer to scrollable Widget to scroll
- **dx** -- pixels to scroll horizontally
- **dy** -- pixels to scroll vertically
- **anim_en** -- LV_ANIM_ON: scroll with animation; LV_ANIM_OFF: scroll immediately

```
void lv_obj_scroll_to(lv_obj_t *obj, int32_t x, int32_t y, lv_anim_enable_t anim_en)
```

Scroll to given coordinate on Widget. x and y will be limited internally to allow scrolling only on the content area.

Parametri

- **obj** -- pointer to scrollable Widget to scroll
- **x** -- pixels to scroll horizontally
- **y** -- pixels to scroll vertically

- **anim_en** -- LV_ANIM_ON: scroll with animation; LV_ANIM_OFF: scroll immediately

void lv_obj_scroll_to_x(lv_obj_t *obj, int32_t x, *lv_anim_enable_t* anim_en)

Scroll to X coordinate on Widget. x will be limited internally to allow scrolling only on the content area.

Parametri

- **obj** -- pointer to scrollable Widget to scroll
- **x** -- pixels to scroll horizontally
- **anim_en** -- LV_ANIM_ON: scroll with animation; LV_ANIM_OFF: scroll immediately

void lv_obj_scroll_to_y(lv_obj_t *obj, int32_t y, *lv_anim_enable_t* anim_en)

Scroll to Y coordinate on Widget. y will be limited internally to allow scrolling only on the content area.

Parametri

- **obj** -- pointer to scrollable Widget to scroll
- **y** -- pixels to scroll vertically
- **anim_en** -- LV_ANIM_ON: scroll with animation; LV_ANIM_OFF: scroll immediately

void lv_obj_scroll_to_view(lv_obj_t *obj, *lv_anim_enable_t* anim_en)

Scroll obj's parent Widget until obj becomes visible.

Parametri

- **obj** -- pointer to Widget to scroll into view
- **anim_en** -- LV_ANIM_ON: scroll with animation; LV_ANIM_OFF: scroll immediately

void lv_obj_scroll_to_view_recursive(lv_obj_t *obj, *lv_anim_enable_t* anim_en)

Scroll obj's parent Widgets recursively until obj becomes visible. Widget will be scrolled into view even it has nested scrollable parents.

Parametri

- **obj** -- pointer to Widget to scroll into view
- **anim_en** -- LV_ANIM_ON: scroll with animation; LV_ANIM_OFF: scroll immediately

bool lv_obj_is_scrolling(const lv_obj_t *obj)

Tell whether Widget is being scrolled or not at this moment

Parametri

obj -- pointer to Widget

Ritorna

true: obj is being scrolled

void lv_obj_stop_scroll_anim(const lv_obj_t *obj)

Stop scrolling the current object

Parametri

obj -- The object being scrolled

void lv_obj_update_snap(lv_obj_t *obj, *lv_anim_enable_t* anim_en)

Check children of obj and scroll obj to fulfill scroll_snap settings.

Parametri

- **obj** -- Widget whose children need to be checked and snapped
- **anim_en** -- LV_ANIM_ON/OFF

void lv_obj_get_scrollbar_area(lv_obj_t *obj, *lv_area_t* *hor, *lv_area_t* *ver)

Get the area of the scrollbars

Parametri

- **obj** -- pointer to Widget
- **hor** -- pointer to store the area of the horizontal scrollbar
- **ver** -- pointer to store the area of the vertical scrollbar

void lv_obj_scrollbar_invalidate(lv_obj_t *obj)

Invalidate the area of the scrollbars

Parametri

- obj** -- pointer to Widget

```
void lv_obj_readjust_scroll(lv_obj_t *obj, lv_anim_enable_t anim_en)
```

Checks if the content is scrolled "in" and adjusts it to a normal position.

Parametri

- **obj** -- pointer to Widget
- **anim_en** -- LV_ANIM_ON/OFF

14.12.14 lv_obj_scroll_private.h

Funzioni

```
lv_result_t lv_obj_scroll_by_raw(lv_obj_t *obj, int32_t x, int32_t y)
```

Low level function to scroll by given x and y coordinates. LV_EVENT_SCROLL is sent.

Parametri

- **obj** -- pointer to an object to scroll
- **x** -- pixels to scroll horizontally
- **y** -- pixels to scroll vertically

Ritorna

LV_RESULT_INVALID: to object was deleted in LV_EVENT_SCROLL; LV_RESULT_OK: if the object is still valid

14.12.15 lv_obj_style.h

Typedef

```
typedef uint32_t lv_style_selector_t
```

A joint type for *lv_part_t* and *lv_state_t*. Example values

- 0: means LV_PART_MAIN | LV_STATE_DEFAULT
- LV_STATE_PRSSED
- LV_PART_KNOB
- LV_PART_KNOB | LV_STATE_PRESSED | LV_STATE_CHECKED

Enum

```
enum lv_state_t
```

Possible states of a widget. OR-ed values are possible

Values:

enumerator **LV_STATE_DEFAULT**

enumerator **LV_STATE_CHECKED**

enumerator **LV_STATE_FOCUSED**

enumerator **LV_STATE_FOCUS_KEY**

enumerator **LV_STATE_EDITED**

enumerator **LV_STATE_HOVERED**

enumerator **LV_STATE_PRESSED**

enumerator **LV_STATE_SCROLLLED**

enumerator **LV_STATE_DISABLED**

enumerator **LV_STATE_USER_1**

enumerator **LV_STATE_USER_2**

enumerator **LV_STATE_USER_3**

enumerator **LV_STATE_USER_4**

enumerator **LV_STATE_ANY**

Special value can be used in some functions to target all states

enum **lv_part_t**

The possible parts of widgets. The parts can be considered as the internal building block of the widgets. E.g. slider = background + indicator + knob Not all parts are used by every widget

Values:

enumerator **LV_PART_MAIN**

A background like rectangle

enumerator **LV_PART_SCROLLBAR**

The scrollbar(s)

enumerator **LV_PART_INDICATOR**

Indicator, e.g. for slider, bar, switch, or the tick box of the checkbox

enumerator **LV_PART_KNOB**

Like handle to grab to adjust the value

enumerator **LV_PART_SELECTED**

Indicate the currently selected option or section

enumerator **LV_PART_ITEMS**

Used if the widget has multiple similar elements (e.g. table cells)

enumerator **LV_PART_CURSOR**

Mark a specific place e.g. for text area's cursor or on a chart

enumerator **LV_PART_CUSTOM_FIRST**

Extension point for custom widgets

enumerator **LV_PART_ANY**

Special value can be used in some functions to target all parts

enum lv_style_state_cmp_t

Values:

enumerator LV_STYLE_STATE_CMP_SAME

The style properties in the 2 states are identical

enumerator LV_STYLE_STATE_CMP_DIFF_REDRAW

The differences can be shown with a simple redraw

enumerator LV_STYLE_STATE_CMP_DIFF_DRAW_PAD

The differences can be shown with a simple redraw

enumerator LV_STYLE_STATE_CMP_DIFF_LAYOUT

The differences can be shown with a simple redraw

Funzioni

void lv_obj_add_style(lv_obj_t *obj, const lv_style_t *style, lv_style_selector_t selector)

Add a style to an object.

Examples:

```
lv_obj_add_style(btn, &style_btn, 0); //Default button style
lv_obj_add_style(btn, &btn_red, LV_STATE_PRESSED); //Overwrite only some colors to red when pressed
```

Parametri

- **obj** -- pointer to an object
- **style** -- pointer to a style to add
- **selector** -- OR-ed value of parts and state to which the style should be added

bool lv_obj_replace_style(lv_obj_t *obj, const lv_style_t *old_style, const lv_style_t *new_style, lv_style_selector_t selector)

Replaces a style of an object, preserving the order of the style stack (local styles and transitions are ignored).

Examples:

```
lv_obj_replace_style(obj, &yellow_style, &blue_style, LV_PART_ANY | LV_STATE_ANY); //Replace a specific style
lv_obj_replace_style(obj, &yellow_style, &blue_style, LV_PART_MAIN | LV_STATE_PRESSED); //Replace a specific style assigned to the main part when it is pressed
```

Parametri

- **obj** -- pointer to an object
- **old_style** -- pointer to a style to replace.
- **new_style** -- pointer to a style to replace the old style with.
- **selector** -- OR-ed values of states and a part to replace only styles with matching selectors. LV_STATE_ANY and LV_PART_ANY can be used

void lv_obj_remove_style(lv_obj_t *obj, const lv_style_t *style, lv_style_selector_t selector)

Remove a style from an object.

Examples:

```
lv_obj_remove_style(obj, &style, LV_PART_ANY | LV_STATE_ANY); //Remove a specific style
lv_obj_remove_style(obj, NULL, LV_PART_MAIN | LV_STATE_ANY); //Remove all styles from the main part
lv_obj_remove_style(obj, NULL, LV_PART_ANY | LV_STATE_ANY); //Remove all styles
```

Parametri

- **obj** -- pointer to an object
- **style** -- pointer to a style to remove. Can be NULL to check only the selector

- **selector** -- OR-ed values of states and a part to remove only styles with matching selectors. LV_STATE_ANY and LV_PART_ANY can be used

`void lv_obj_remove_style_all(lv_obj_t *obj)`

Remove all styles from an object

Parametri

- **obj** -- pointer to an object

`void lv_obj_report_style_change(lv_style_t *style)`

Notify all object if a style is modified

Parametri

- **style** -- pointer to a style. Only the objects with this style will be notified (NULL to notify all objects)

`void lv_obj_refresh_style(lv_obj_t *obj, lv_part_t part, lv_style_prop_t prop)`

Notify an object and its children about its style is modified.

Parametri

- **obj** -- pointer to an object
- **part** -- the part whose style was changed. E.g. LV_PART_ANY, LV_PART_MAIN
- **prop** -- LV_STYLE_PROP_ANY or an LV_STYLE_... property. It is used to optimize what needs to be refreshed. LV_STYLE_PROP_INV to perform only a style cache update

`void lv_obj_style_set_disabled(lv_obj_t *obj, const lv_style_t *style, lv_style_selector_t selector, bool dis)`

Temporary disable a style for a selector. It will look like is the style wasn't added

Parametri

- **obj** -- pointer to an object
- **style** -- pointer to a style
- **selector** -- the selector of a style (e.g. LV_STATE_PRESSED | LV_PART_KNOB)
- **dis** -- true: disable the style, false: enable the style

`bool lv_obj_style_get_disabled(lv_obj_t *obj, const lv_style_t *style, lv_style_selector_t selector)`

Get if a given style is disabled on an object.

Parametri

- **obj** -- pointer to an object
- **style** -- pointer to a style
- **selector** -- the selector of a style (e.g. LV_STATE_PRESSED | LV_PART_KNOB)

Ritorna

true: disable the style, false: enable the style

`void lv_obj_enable_style_refresh(bool en)`

Enable or disable automatic style refreshing when a new style is added/removed to/from an object or any other style change happens.

Parametri

- **en** -- true: enable refreshing; false: disable refreshing

`lv_style_value_t lv_obj_get_style_prop(const lv_obj_t *obj, lv_part_t part, lv_style_prop_t prop)`

Get the value of a style property. The current state of the object will be considered. Inherited properties will be inherited. If a property is not set a default value will be returned.

Parametri

- **obj** -- pointer to an object
- **part** -- a part from which the property should be get
- **prop** -- the property to get

Ritorna

the value of the property. Should be read from the correct field of the `lv_style_value_t` according to the type of the property.

`bool lv_obj_has_style_prop(const lv_obj_t *obj, lv_style_selector_t selector, lv_style_prop_t prop)`

Check if an object has a specified style property for a given style selector.

Parametri

- **obj** -- pointer to an object
- **selector** -- the style selector to be checked, defining the scope of the style to be examined.
- **prop** -- the property to be checked.

Ritorna

true if the object has the specified selector and property, false otherwise.

```
void lv_obj_set_local_style_prop(lv_obj_t *obj, lv_style_prop_t prop, lv_style_value_t value,
                                lv_style_selector_t selector)
```

Set local style property on an object's part and state.

Parametri

- **obj** -- pointer to an object
- **prop** -- the property
- **value** -- value of the property. The correct element should be set according to the type of the property
- **selector** -- OR-ed value of parts and state for which the style should be set

```
lv_style_res_t lv_obj_get_local_style_prop(lv_obj_t *obj, lv_style_prop_t prop, lv_style_value_t *value,
                                           lv_style_selector_t selector)
```

```
bool lv_obj_remove_local_style_prop(lv_obj_t *obj, lv_style_prop_t prop, lv_style_selector_t selector)
```

Remove a local style property from a part of an object with a given state.

Parametri

- **obj** -- pointer to an object
- **prop** -- a style property to remove.
- **selector** -- OR-ed value of parts and state for which the style should be removed

Ritorna

true the property was found and removed; false: the property was not found

```
lv_style_value_t lv_obj_style_apply_color_filter(const lv_obj_t *obj, lv_part_t part, lv_style_value_t v)
```

Used internally for color filtering

```
void lv_obj_fade_in(lv_obj_t *obj, uint32_t time, uint32_t delay)
```

Fade in an an object and all its children.

Parametri

- **obj** -- the object to fade in
- **time** -- time of fade
- **delay** -- delay to start the animation

```
void lv_obj_fade_out(lv_obj_t *obj, uint32_t time, uint32_t delay)
```

Fade out an an object and all its children.

Parametri

- **obj** -- the object to fade out
- **time** -- time of fade
- **delay** -- delay to start the animation

```
static inline lv_state_t lv_obj_style_get_selector_state(lv_style_selector_t selector)
```

```
static inline lv_part_t lv_obj_style_get_selector_part(lv_style_selector_t selector)
```

```
static inline void lv_obj_set_style_pad_all(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
```

```
static inline void lv_obj_set_style_pad_hor(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
```

```
static inline void lv_obj_set_style_pad_ver(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
```

```
static inline void lv_obj_set_style_margin_all(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
```

```
static inline void lv_obj_set_style_margin_hor(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
```

```
static inline void lv_obj_set_style_margin_ver(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
static inline void lv_obj_set_style_pad_gap(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
static inline void lv_obj_set_style_size(lv_obj_t *obj, int32_t width, int32_t height, lv_style_selector_t
selector)

static inline void lv_obj_set_style_transform_scale(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
static inline int32_t lv_obj_get_style_space_left(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_space_right(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_space_top(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_space_bottom(const lv_obj_t *obj, lv_part_t part)

lv_text_align_t lv_obj_calculate_style_text_align(const lv_obj_t *obj, lv_part_t part, const char *txt)
static inline int32_t lv_obj_get_style_transform_scale_x_safe(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_scale_y_safe(const lv_obj_t *obj, lv_part_t part)

lv_opa_t lv_obj_get_style_opa_recursive(const lv_obj_t *obj, lv_part_t part)
```

Get the opa style property from all parents and multiply and >> 8 them.

Parametri

- **obj** -- the object whose opacity should be get
- **part** -- the part whose opacity should be get. Non-MAIN parts will consider the opa of the MAIN part too

Ritorna

the final opacity considering the parents' opacity too

```
lv_color32_t lv_obj_style_apply_recolor(const lv_obj_t *obj, lv_part_t part, lv_color32_t color)
```

Apply recolor effect to the input color based on the object's style properties.

Parametri

- **obj** -- the target object containing recolor style properties
- **part** -- the part to retrieve recolor styles.
- **color** -- the original color to be modified

Ritorna

the blended color after applying recolor and opacity

```
lv_color32_t lv_obj_get_style_recolor_recursive(const lv_obj_t *obj, lv_part_t part)
```

Get the recolor style property from all parents and blend them recursively.

Parametri

- **obj** -- the object whose recolor value should be retrieved
- **part** -- the target part to check. Non-MAIN parts will also consider the recolor value from the MAIN part during calculation

Ritorna

the final blended recolor value combining all parent's recolor values

14.12.16 lv_obj_style_gen.h

Funzioni

```
static inline int32_t lv_obj_get_style_width(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_min_width(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_max_width(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_height(const lv_obj_t *obj, lv_part_t part)
```

```
static inline int32_t lv_obj_get_style_min_height(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_max_height(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_length(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_x(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_y(const lv_obj_t *obj, lv_part_t part)
static inline lv_align_t lv_obj_get_style_align(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_width(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_height(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_translate_x(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_translate_y(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_translate_radial(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_scale_x(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_scale_y(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_rotation(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_pivot_x(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_pivot_y(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_skew_x(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_skew_y(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_pad_top(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_pad_bottom(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_pad_left(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_pad_right(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_pad_row(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_pad_column(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_pad_radial(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_margin_top(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_margin_bottom(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_margin_left(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_margin_right(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_bg_color(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_bg_color_filtered(const lv_obj_t *obj, lv_part_t part)
static inline lv_opa_t lv_obj_get_style_bg_opa(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_bg_grad_color(const lv_obj_t *obj, lv_part_t part)
```

```

static inline lv_color_t lv_obj_get_style_bg_grad_color_filtered(const lv_obj_t *obj, lv_part_t part)
static inline lv_grad_dir_t lv_obj_get_style_bg_grad_dir(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_bg_main_stop(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_bg_grad_stop(const lv_obj_t *obj, lv_part_t part)
static inline lv_opa_t lv_obj_get_style_bg_main_opa(const lv_obj_t *obj, lv_part_t part)
static inline lv_opa_t lv_obj_get_style_bg_grad_opa(const lv_obj_t *obj, lv_part_t part)
static inline const lv_grad_dsc_t *lv_obj_get_style_bg_grad(const lv_obj_t *obj, lv_part_t part)
static inline const void *lv_obj_get_style_bg_image_src(const lv_obj_t *obj, lv_part_t part)
static inline lv_opa_t lv_obj_get_style_bg_image_opa(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_bg_image_recolor(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_bg_image_recolor_filtered(const lv_obj_t *obj, lv_part_t part)
static inline lv_opa_t lv_obj_get_style_bg_image_recolor_opa(const lv_obj_t *obj, lv_part_t part)
static inline bool lv_obj_get_style_bg_image_tiled(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_border_color(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_border_color_filtered(const lv_obj_t *obj, lv_part_t part)
static inline lv_opa_t lv_obj_get_style_border_opa(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_border_width(const lv_obj_t *obj, lv_part_t part)
static inline lv_border_side_t lv_obj_get_style_border_side(const lv_obj_t *obj, lv_part_t part)
static inline bool lv_obj_get_style_border_post(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_outline_width(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_outline_color(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_outline_color_filtered(const lv_obj_t *obj, lv_part_t part)
static inline lv_opa_t lv_obj_get_style_outline_opa(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_outline_pad(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_shadow_width(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_shadow_offset_x(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_shadow_offset_y(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_shadow_spread(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_shadow_color(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_shadow_color_filtered(const lv_obj_t *obj, lv_part_t part)
static inline lv_opa_t lv_obj_get_style_shadow_opa(const lv_obj_t *obj, lv_part_t part)
static inline lv_opa_t lv_obj_get_style_image_opa(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_image_recolor(const lv_obj_t *obj, lv_part_t part)

```

```

static inline lv_color_t lv_obj_get_style_image_recolor_filtered(const lv_obj_t *obj, lv_part_t part)

static inline lv_opa_t lv_obj_get_style_image_recolor_opa(const lv_obj_t *obj, lv_part_t part)

static inline const lv_image_colorkey_t *lv_obj_get_style_image_colorkey(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_line_width(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_line_dash_width(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_line_dash_gap(const lv_obj_t *obj, lv_part_t part)

static inline bool lv_obj_get_style_line_rounded(const lv_obj_t *obj, lv_part_t part)

static inline lv_color_t lv_obj_get_style_line_color(const lv_obj_t *obj, lv_part_t part)

static inline lv_color_t lv_obj_get_style_line_color_filtered(const lv_obj_t *obj, lv_part_t part)

static inline lv_opa_t lv_obj_get_style_line_opa(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_arc_width(const lv_obj_t *obj, lv_part_t part)

static inline bool lv_obj_get_style_arc_rounded(const lv_obj_t *obj, lv_part_t part)

static inline lv_color_t lv_obj_get_style_arc_color(const lv_obj_t *obj, lv_part_t part)

static inline lv_color_t lv_obj_get_style_arc_color_filtered(const lv_obj_t *obj, lv_part_t part)

static inline lv_opa_t lv_obj_get_style_arc_opa(const lv_obj_t *obj, lv_part_t part)

static inline const void *lv_obj_get_style_arc_image_src(const lv_obj_t *obj, lv_part_t part)

static inline lv_color_t lv_obj_get_style_text_color(const lv_obj_t *obj, lv_part_t part)

static inline lv_color_t lv_obj_get_style_text_color_filtered(const lv_obj_t *obj, lv_part_t part)

static inline lv_opa_t lv_obj_get_style_text_opa(const lv_obj_t *obj, lv_part_t part)

static inline const lv_font_t *lv_obj_get_style_text_font(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_text_letter_space(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_text_line_space(const lv_obj_t *obj, lv_part_t part)

static inline lv_text_decor_t lv_obj_get_style_text_decor(const lv_obj_t *obj, lv_part_t part)

static inline lv_text_align_t lv_obj_get_style_text_align(const lv_obj_t *obj, lv_part_t part)

static inline lv_color_t lv_obj_get_style_text_outline_stroke_color(const lv_obj_t *obj, lv_part_t part)

static inline lv_color_t lv_obj_get_style_text_outline_stroke_color_filtered(const lv_obj_t *obj,
    lv_part_t part)

static inline int32_t lv_obj_get_style_text_outline_stroke_width(const lv_obj_t *obj, lv_part_t part)

static inline lv_opa_t lv_obj_get_style_text_outline_stroke_opa(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_radius(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_radial_offset(const lv_obj_t *obj, lv_part_t part)

static inline bool lv_obj_get_style_clip_corner(const lv_obj_t *obj, lv_part_t part)

static inline lv_opa_t lv_obj_get_style_opa(const lv_obj_t *obj, lv_part_t part)

```

```

static inline lv_opa_t lv_obj_get_style_opa_layered(const lv_obj_t *obj, lv_part_t part)

static inline const lv_color_filter_dsc_t *lv_obj_get_style_color_filter_dsc(const lv_obj_t *obj, lv_part_t part)

static inline lv_opa_t lv_obj_get_style_color_filter_opa(const lv_obj_t *obj, lv_part_t part)

static inline lv_color_t lv_obj_get_style_recolor(const lv_obj_t *obj, lv_part_t part)

static inline lv_opa_t lv_obj_get_style_recolor_opa(const lv_obj_t *obj, lv_part_t part)

static inline const lv_anim_t *lv_obj_get_style_anim(const lv_obj_t *obj, lv_part_t part)

static inline uint32_t lv_obj_get_style_anim_duration(const lv_obj_t *obj, lv_part_t part)

static inline const lv_style_transition_dsc_t *lv_obj_get_style_transition(const lv_obj_t *obj, lv_part_t part)

static inline lv_blend_mode_t lv_obj_get_style_blend_mode(const lv_obj_t *obj, lv_part_t part)

static inline uint16_t lv_obj_get_style_layout(const lv_obj_t *obj, lv_part_t part)

static inline lv_base_dir_t lv_obj_get_style_base_dir(const lv_obj_t *obj, lv_part_t part)

static inline const void *lv_obj_get_style_bitmap_src(const lv_obj_t *obj, lv_part_t part)

static inline uint32_t lv_obj_get_style_rotary_sensitivity(const lv_obj_t *obj, lv_part_t part)

static inline lv_flex_flow_t lv_obj_get_style_flex_flow(const lv_obj_t *obj, lv_part_t part)

static inline lv_flex_align_t lv_obj_get_style_flex_main_place(const lv_obj_t *obj, lv_part_t part)

static inline lv_flex_align_t lv_obj_get_style_flex_cross_place(const lv_obj_t *obj, lv_part_t part)

static inline lv_flex_align_t lv_obj_get_style_flex_track_place(const lv_obj_t *obj, lv_part_t part)

static inline uint8_t lv_obj_get_style_flex_grow(const lv_obj_t *obj, lv_part_t part)

static inline const int32_t *lv_obj_get_style_grid_column_dsc_array(const lv_obj_t *obj, lv_part_t part)

static inline lv_grid_align_t lv_obj_get_style_grid_column_align(const lv_obj_t *obj, lv_part_t part)

static inline const int32_t *lv_obj_get_style_grid_row_dsc_array(const lv_obj_t *obj, lv_part_t part)

static inline lv_grid_align_t lv_obj_get_style_grid_row_align(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_grid_cell_column_pos(const lv_obj_t *obj, lv_part_t part)

static inline lv_grid_align_t lv_obj_get_style_grid_cell_x_align(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_grid_cell_column_span(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_grid_cell_row_pos(const lv_obj_t *obj, lv_part_t part)

static inline lv_grid_align_t lv_obj_get_style_grid_cell_y_align(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_grid_cell_row_span(const lv_obj_t *obj, lv_part_t part)

void lv_obj_set_style_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_min_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_max_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_height(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

```

```
void lv_obj_set_style_min_height(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_max_height(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_length(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_x(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_y(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_align(lv_obj_t *obj, lv_align_t value, lv_style_selector_t selector)
void lv_obj_set_style_transform_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_transform_height(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_translate_x(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_translate_y(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_translate_radial(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_transform_scale_x(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_transform_scale_y(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_transform_rotation(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_transform_pivot_x(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_transform_pivot_y(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_transform_skew_x(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_transform_skew_y(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_pad_top(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_pad_bottom(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_pad_left(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_pad_right(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_pad_row(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_pad_column(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_pad_radial(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_margin_top(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_margin_bottom(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_margin_left(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_margin_right(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_color(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_grad_color(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_grad_dir(lv_obj_t *obj, lv_grad_dir_t value, lv_style_selector_t selector)
```

```

void lv_obj_set_style_bg_main_stop(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_grad_stop(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_main_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_grad_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_grad(lv_obj_t *obj, const lv_grad_dsc_t *value, lv_style_selector_t selector)
void lv_obj_set_style_bg_image_src(lv_obj_t *obj, const void *value, lv_style_selector_t selector)
void lv_obj_set_style_bg_image_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_image_recolor(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_image_recolor_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_image_tiled(lv_obj_t *obj, bool value, lv_style_selector_t selector)
void lv_obj_set_style_border_color(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_border_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_border_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_border_side(lv_obj_t *obj, lv_border_side_t value, lv_style_selector_t selector)
void lv_obj_set_style_border_post(lv_obj_t *obj, bool value, lv_style_selector_t selector)
void lv_obj_set_style_outline_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_outline_color(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_outline_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_outline_pad(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_shadow_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_shadow_offset_x(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_shadow_offset_y(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_shadow_spread(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_shadow_color(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_shadow_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_image_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_image_recolor(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_image_recolor_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_image_colorkey(lv_obj_t *obj, const lv_image_colorkey_t *value, lv_style_selector_t selector)

void lv_obj_set_style_line_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_line_dash_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_line_dash_gap(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

```

```

void lv_obj_set_style_line_rounded(lv_obj_t *obj, bool value, lv_style_selector_t selector)
void lv_obj_set_style_line_color(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_line_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_arc_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_arc_rounded(lv_obj_t *obj, bool value, lv_style_selector_t selector)
void lv_obj_set_style_arc_color(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_arc_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_arc_image_src(lv_obj_t *obj, const void *value, lv_style_selector_t selector)
void lv_obj_set_style_text_color(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_text_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_text_font(lv_obj_t *obj, const lv_font_t *value, lv_style_selector_t selector)
void lv_obj_set_style_text_letter_space(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_text_line_space(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_text_decor(lv_obj_t *obj, lv_text_decor_t value, lv_style_selector_t selector)
void lv_obj_set_style_text_align(lv_obj_t *obj, lv_text_align_t value, lv_style_selector_t selector)
void lv_obj_set_style_text_outline_stroke_color(lv_obj_t *obj, lv_color_t value, lv_style_selector_t
                                              selector)

void lv_obj_set_style_text_outline_stroke_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_text_outline_stroke_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_radius(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_radial_offset(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_clip_corner(lv_obj_t *obj, bool value, lv_style_selector_t selector)
void lv_obj_set_style_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_opa_layered(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_color_filter_dsc(lv_obj_t *obj, const lv_color_filter_dsc_t *value, lv_style_selector_t
                                       selector)

void lv_obj_set_style_color_filter_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_recolor(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_recolor_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_anim(lv_obj_t *obj, const lv_anim_t *value, lv_style_selector_t selector)
void lv_obj_set_style_anim_duration(lv_obj_t *obj, uint32_t value, lv_style_selector_t selector)
void lv_obj_set_style_transition(lv_obj_t *obj, const lv_style_transition_desc_t *value, lv_style_selector_t
                                 selector)

void lv_obj_set_style_blend_mode(lv_obj_t *obj, lv_blend_mode_t value, lv_style_selector_t selector)

```

```

void lv_obj_set_style_layout(lv_obj_t *obj, uint16_t value, lv_style_selector_t selector)
void lv_obj_set_style_base_dir(lv_obj_t *obj, lv_base_dir_t value, lv_style_selector_t selector)
void lv_obj_set_style_bitmap_mask_src(lv_obj_t *obj, const void *value, lv_style_selector_t selector)
void lv_obj_set_style_rotary_sensitivity(lv_obj_t *obj, uint32_t value, lv_style_selector_t selector)
void lv_obj_set_style_flex_flow(lv_obj_t *obj, lv_flex_flow_t value, lv_style_selector_t selector)
void lv_obj_set_style_flex_main_place(lv_obj_t *obj, lv_flex_align_t value, lv_style_selector_t selector)
void lv_obj_set_style_flex_cross_place(lv_obj_t *obj, lv_flex_align_t value, lv_style_selector_t selector)
void lv_obj_set_style_flex_track_place(lv_obj_t *obj, lv_flex_align_t value, lv_style_selector_t selector)
void lv_obj_set_style_flex_grow(lv_obj_t *obj, uint8_t value, lv_style_selector_t selector)
void lv_obj_set_style_grid_column_dsc_array(lv_obj_t *obj, const int32_t *value, lv_style_selector_t
                                          selector)
void lv_obj_set_style_grid_column_align(lv_obj_t *obj, lv_grid_align_t value, lv_style_selector_t selector)
void lv_obj_set_style_grid_row_dsc_array(lv_obj_t *obj, const int32_t *value, lv_style_selector_t selector)
void lv_obj_set_style_grid_row_align(lv_obj_t *obj, lv_grid_align_t value, lv_style_selector_t selector)
void lv_obj_set_style_grid_cell_column_pos(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_grid_cell_x_align(lv_obj_t *obj, lv_grid_align_t value, lv_style_selector_t selector)
void lv_obj_set_style_grid_cell_column_span(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_grid_cell_row_pos(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_grid_cell_y_align(lv_obj_t *obj, lv_grid_align_t value, lv_style_selector_t selector)
void lv_obj_set_style_grid_cell_row_span(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

```

14.12.17 lv_obj_style_private.h

Funzioni

`void lv_obj_style_init(void)`

Initialize the object related style manager module. Called by LVGL in `lv_init()`

`void lv_obj_style_deinit(void)`

Deinitialize the object related style manager module. Called by LVGL in `lv_deinit()`

`void lv_obj_style_create_transition(lv_obj_t *obj, lv_part_t part, lv_state_t prev_state, lv_state_t new_state,
 const lv_obj_style_transition_dsc_t *tr)`

Used internally to create a style transition

Parametri

- `obj` --
- `part` --
- `prev_state` --
- `new_state` --
- `tr` --

`lv_style_state_cmp_t lv_obj_style_state_compare(lv_obj_t *obj, lv_state_t state1, lv_state_t state2)`

Used internally to compare the appearance of an object in 2 states

Parametri

- `obj` --

- **state1** --
- **state2** --

Ritorna

```
void lv_obj_update_layer_type(lv_obj_t *obj)
```

Update the layer type of a widget based on its current styles. The result will be stored in `obj->spec_attr->layer_type`

Parametri

`obj` -- the object whose layer should be updated

```
struct _lv_obj_style_t
```

Membri Pubblici

```
const lv_style_t *style
```

```
uint32_t selector
```

```
uint32_t is_local
```

```
uint32_t is_trans
```

```
uint32_t is_disabled
```

```
struct _lv_obj_style_transition_desc_t
```

Membri Pubblici

```
uint16_t time
```

```
uint16_t delay
```

```
lv_style_selector_t selector
```

```
lv_style_prop_t prop
```

```
lv_anim_path_cb_t path_cb
```

```
void *user_data
```

14.12.18 lv_obj_tree.h

Typedef

```
typedef lv_obj_tree_walk_res_t (*lv_obj_tree_walk_cb_t)(lv_obj_t*, void*)
```

Enum

enum **lv_obj_tree_walk_res_t**

Values:

enumerator **LV_OBJ_TREE_WALK_NEXT**

enumerator **LV_OBJ_TREE_WALK_SKIP_CHILDREN**

enumerator **LV_OBJ_TREE_WALK_END**

Funzioni

void **lv_obj_delete**(lv_obj_t *obj)

Delete an object and all of its children. Also remove the objects from their group and remove all animations (if any). Send **LV_EVENT_DELETE** to deleted objects.

Parametri

obj -- pointer to an object

void **lv_obj_clean**(lv_obj_t *obj)

Delete all children of an object. Also remove the objects from their group and remove all animations (if any). Send **LV_EVENT_DELETE** to deleted objects.

Parametri

obj -- pointer to an object

void **lv_obj_delete_delayed**(lv_obj_t *obj, uint32_t delay_ms)

Delete an object after some delay

Parametri

- **obj** -- pointer to an object
- **delay_ms** -- time to wait before delete in milliseconds

void **lv_obj_delete_anim_completed_cb**(lv_anim_t *a)

A function to be easily used in animation ready callback to delete an object when the animation is ready

Parametri

a -- pointer to the animation

void **lv_obj_delete_async**(lv_obj_t *obj)

Helper function for asynchronously deleting objects. Useful for cases where you can't delete an object directly in an **LV_EVENT_DELETE** handler (i.e. parent).

 **Vedi anche**

[lv_async_call](#)

Parametri

obj -- object to delete

void **lv_obj_set_parent**(lv_obj_t *obj, lv_obj_t *parent)

Move the parent of an object. The relative coordinates will be kept.

Parametri

- **obj** -- pointer to an object whose parent needs to be changed
- **parent** -- pointer to the new parent

void **lv_obj_swap**(lv_obj_t *obj1, lv_obj_t *obj2)

Swap the positions of two objects. When used in listboxes, it can be used to sort the listbox items.

Parametri

- **obj1** -- pointer to the first object

- **obj2** -- pointer to the second object

void lv_obj_move_to_index(lv_obj_t *obj, int32_t index)

moves the object to the given index in its parent. When used in listboxes, it can be used to sort the listbox items.

Nota

to move to the background: `lv_obj_move_to_index(obj, 0)`

Nota

to move forward (up): `lv_obj_move_to_index(obj, lv_obj_get_index(obj) - 1)`

Parametri

- **obj** -- pointer to the object to be moved.
- **index** -- new index in parent. -1 to count from the back

lv_obj_t *lv_obj_get_screen(const lv_obj_t *obj)

Get the screen of an object

Parametri

obj -- pointer to an object

Ritorna

pointer to the object's screen

lv_display_t *lv_obj_get_display(const lv_obj_t *obj)

Get the display of the object

Parametri

obj -- pointer to an object

Ritorna

pointer to the object's display

lv_obj_t *lv_obj_get_parent(const lv_obj_t *obj)

Get the parent of an object

Parametri

obj -- pointer to an object

Ritorna

the parent of the object. (NULL if obj was a screen)

lv_obj_t *lv_obj_get_child(const lv_obj_t *obj, int32_t idx)

Get the child of an object by the child's index.

Parametri

- **obj** -- pointer to an object whose child should be get
- **idx** -- the index of the child. 0: the oldest (firstly created) child 1: the second oldest child count-1: the youngest -1: the youngest -2: the second youngest

Ritorna

pointer to the child or NULL if the index was invalid

lv_obj_t *lv_obj_get_child_by_type(const lv_obj_t *obj, int32_t idx, const lv_obj_class_t *class_p)

Get the child of an object by the child's index. Consider the children only with a given type.

Parametri

- **obj** -- pointer to an object whose child should be get
- **idx** -- the index of the child. 0: the oldest (firstly created) child 1: the second oldest child count-1: the youngest -1: the youngest -2: the second youngest
- **class_p** -- the type of the children to check

Ritorna

pointer to the child or NULL if the index was invalid

`lv_obj_t *lv_obj_get_sibling(const lv_obj_t *obj, int32_t idx)`

Return a sibling of an object

Parametri

- **obj** -- pointer to an object whose sibling should be get
- **idx** -- 0: obj itself -1: the first older sibling -2: the next older sibling 1: the first younger sibling 2: the next younger sibling etc

Ritorna

pointer to the requested sibling or NULL if there is no such sibling

`lv_obj_t *lv_obj_get_sibling_by_type(const lv_obj_t *obj, int32_t idx, const lv_obj_class_t *class_p)`

Return a sibling of an object. Consider the siblings only with a given type.

Parametri

- **obj** -- pointer to an object whose sibling should be get
- **idx** -- 0: obj itself -1: the first older sibling -2: the next older sibling 1: the first younger sibling 2: the next younger sibling etc
- **class_p** -- the type of the children to check

Ritorna

pointer to the requested sibling or NULL if there is no such sibling

`uint32_t lv_obj_get_child_count(const lv_obj_t *obj)`

Get the number of children

Parametri

obj -- pointer to an object

Ritorna

the number of children

`uint32_t lv_obj_get_child_count_by_type(const lv_obj_t *obj, const lv_obj_class_t *class_p)`

Get the number of children having a given type.

Parametri

- **obj** -- pointer to an object
- **class_p** -- the type of the children to check

Ritorna

the number of children

`void lv_obj_set_name(lv_obj_t *obj, const char *name)`

Set a name for a widget. The name will be allocated and freed when the widget is deleted or a new name is set.

Nota

If the name ends with a #, older siblings with the same name will be counted, and the # will be replaced by the index of the given widget. For example, creating multiple widgets with the name "mybtn#" will result in resolved names like "mybtn_0", "mybtn_1", "mybtn_2", etc. The name is resolved when `lv_obj_get_name_resolved` is called, so the result reflects the currently existing widgets at that time.

Parametri

- **obj** -- pointer to an object
- **name** -- the name to set. If set to NULL the default "<widget_type>#" name will be used.

`void lv_obj_set_name_static(lv_obj_t *obj, const char *name)`

Set a name for a widget. Only a pointer will be saved.

Nota

If the name ends with a #, older siblings with the same name will be counted, and the # will be replaced by the index of the given widget. For example, creating multiple widgets with the name "mybtn#" will result in resolved names like "mybtn_0", "mybtn_1", "mybtn_2", etc. The name is resolved when `lv_obj_get_name_resolved` is called, so the result reflects the currently existing widgets at that time.

Parametri

- **obj** -- pointer to an object
- **name** -- the name to set. If set to NULL the default "<widget_type>_#" name will be used.

```
const char *lv_obj_get_name(const lv_obj_t *obj)
```

Get the set name as it was set.

Parametri

- **obj** -- pointer to an object

Ritorna

get the set name or NULL if it wasn't set yet

```
void lv_obj_get_name_resolved(const lv_obj_t *obj, char buf[], size_t buf_size)
```

Get the set name or craft a name automatically.

 **Nota**

If the name ends with a #, older siblings with the same name will be counted, and the # will be replaced by the index of the given widget. For example, creating multiple widgets with the name "mybtn_#" will result in resolved names like "mybtn_0", "mybtn_1", "mybtn_2", etc. The name is resolved when `lv_obj_get_name_resolved` is called, so the result reflects the currently existing widgets at that time.

Parametri

- **obj** -- pointer to an object
- **buf** -- buffer to store the name
- **buf_size** -- the size of the buffer in bytes

```
lv_obj_t *lv_obj_find_by_name(const lv_obj_t *parent, const char *name)
```

Find a child with a given name on a parent. This child doesn't have to be the direct child of the parent. First direct children of the parent will be checked, and the direct children of the first child, etc. (Breadth-first search).

If the name of a widget was not set a name like "lv_button_1" will be created for it using `lv_obj_get_name_resolved`.

Parametri

- **parent** -- the widget where the search should start

Ritorna

the found widget or NULL if not found.

```
lv_obj_t *lv_obj_get_child_by_name(const lv_obj_t *parent, const char *name_path)
```

Get an object by name. The name can be a path too, for example "main_container/lv_button_1/label". In this case the first part of the name-path should be the direct child of the parent, the second part, should be the direct child of first one, etc.

If the name of a widget was not set a name like "lv_button_1" will be created for it using `lv_obj_get_name_resolved`.

Parametri

- **parent** -- the widget where the search should start

Ritorna

the found widget or NULL if not found.

```
int32_t lv_obj_get_index(const lv_obj_t *obj)
```

Get the index of a child.

Parametri

- **obj** -- pointer to an object

Ritorna

the child index of the object. E.g. 0: the oldest (firstly created child). (-1 if child could not be found or no parent exists)

```
int32_t lv_obj_get_index_by_type(const lv_obj_t *obj, const lv_obj_class_t *class_p)
```

Get the index of a child. Consider the children only with a given type.

Parametri

- **obj** -- pointer to an object
- **class_p** -- the type of the children to check

Ritorna

the child index of the object. E.g. 0: the oldest (firstly created child with the given class).

(-1 if child could not be found or no parent exists)

```
void lv_obj_tree_walk(lv_obj_t *start_obj, lv_obj_tree_walk_cb_t cb, void *user_data)
```

Iterate through all children of any object.

Parametri

- **start_obj** -- start integrating from this object
- **cb** -- call this callback on the objects
- **user_data** -- pointer to any user related data (will be passed to cb)

```
void lv_obj_dump_tree(lv_obj_t *start_obj)
```

Iterate through all children of any object and print their ID.

Parametri

- **start_obj** -- start integrating from this object

14.12.19 lv_observer.h

Typedef

```
typedef void (*lv_observer_cb_t)(lv_observer_t *observer, lv_subject_t *subject)
```

Callback called to notify Observer that Subject's value has changed

Param observer

pointer to Observer

Param subject

pointer to Subject being observed

Enum

enum lv_subject_type_t

Values for *lv_subject_t*'s type field

Values:

enumerator **LV SUBJECT TYPE INVALID**

indicates Subject not initialized yet

enumerator **LV SUBJECT TYPE NONE**

a null value like None or NILT

enumerator **LV SUBJECT TYPE INT**

an int32_t

enumerator **LV SUBJECT TYPE FLOAT**

a float, requires LV_USE_FLOAT 1

enumerator **LV SUBJECT TYPE POINTER**

a void pointer

enumerator **LV SUBJECT TYPE COLOR**

an *lv_color_t*

enumerator **LV SUBJECT TYPE GROUP**

an array of Subjects

enumerator **LV SUBJECT TYPE STRING**

a char pointer

Funzioni

void lv_subject_init_int(*lv_subject_t* *subject, int32_t value)

Initialize an integer-type Subject.

Parametri

- **subject** -- pointer to Subject
- **value** -- initial value

void lv_subject_set_int(*lv_subject_t* *subject, int32_t value)

Set value of an integer Subject and notify Observers.

Parametri

- **subject** -- pointer to Subject
- **value** -- new value

int32_t lv_subject_get_int(*lv_subject_t* *subject)

Get current value of an integer Subject.

Parametri

subject -- pointer to Subject

Ritorna

current value

int32_t lv_subject_get_previous_int(*lv_subject_t* *subject)

Get previous value of an integer Subject.

Parametri

subject -- pointer to Subject

Ritorna

current value

void lv_subject_set_min_value_int(*lv_subject_t* *subject, int32_t min_value)

Set a minimum value for an integer subject

Parametri

- **subject** -- pointer to Subject
- **min_value** -- the minimum value

void lv_subject_set_max_value_int(*lv_subject_t* *subject, int32_t max_value)

Set a maximum value for an integer subject

Parametri

- **subject** -- pointer to Subject
- **max_value** -- the maximum value

void lv_subject_init_float(*lv_subject_t* *subject, float value)

Initialize an float-type Subject.

Parametri

- **subject** -- pointer to Subject
- **value** -- initial value

void lv_subject_set_float(*lv_subject_t* *subject, float value)

Set value of an float Subject and notify Observers.

Parametri

- **subject** -- pointer to Subject
- **value** -- new value

`float lv_subject_get_float(lv_subject_t *subject)`

Get current value of an float Subject.

Parametri

- **subject** -- pointer to Subject

Ritorna

current value

`float lv_subject_get_previous_float(lv_subject_t *subject)`

Get previous value of an float Subject.

Parametri

- **subject** -- pointer to Subject

Ritorna

current value

`void lv_subject_set_min_value_float(lv_subject_t *subject, float min_value)`

Set a minimum value for a float subject

Parametri

- **subject** -- pointer to Subject
- **min_value** -- the minimum value

`void lv_subject_set_max_value_float(lv_subject_t *subject, float max_value)`

Set a maximum value for a float subject

Parametri

- **subject** -- pointer to Subject
- **max_value** -- the maximum value

`void lv_subject_init_string(lv_subject_t *subject, char *buf, char *prev_buf, size_t size, const char *value)`

Initialize a string-type Subject.

 **Nota**

A string Subject stores its own copy of the string, not just the pointer.

Parametri

- **subject** -- pointer to Subject
- **buf** -- pointer to buffer to store string
- **prev_buf** -- pointer to buffer to store previous string; can be NULL if not used
- **size** -- size of buffer(s)
- **value** -- initial value of string, e.g. "hello"

`void lv_subject_copy_string(lv_subject_t *subject, const char *buf)`

Copy a string to a Subject and notify Observers if it changed.

Parametri

- **subject** -- pointer to Subject
- **buf** -- new string

`void lv_subject_snprintf(lv_subject_t *subject, const char *format, ...)`

Format a new string, updating Subject, and notify Observers if it changed.

Parametri

- **subject** -- pointer to Subject
- **format** -- format string

`const char *lv_subject_get_string(lv_subject_t *subject)`

Get current value of a string Subject.

Parametri

- **subject** -- pointer to Subject

Ritorna

pointer to buffer containing current value

```
const char *lv_subject_get_previous_string(lv_subject_t *subject)
```

Get previous value of a string Subject.

 **Nota**

NULL will be returned if NULL was passed in `lv_subject_init_string()` as prev_buf.

Parametri

- **subject** -- pointer to Subject

Ritorna

pointer to buffer containing previous value

```
void lv_subject_init_pointer(lv_subject_t *subject, void *value)
```

Initialize a pointer-type Subject.

Parametri

- **subject** -- pointer to Subject
- **value** -- initial value

```
void lv_subject_set_pointer(lv_subject_t *subject, void *ptr)
```

Set value of a pointer Subject and notify Observers (regardless of whether it changed).

Parametri

- **subject** -- pointer to Subject
- **ptr** -- new value

```
const void *lv_subject_get_pointer(lv_subject_t *subject)
```

Get current value of a pointer Subject.

Parametri

- **subject** -- pointer to Subject

Ritorna

current value

```
const void *lv_subject_get_previous_pointer(lv_subject_t *subject)
```

Get previous value of a pointer Subject.

Parametri

- **subject** -- pointer to Subject

Ritorna

previous value

```
void lv_subject_init_color(lv_subject_t *subject, lv_color_t color)
```

Initialize a color-type Subject.

Parametri

- **subject** -- pointer to Subject
- **color** -- initial value

```
void lv_subject_set_color(lv_subject_t *subject, lv_color_t color)
```

Set value of a color Subject and notify Observers if it changed.

Parametri

- **subject** -- pointer to Subject
- **color** -- new value

```
lv_color_t lv_subject_get_color(lv_subject_t *subject)
```

Get current value of a color Subject.

Parametri

- **subject** -- pointer to Subject

Ritorna

current value

`lv_color_t lv_subject_get_previous_color(lv_subject_t *subject)`

Get previous value of a color Subject.

Parametri

`subject` -- pointer to Subject

Ritorna

previous value

`void lv_subject_init_group(lv_subject_t *group_subject, lv_subject_t *list[], uint32_t list_len)`

Initialize a Group-type Subject.

Parametri

- `group_subject` -- pointer to Group-type Subject
- `list` -- list of other Subject addresses; when any of these have values updated, Observers of `group_subject` will be notified.
- `list_len` -- number of elements in `list[]`

`void lv_subject_deinit(lv_subject_t *subject)`

Remove all Observers from a Subject and free allocated memory, and delete any associated Widget-Binding events. This leaves `subject` "disconnected" from all Observers and all associated Widget events established through Widget Binding.

Nota

This can safely be called regardless of whether any Observers added with `lv_subject_add_observer_obj()` or bound to a Widget Property with one of the `..._bind_...` functions.

Parametri

`subject` -- pointer to Subject

`lv_subject_t *lv_subject_get_group_element(lv_subject_t *subject, int32_t index)`

Get an element from Subject Group's list.

Parametri

- `subject` -- pointer to Group-type Subject
- `index` -- index of element to get

Ritorna

pointer to indexed Subject from list, or NULL if index is out of bounds

`lv_observer_t *lv_subject_add_observer(lv_subject_t *subject, lv_observer_cb_t observer_cb, void *user_data)`

Add Observer to Subject. When Subject's value changes `observer_cb` will be called.

Parametri

- `subject` -- pointer to Subject
- `observer_cb` -- notification callback
- `user_data` -- optional user data

Ritorna

pointer to newly-created Observer

`lv_observer_t *lv_subject_add_observer_obj(lv_subject_t *subject, lv_observer_cb_t observer_cb, lv_obj_t *obj, void *user_data)`

Add Observer to Subject for a Widget. When the Widget is deleted, Observer will be unsubscribed from Subject automatically.

Nota

Do not call `lv_observer_remove()` on Observers created this way. Only clean up such Observers by either:

- deleting the Widget, or
- calling `lv_subject_deinit()` to gracefully de-couple and remove all Observers.

Parametri

- **subject** -- pointer to Subject
- **observer_cb** -- notification callback
- **obj** -- pointer to Widget
- **user_data** -- optional user data

Ritorna

pointer to newly-created Observer

```
lv_observer_t *lv_subject_add_observer_with_target(lv_subject_t *subject, lv_observer_cb_t observer_cb,
                                                void *target, void *user_data)
```

Add an Observer to a Subject and also save a target pointer.

Parametri

- **subject** -- pointer to Subject
- **observer_cb** -- notification callback
- **target** -- any pointer (NULL is okay)
- **user_data** -- optional user data

Ritorna

pointer to newly-created Observer

```
void lv_observer_remove(lv_observer_t *observer)
```

Remove Observer from its Subject.

Parametri

observer -- pointer to Observer

```
void lv_obj_remove_from_subject(lv_obj_t *obj, lv_subject_t *subject)
```

Remove Observers associated with Widget **obj** from specified **subject** or all Subjects.

 **Nota**

This function can be used e.g. when a Widget's Subject(s) needs to be replaced by other Subject(s)

Parametri

- **obj** -- pointer to Widget whose Observers should be removed
- **subject** -- Subject to remove Widget from, or NULL to remove from all Subjects

```
void *lv_observer_get_target(lv_observer_t *observer)
```

Get target of an Observer.

Parametri

observer -- pointer to Observer

Ritorna

pointer to saved target

```
lv_obj_t *lv_observer_get_target_obj(lv_observer_t *observer)
```

Get target Widget of Observer. This is the same as `lv_observer_get_target()`, except it returns **target** as an `lv_obj_t *`.

Parametri

observer -- pointer to Observer

Ritorna

pointer to saved Widget target

```
void *lv_observer_get_user_data(const lv_observer_t *observer)
```

Get Observer's user data.

Parametri

observer -- pointer to Observer

Ritorna

void pointer to saved user data

```
void lv_subject_notify(lv_subject_t *subject)
```

Notify all Observers of Subject.

Parametri

- **subject** -- pointer to Subject

```
lv_subject_increment_dsc_t *lv_obj_add_subject_increment_event(lv_obj_t *obj, lv_subject_t *subject,
                                                               lv_event_code_t trigger, int32_t step)
```

Add an event handler to increment (or decrement) the value of a subject on a trigger.

Parametri

- **obj** -- pointer to a widget
- **subject** -- pointer to a subject to change
- **trigger** -- the trigger on which the subject should be changed
- **step** -- value to add on trigger if the minimum value is reached, the maximum value will be set on rollover.

```
void lv_obj_set_subject_increment_event_min_value(lv_obj_t *obj, lv_subject_increment_dsc_t *dsc,
                                                 int32_t min_value)
```

Set the minimum subject value to set by the event

Parametri

- **obj** -- pointer to the Widget to which the event is attached
- **dsc** -- pointer to the descriptor returned by `lv_obj_add_subject_increment_event()`
- **min_value** -- the minimum value to set

```
void lv_obj_set_subject_increment_event_max_value(lv_obj_t *obj, lv_subject_increment_dsc_t *dsc,
                                                 int32_t max_value)
```

Set the maximum subject value to set by the event

Parametri

- **obj** -- pointer to the Widget to which the event is attached
- **dsc** -- pointer to the descriptor returned by `lv_obj_add_subject_increment_event()`
- **max_value** -- the maximum value to set

```
void lv_obj_set_subject_increment_event_rollover(lv_obj_t *obj, lv_subject_increment_dsc_t *dsc, bool
                                                rollover)
```

Set what to do when the min/max value is crossed.

Nota

the subject also can have min/max values and always the smaller range will be considered

Parametri

- **obj** -- pointer to the Widget to which the event is attached
- **dsc** -- pointer to the descriptor returned by `lv_obj_add_subject_increment_event()`
- **rollover** -- false: stop at the min/max value; true: jump to the other end

```
void lv_obj_add_subject_toggle_event(lv_obj_t *obj, lv_subject_t *subject, lv_event_code_t trigger)
```

Toggle the value of an integer subject on an event. If it was != 0 it will be 0. If it was 0, it will be 1.

Parametri

- **obj** -- pointer to a widget
- **subject** -- pointer to a subject to toggle
- **trigger** -- the trigger on which the subject should be changed

```
void lv_obj_add_subject_set_int_event(lv_obj_t *obj, lv_subject_t *subject, lv_event_code_t trigger, int32_t
                                       value)
```

Set the value of an integer subject.

Parametri

- **obj** -- pointer to a widget
- **subject** -- pointer to a subject to change
- **trigger** -- the trigger on which the subject should be changed

- **value** -- the value to set

```
void lv_obj_add_subject_set_float_event(lv_obj_t *obj, lv_subject_t *subject, lv_event_code_t trigger, float value)
```

Set the value of a float subject.

Parametri

- **obj** -- pointer to a widget
- **subject** -- pointer to a subject to change
- **trigger** -- the trigger on which the subject should be changed
- **value** -- the value to set

```
void lv_obj_add_subject_set_string_event(lv_obj_t *obj, lv_subject_t *subject, lv_event_code_t trigger, const char *value)
```

Set the value of a string subject.

Parametri

- **obj** -- pointer to a widget
- **subject** -- pointer to a subject to change
- **trigger** -- the trigger on which the subject should be changed
- **value** -- the value to set

```
lv_observer_t *lv_obj_bind_style(lv_obj_t *obj, const lv_style_t *style, lv_style_selector_t selector, lv_subject_t *subject, int32_t ref_value)
```

Disable a style if a subject's value is not equal to a reference value

Parametri

- **obj** -- pointer to Widget
- **style** -- pointer to a style
- **selector** -- pointer to a selector
- **subject** -- pointer to Subject
- **ref_value** -- reference value to compare Subject's value with

Ritorna

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_flag_if_eq(lv_obj_t *obj, lv_subject_t *subject, lv_obj_flag_t flag, int32_t ref_value)
```

Set Widget's flag(s) if an integer Subject's value is equal to a reference value, clear flag otherwise.

Parametri

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **flag** -- flag(s) (can be bit-wise OR-ed) to set or clear (e.g. LV_OBJ_FLAG_HIDDEN)
- **ref_value** -- reference value to compare Subject's value with

Ritorna

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_flag_if_not_eq(lv_obj_t *obj, lv_subject_t *subject, lv_obj_flag_t flag, int32_t ref_value)
```

Set Widget's flag(s) if an integer Subject's value is not equal to a reference value, clear flag otherwise.

Parametri

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **flag** -- flag(s) (can be bit-wise OR-ed) to set or clear (e.g. LV_OBJ_FLAG_HIDDEN)
- **ref_value** -- reference value to compare Subject's value with

Ritorna

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_flag_if_gt(lv_obj_t *obj, lv_subject_t *subject, lv_obj_flag_t flag, int32_t ref_value)
```

Set Widget's flag(s) if an integer Subject's value is greater than a reference value, clear flag otherwise.

Parametri

- **obj** -- pointer to Widget

- **subject** -- pointer to Subject
- **flag** -- flag(s) (can be bit-wise OR-ed) to set or clear (e.g. LV_OBJ_FLAG_HIDDEN)
- **ref_value** -- reference value to compare Subject's value with

Ritorna

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_flag_if_ge(lv_obj_t *obj, lv_subject_t *subject, lv_obj_flag_t flag, int32_t ref_value)
```

Set Widget's flag(s) if an integer Subject's value is greater than or equal to a reference value, clear flag otherwise.

Parametri

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **flag** -- flag(s) (can be bit-wise OR-ed) to set or clear (e.g. LV_OBJ_FLAG_HIDDEN)
- **ref_value** -- reference value to compare Subject's value with

Ritorna

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_flag_if_lt(lv_obj_t *obj, lv_subject_t *subject, lv_obj_flag_t flag, int32_t ref_value)
```

Set Widget's flag(s) if an integer Subject's value is less than a reference value, clear flag otherwise.

Parametri

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **flag** -- flag(s) (can be bit-wise OR-ed) to set or clear (e.g. LV_OBJ_FLAG_HIDDEN)
- **ref_value** -- reference value to compare Subject's value with

Ritorna

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_flag_if_le(lv_obj_t *obj, lv_subject_t *subject, lv_obj_flag_t flag, int32_t ref_value)
```

Set Widget's flag(s) if an integer Subject's value is less than or equal to a reference value, clear flag otherwise.

Parametri

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **flag** -- flag(s) (can be bit-wise OR-ed) to set or clear (e.g. LV_OBJ_FLAG_HIDDEN)
- **ref_value** -- reference value to compare Subject's value with

Ritorna

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_state_if_eq(lv_obj_t *obj, lv_subject_t *subject, lv_state_t state, int32_t ref_value)
```

Set Widget's state(s) if an integer Subject's value is equal to a reference value, clear flag otherwise.

Parametri

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **state** -- state(s) (can be bit-wise OR-ed) to set or clear (e.g. LV_STATE_CHECKED)
- **ref_value** -- reference value to compare Subject's value with

Ritorna

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_state_if_not_eq(lv_obj_t *obj, lv_subject_t *subject, lv_state_t state, int32_t ref_value)
```

Set a Widget's state(s) if an integer Subject's value is not equal to a reference value, clear flag otherwise

Parametri

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **state** -- state(s) (can be bit-wise OR-ed) to set or clear (e.g. LV_STATE_CHECKED)
- **ref_value** -- reference value to compare Subject's value with

Ritorna

pointer to newly-created Observer

`lv_observer_t *lv_obj_bind_state_if_gt(lv_obj_t *obj, lv_subject_t *subject, lv_state_t state, int32_t ref_value)`

Set Widget's state(s) if an integer Subject's value is greater than a reference value, clear flag otherwise.

Parametri

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **state** -- state(s) (can be bit-wise OR-ed) to set or clear (e.g. LV_STATE_CHECKED)
- **ref_value** -- reference value to compare Subject's value with

Ritorna

pointer to newly-created Observer

`lv_observer_t *lv_obj_bind_state_if_ge(lv_obj_t *obj, lv_subject_t *subject, lv_state_t state, int32_t ref_value)`

Set Widget's state(s) if an integer Subject's value is greater than or equal to a reference value, clear flag otherwise.

Parametri

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **state** -- state(s) (can be bit-wise OR-ed) to set or clear (e.g. LV_STATE_CHECKED)
- **ref_value** -- reference value to compare Subject's value with

Ritorna

pointer to newly-created Observer

`lv_observer_t *lv_obj_bind_state_if_lt(lv_obj_t *obj, lv_subject_t *subject, lv_state_t state, int32_t ref_value)`

Set Widget's state(s) if an integer Subject's value is less than a reference value, clear flag otherwise.

Parametri

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **state** -- state(s) (can be bit-wise OR-ed) to set or clear (e.g. LV_STATE_CHECKED)
- **ref_value** -- reference value to compare Subject's value with

Ritorna

pointer to newly-created Observer

`lv_observer_t *lv_obj_bind_state_if_le(lv_obj_t *obj, lv_subject_t *subject, lv_state_t state, int32_t ref_value)`

Set Widget's state(s) if an integer Subject's value is less than or equal to a reference value, clear flag otherwise.

Parametri

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **state** -- state(s) (can be bit-wise OR-ed) to set or clear (e.g. LV_STATE_CHECKED)
- **ref_value** -- reference value to compare Subject's value with

Ritorna

pointer to newly-created Observer

`lv_observer_t *lv_obj_bind_checked(lv_obj_t *obj, lv_subject_t *subject)`

Set an integer Subject to 1 when a Widget is checked and set it 0 when unchecked, and clear Widget's checked state when Subject's value changes to 0 and set it when non-zero.

Nota

Ensure Widget's LV_OBJ_FLAG_CHECKABLE flag is set.

Parametri

- **obj** -- pointer to Widget
- **subject** -- pointer to a Subject

Ritorna

pointer to newly-created Observer

union `lv_subject_value_t`

#include <lv_observer.h> A common type to handle all the various observable types in the same way

Membri Pubblici

int32_t num
Integer number (opacity, enums, booleans or "normal" numbers)

const void *pointer
Constant pointer (string buffer, format string, font, cone text, etc.)

lv_color_t color
Color

float float_v
Floating point value

struct lv_subject_t
#include <lv_observer.h> The Subject (an observable value)

Membri Pubblici

lv_ll_t subs_ll
Subscribers

lv_subject_value_t value
Current value

lv_subject_value_t prev_value
Previous value

lv_subject_value_t min_value
Minimum value for min. int or float

lv_subject_value_t max_value
Maximum value for max. int or float

void *user_data
Additional parameter, can be used freely by user

uint32_t type
One of the LV SUBJECT_TYPE_... values

uint32_t size
String buffer size or group length

uint32_t notify_restart_query
If an Observer was deleted during notification, start notifying from the beginning.

14.12.20 lv_observer_private.h

struct _lv_observer_t

#include <lv_observer_private.h> The observer object: a descriptor returned when subscribing LVGL widgets to subjects

Membri Pubblici***lv_subject_t* *subject**

Observed subject

***lv_observer_cb_t* cb**

Callback that notifies when value changes

void *target

A target for the observer, e.g. a widget or any pointer

void *user_data

Additional parameter supplied when subscribing

uint32_t auto_free_user_data

Automatically free user data when observer is removed

uint32_t notified

Was observer already notified?

uint32_t for_objIs target a pointer to a Widget (*lv_obj_t* *)?**struct _lv_subject_increment_dsc_t**#include <lv_observer_private.h> Descriptor created by *lv_obj_add_subject_increment_event()***Membri Pubblici*****lv_subject_t* *subject**

The subject to adjust

int32_t step

The step add to the subject

bool rollover

Where to start over from the other end when one end is exceeded

int32_t min_value

Don't set a value smaller than this

int32_t max_value

Don't set a value larger than this

14.12.21 lv_refr.h

Funzioni**void lv_refr_now(lv_display_t *disp)**Redraw the invalidated areas now. Normally the redrawing is periodically executed in *lv_timer_handler* but a long blocking process can prevent the call of *lv_timer_handler*. In this case if the GUI is updated in the process (e.g. progress bar) this function can be called when the screen should be updated.

Parametri

disp -- pointer to display to refresh. NULL to refresh all displays.

void lv_obj_redraw(lv_layer_t *layer, lv_obj_t *obj)

Redrawn on object and all its children using the passed draw context

Parametri

- **layer** -- pointer to a layer where to draw.
- **obj** -- the start object from the redraw should start

void lv_display_refr_timer(lv_timer_t *timer)

Called periodically to handle the refreshing

Parametri

timer -- pointer to the timer itself, or NULL

14.12.22 lv_refr_private.h

Funzioni

void lv_refr_init(void)

Initialize the screen refresh subsystem

void lv_refr_deinit(void)

Deinitialize the screen refresh subsystem

void lv_inv_area(lv_display_t *disp, const lv_area_t *area_p)

Invalidate an area on display to redraw it

Parametri

- **area_p** -- pointer to area which should be invalidated (NULL: delete the invalidated areas)
- **disp** -- pointer to display where the area should be invalidated (NULL can be used if there is only one display)

lv_display_t *lv_refr_get_disp_refreshing(void)

Get the display which is being refreshed

Ritorna

the display being refreshed

void lv_refr_set_disp_refreshing(lv_display_t *disp)

Set the display which is being refreshed

Parametri

disp -- the display being refreshed

lv_obj_t *lv_refr_get_top_obj(const lv_area_t *area_p, lv_obj_t *obj)

Search the most top object which fully covers an area

Parametri

- **area_p** -- pointer to an area
- **obj** -- the first object to start the searching (typically a screen)

Ritorna

void lv_obj_refr(lv_layer_t *layer, lv_obj_t *obj)

Render an object to a layer

Parametri

- **layer** -- target drawing layer
- **obj** -- object to render

14.13 debugging

14.13.1 monkey

lv_monkey.h**Typedef**

```
typedef struct _lv_monkey_t lv_monkey_t
```

Funzioni

void lv_monkey_config_init(lv_monkey_config_t *config)

Initialize a monkey config with default values

Parametri

config -- pointer to 'lv_monkey_config_t' variable to initialize

lv_monkey_t *lv_monkey_create(const lv_monkey_config_t *config)

Create monkey for test

Parametri

config -- pointer to 'lv_monkey_config_t' variable

Ritorna

pointer to the created monkey

lv_indev_t *lv_monkey_get_indev(lv_monkey_t *monkey)

Get monkey input device

Parametri

monkey -- pointer to a monkey

Ritorna

pointer to the input device

void lv_monkey_set_enable(lv_monkey_t *monkey, bool en)

Enable monkey

Parametri

- **monkey** -- pointer to a monkey
- **en** -- set to true to enable

bool lv_monkey_get_enable(lv_monkey_t *monkey)

Get whether monkey is enabled

Parametri

monkey -- pointer to a monkey

Ritorna

return true if monkey enabled

void lv_monkey_set_user_data(lv_monkey_t *monkey, void *user_data)

Set the user_data field of the monkey

Parametri

- **monkey** -- pointer to a monkey
- **user_data** -- pointer to the new user_data.

void *lv_monkey_get_user_data(lv_monkey_t *monkey)

Get the user_data field of the monkey

Parametri

monkey -- pointer to a monkey

Ritorna

the pointer to the user_data of the monkey

void lv_monkey_delete(lv_monkey_t *monkey)

Delete monkey

Parametri

monkey -- pointer to monkey

struct lv_range_t

Membri Pubblici

```
int32_t min
```

```
int32_t max
```

```
struct lv_urange_t
```

Membri Pubblici

```
uint32_t min
```

```
uint32_t max
```

```
struct _lv_monkey_config_t
```

Membri Pubblici

lv_indev_type_t type

Input device type

lv_urange_t period_range

Monkey execution period

lv_range_t input_range

The range of input value

lv_monkey_private.h

14.13.2 sysmon

lv_sysmon.h

Funzioni

lv_obj_t *lv_sysmon_create(lv_display_t *disp)

Create a new system monitor label

Parametri

disp -- create the sys. mon. on this display's system layer

Ritorna

the create label

void lv_sysmon_show_performance(lv_display_t *disp)

Show system performance monitor: CPU usage and FPS count

Parametri

disp -- target display, NULL: use the default displays

void lv_sysmon_hide_performance(lv_display_t *disp)

Hide system performance monitor

Parametri

disp -- target display, NULL: use the default

`void lv_sysmon_performance_dump(lv_display_t *disp)`
 Dump the FPS data recorded between the last and current dump call.

Parametri

`disp` -- target display, NULL: use the default

`void lv_sysmon_performance_resume(lv_display_t *disp)`
 Resume the system performance monitor.

Parametri

`disp` -- target display, NULL: use the default

`void lv_sysmon_performance_pause(lv_display_t *disp)`
 Pause the system performance monitor.

Nota

When the sysmon is stopped you can use `lv_sysmon_dump_performance` to get performance information.
 See `lv_sysmon_dump_performance` for more information.

Parametri

`disp` -- target display, NULL: use the default

`void lv_sysmon_show_memory(lv_display_t *disp)`
 Show system memory monitor: used memory and the memory fragmentation

Parametri

`disp` -- target display, NULL: use the default displays

`void lv_sysmon_hide_memory(lv_display_t *disp)`
 Hide system memory monitor
Parametri
`disp` -- target display, NULL: use the default displays

lv_sysmon_private.h**Funzioni**

`void lv_sysmon_builtin_init(void)`
 Initialize built-in system monitor, such as performance and memory monitor.

`void lv_sysmon_builtin_deinit(void)`
 DeInitialize built-in system monitor, such as performance and memory monitor.

`struct _lv_sysmon_backend_data_t`

Membri Pubblici

`lv_subject_t subject`

`lv_timer_t *timer`

`struct _lv_sysmon_perf_info_t`

Membri Pubblici

`bool init`

```
uint32_t refr_start  
  
uint32_t refr_interval_sum  
  
uint32_t refr_elaps_sum  
  
uint32_t refr_cnt  
  
uint32_t render_start  
  
uint32_t render_elaps_sum  
  
uint32_t render_cnt  
  
uint32_t flush_in_render_start  
  
uint32_t flush_in_render_elaps_sum  
  
uint32_t flush_not_in_render_start  
  
uint32_t flush_not_in_render_elaps_sum  
  
uint32_t last_report_timestamp  
  
uint32_t render_in_progress  
  
struct lv_sysmon_perf_info_t measured  
  
uint32_t fps  
  
uint32_t cpu  
  
uint32_t refr_avg_time  
  
uint32_t render_avg_time  
    Pure rendering time without flush time  
  
uint32_t flush_avg_time  
    Pure flushing time without rendering time  
  
uint32_t cpu_avg_total  
  
uint32_t fps_avg_total  
  
uint32_t run_cnt  
  
struct lv_sysmon_perf_info_t calculated
```

14.13.3 test

lv_test.h

lv_test_display.h

Funzioni

`lv_display_t *lv_test_display_create(int32_t hor_res, int32_t ver_res)`

lv_test_helpers.h

Define

`LV_HEAP_CHECK(x)`

`LV_TEST_WIDTH_TO_STRIDE(w, px_size)`

Funzioni

`void lv_test_wait(uint32_t ms)`

Emulate a delay. It's not real delay, but it tricks LVGL to think that the required time has been elapsed. `lv_timer_handler` is called after each millisecond, meaning all the events will be fired inside this function. At the end the animations and display will be also updated.

Parametri

`ms` -- the number of milliseconds to pass

`void lv_test_fast_forward(uint32_t ms)`

Emulates some time passing. Update the animations and the display only once at the end.

Parametri

`ms` -- the number of milliseconds to pass

`static inline size_t lv_test_get_free_mem(void)`

lv_test_indev.h

Funzioni

`void lv_test_indev_create_all(void)`

Create a mouse (pointer), keypad, and encoder indevs. They can be controlled via function calls during the test

`void lv_test_indev_delete_all(void)`

Delete all test input devices

`lv_indev_t *lv_test_indev_get_indev(lv_indev_type_t type)`

Get one of the indev created in `lv_test_indev_create_all`

Parametri

`type` -- type of the indev to get

Ritorna

the indev

`void lv_test_mouse_move_to(int32_t x, int32_t y)`

Move the mouse to the given coordinates. This function doesn't wait, but just changes the state and returns immediately.

Parametri

- `x` -- the target absolute X coordinate
- `y` -- the target absolute Y coordinate

`void lv_test_mouse_move_to_obj(lv_obj_t *obj)`

Move the mouse to the center of a widget This function doesn't wait, but just changes the state and returns immediately.

Parametri

- **obj** -- pointer to an widget

void lv_test_mouse_move_by(int32_t x, int32_t y)

Move the mouse cursor. Keep the pressed or released state This function doesn't wait, but just changes the state and returns immediately.

Parametri

- **x** -- the difference in X to move
- **y** -- the difference in Y to move

void lv_test_mouse_press(void)

Make the mouse button pressed. This function doesn't wait, but just changes the state and returns immediately.

void lv_test_mouse_release(void)

Make the mouse button released. This function doesn't wait, but just changes the state and returns immediately.

void lv_test_mouse_click_at(int32_t x, int32_t y)

Emulate a click on a given point. First set the released state, wait a little, press, wait, and release again. The wait time is 50ms. Internally `lv_timer_handler` is called, meaning all the events will be fired inside this function.

Parametri

- **x** -- the target absolute X coordinate
- **y** -- the target absolute Y coordinate

void lv_test_key_press(uint32_t k)

Emulate a key press. This function doesn't wait, but just changes the state and returns immediately.

Parametri

- **k** -- the key to press

void lv_test_key_release(void)

Release the previously press key. This function doesn't wait, but just changes the state and returns immediately.

Parametri

- **k** -- the key to press

void lv_test_key_hit(uint32_t k)

Emulate a key hit. First set the released state, wait a little, press, wait, and release again. The wait time is 50ms. Internally `lv_timer_handler` is called, meaning all the events will be fired inside this function.

Parametri

- **k** -- the key to hit

void lv_test_encoder_add_diff(int32_t d)

Emulate encoder rotation, use positive parameter to rotate to the right and negative to rotate to the left. This function doesn't wait, but just changes the state and returns immediately.

Parametri

- **d** -- number of encoder ticks to emulate

void lv_test_encoder_turn(int32_t d)

Emulate an encoder turn a wait 50ms. Use positive parameter to rotate to the right and negative to rotate to the left. Internally `lv_timer_handler` is called, meaning all the events will be fired inside this function.

Parametri

- **d** -- number of encoder ticks to emulate

void lv_test_encoder_press(void)

Emulate an encoder press. This function doesn't wait, but just changes the state and returns immediately.

void lv_test_encoder_release(void)

Emulate an encoder release. This function doesn't wait, but just changes the state and returns immediately.

void lv_test_encoder_click(void)

Emulate am encoder click. First set the released state, wait a little, press, wait, and release again. The wait time is 50ms. Internally `lv_timer_handler` is called, meaning all the events will be fired inside this function.

lv_test_indev_gesture.h**Funzioni**

void lv_test_indev_gesture_create(void)

Create a touch (pointer) indevs. They can be controlled via function calls during the test

void lv_test_indev_gesture_delete(void)

Delete the touch (pointer) indevs.

lv_indev_t *lv_test_indev_get_gesture_indev(lv_indev_type_t type)

Get one of the indev created in lv_test_indev_gesture_create

Parametri

- **type** -- type of the indev to get

Ritorna

the indev

void lv_test_gesture_set_pinch_data(lv_point_t point_0, lv_point_t point_1)

Set two touch points data for pinch gesture

Parametri

- **point_0** -- First touch point coordinates
- **point_1** -- Second touch point coordinates

void lv_test_gesture_pinch_press(void)

Trigger press state of pinch gesture (both touch points pressed)

void lv_test_gesture_pinch_release(void)

Trigger release state of pinch gesture (both touch points released)

void lv_test_gesture_pinch(lv_point_t point_begin_0, lv_point_t point_begin_1, lv_point_t point_end_0, lv_point_t point_end_1)

Simulate a complete pinch gesture operation

Parametri

- **point_begin_0** -- Starting coordinates of first touch point
- **point_begin_1** -- Starting coordinates of second touch point
- **point_end_0** -- Ending coordinates of first touch point
- **point_end_1** -- Ending coordinates of second touch point

lv_test_private.h

struct lv_test_state_t

Membri Pubblici

lv_indev_t *mouse_indev

lv_indev_t *keypad_indev

lv_indev_t *encoder_indev

lv_draw_buf_t draw_buf

int32_t x_act

int32_t y_act

```

uint32_t key_act

int32_t diff_act

bool mouse_pressed

bool key_pressed

bool enc_pressed

lv_indev_t *gesture_indev

lv_indev_touch_data_t *touch_data

uint8_t max_touch_cnt

```

lv_test_screenshot_compare.h

Funzioni

bool **lv_test_screenshot_compare**(const char *fn_ref)

Compare the current content of the test screen with a reference PNG image

- If the reference image is not found it will be created automatically from the rendered screen.
- If the compare fails an <image_name>_err.png file will be created with the rendered content next to the reference image.

It requires libPNG.

Nota

This function assumes that the default display is the test display that was created by `lv_test_display_create()`

Parametri

`fn_ref` -- path to the reference image. Will be appended to `REF_IMGS_PATH` if set.

Ritorna

true: the reference image and the display are the same; false: they are different (<image_name>_err.png is created).

14.14 display

14.14.1 lv_display.h

Define

LV_HOR_RES

The horizontal resolution of the currently active display.

LV_VER_RES

The vertical resolution of the currently active display.

LV_DPX_CALC(dpi, n)

See [lv_dpx\(\)](#) and [lv_display_dpx\(\)](#). Same as Android's DIP. (Different name is chosen to avoid mistype between LV_DPI and LV_DIP)

- 40 dip is 40 px on a 160 DPI screen (distance = 1/4 inch).
- 40 dip is 80 px on a 320 DPI screen (distance still = 1/4 inch).

Vedi anche

<https://stackoverflow.com/questions/2025282/what-is-the-difference-between-px-dip-dp-and-sp>

LV_DPX(n)**Typedef**

```
typedef void (*lv_display_flush_cb_t)(lv_display_t *disp, const lv_area_t *area, uint8_t *px_map)
```

```
typedef void (*lv_display_flush_wait_cb_t)(lv_display_t *disp)
```

Enumenum **lv_display_rotation_t**

Values:

enumerator **LV_DISPLAY_ROTATION_0**

enumerator **LV_DISPLAY_ROTATION_90**

enumerator **LV_DISPLAY_ROTATION_180**

enumerator **LV_DISPLAY_ROTATION_270**

enum **lv_display_render_mode_t**

Values:

enumerator **LV_DISPLAY_RENDER_MODE_PARTIAL**

Use the buffer(s) to render the screen in smaller parts. This way the buffers can be smaller than the display to save RAM. At least 1/10 screen size buffer(s) are recommended.

enumerator **LV_DISPLAY_RENDER_MODE_DIRECT**

The buffer(s) has to be screen sized and LVGL will render into the correct location of the buffer. This way the buffer always contains the whole image. Only the changed areas will be updated. With 2 buffers the buffers' content are kept in sync automatically and in flush_cb only address change is required.

enumerator **LV_DISPLAY_RENDER_MODE_FULL**

Always redraw the whole screen even if only one pixel has been changed. With 2 buffers in flush_cb only an address change is required.

enum **lv_screen_load_anim_t**

Values:

enumerator **LV_SCREEN_LOAD_ANIM_NONE**

enumerator **LV_SCREEN_LOAD_ANIM_OVER_LEFT**

enumerator **LV_SCREEN_LOAD_ANIM_OVER_RIGHT**

enumerator **LV_SCREEN_LOAD_ANIM_OVER_TOP**

enumerator **LV_SCREEN_LOAD_ANIM_OVER_BOTTOM**

enumerator **LV_SCREEN_LOAD_ANIM_MOVE_LEFT**

enumerator **LV_SCREEN_LOAD_ANIM_MOVE_RIGHT**

enumerator **LV_SCREEN_LOAD_ANIM_MOVE_TOP**

enumerator **LV_SCREEN_LOAD_ANIM_MOVE_BOTTOM**

enumerator **LV_SCREEN_LOAD_ANIM_FADE_IN**

enumerator **LV_SCREEN_LOAD_ANIM_FADE_ON**

enumerator **LV_SCREEN_LOAD_ANIM_FADE_OUT**

enumerator **LV_SCREEN_LOAD_ANIM_OUT_LEFT**

enumerator **LV_SCREEN_LOAD_ANIM_OUT_RIGHT**

enumerator **LV_SCREEN_LOAD_ANIM_OUT_TOP**

enumerator **LV_SCREEN_LOAD_ANIM_OUT_BOTTOM**

Funzioni

`lv_display_t *lv_display_create(int32_t hor_res, int32_t ver_res)`

Create a new display with the given resolution

Parametri

- **hor_res** -- horizontal resolution in pixels
- **ver_res** -- vertical resolution in pixels

Ritorna

pointer to a display object or NULL on error

`void lv_display_delete(lv_display_t *disp)`

Remove a display

Parametri

disp -- pointer to display

`void lv_display_set_default(lv_display_t *disp)`

Set a default display. The new screens will be created on it by default.

Parametri

- **disp** -- pointer to a display

`lv_display_t *lv_display_get_default(void)`

Get the default display

Ritorna

- pointer to the default display

`lv_display_t *lv_display_get_next(lv_display_t *disp)`

Get the next display.

Parametri

- **disp** -- pointer to the current display. NULL to initialize.

Ritorna

- the next display or NULL if no more. Gives the first display when the parameter is NULL.

`void lv_display_set_resolution(lv_display_t *disp, int32_t hor_res, int32_t ver_res)`

Sets the resolution of a display. LV_EVENT_RESOLUTION_CHANGED event will be sent. Here the native resolution of the device should be set. If the display will be rotated later with `lv_display_set_rotation` LVGL will swap the hor. and ver. resolution automatically.

Parametri

- **disp** -- pointer to a display
- **hor_res** -- the new horizontal resolution
- **ver_res** -- the new vertical resolution

`void lv_display_set_physical_resolution(lv_display_t *disp, int32_t hor_res, int32_t ver_res)`

It's not mandatory to use the whole display for LVGL, however in some cases physical resolution is important. For example the touchpad still sees whole resolution and the values needs to be converted to the active LVGL display area.

Parametri

- **disp** -- pointer to a display
- **hor_res** -- the new physical horizontal resolution, or -1 to assume it's the same as the normal hor. res.
- **ver_res** -- the new physical vertical resolution, or -1 to assume it's the same as the normal hor. res.

`void lv_display_set_offset(lv_display_t *disp, int32_t x, int32_t y)`

If physical resolution is not the same as the normal resolution the offset of the active display area can be set here.

Parametri

- **disp** -- pointer to a display
- **x** -- X offset
- **y** -- Y offset

`void lv_display_set_rotation(lv_display_t *disp, lv_display_rotation_t rotation)`

Set the rotation of this display. LVGL will swap the horizontal and vertical resolutions internally.

Parametri

- **disp** -- pointer to a display (NULL to use the default display)
- **rotation** -- LV_DISPLAY_ROTATION_0/90/180/270

`void lv_display_set_matrix_rotation(lv_display_t *disp, bool enable)`

Use matrix rotation for the display. This function is depended on LV_DRAW_TRANSFORM_USE_MATRIX

Parametri

- **disp** -- pointer to a display (NULL to use the default display)
- **enable** -- true: enable matrix rotation, false: disable

`void lv_display_set_dpi(lv_display_t *disp, int32_t dpi)`

Set the DPI (dot per inch) of the display. $dpi = \sqrt{hor_res^2 + ver_res^2} / \text{diagonal}$ "

Parametri

- **disp** -- pointer to a display
- **dpi** -- the new DPI

`int32_t lv_display_get_horizontal_resolution(const lv_display_t *disp)`

Get the horizontal resolution of a display.

Parametri

disp -- pointer to a display (NULL to use the default display)

Ritorna

 the horizontal resolution of the display.

`int32_t lv_display_get_vertical_resolution(const lv_display_t *disp)`

Get the vertical resolution of a display

Parametri

disp -- pointer to a display (NULL to use the default display)

Ritorna

 the vertical resolution of the display

`int32_t lv_display_get_original_horizontal_resolution(const lv_display_t *disp)`

Get the original horizontal resolution of a display without considering rotation

Parametri

disp -- pointer to a display (NULL to use the default display)

Ritorna

 the horizontal resolution of the display.

`int32_t lv_display_get_original_vertical_resolution(const lv_display_t *disp)`

Get the original vertical resolution of a display without considering rotation

Parametri

disp -- pointer to a display (NULL to use the default display)

Ritorna

 the vertical resolution of the display

`int32_t lv_display_get_physical_horizontal_resolution(const lv_display_t *disp)`

Get the physical horizontal resolution of a display

Parametri

disp -- pointer to a display (NULL to use the default display)

Ritorna

 the physical horizontal resolution of the display

`int32_t lv_display_get_physical_vertical_resolution(const lv_display_t *disp)`

Get the physical vertical resolution of a display

Parametri

disp -- pointer to a display (NULL to use the default display)

Ritorna

 the physical vertical resolution of the display

`int32_t lv_display_get_offset_x(const lv_display_t *disp)`

Get the horizontal offset from the full / physical display

Parametri

disp -- pointer to a display (NULL to use the default display)

Ritorna

 the horizontal offset from the physical display

`int32_t lv_display_get_offset_y(const lv_display_t *disp)`

Get the vertical offset from the full / physical display

Parametri

disp -- pointer to a display (NULL to use the default display)

Ritorna

 the horizontal offset from the physical display

`lv_display_rotation_t lv_display_get_rotation(lv_display_t *disp)`

Get the current rotation of this display.

Parametri

- **disp** -- pointer to a display (NULL to use the default display)

Ritorna

the current rotation

```
bool lv_display_get_matrix_rotation(lv_display_t *disp)
```

Get if matrix rotation is enabled for a display or not

Parametri

- **disp** -- pointer to a display (NULL to use the default display)

Ritorna

true: matrix rotation is enabled; false: disabled

```
int32_t lv_display_get_dpi(const lv_display_t *disp)
```

Get the DPI of the display

Parametri

- **disp** -- pointer to a display (NULL to use the default display)

Ritorna

dpi of the display

```
void lv_display_set_buffers(lv_display_t *disp, void *buf1, void *buf2, uint32_t buf_size,
                           lv_display_render_mode_t render_mode)
```

Set the buffers for a display, similarly to `lv_display_set_draw_buffers`, but accept the raw buffer pointers. For DIRECT/FULL rendering modes, the buffer size must be at least `hor_res * ver_res * lv_color_format_get_size(lv_display_get_color_format(disp))`

Parametri

- **disp** -- pointer to a display
- **buf1** -- first buffer
- **buf2** -- second buffer (can be NULL)
- **buf_size** -- buffer size in byte
- **render_mode** -- LV_DISPLAY_RENDER_MODE_PARTIAL/DIRECT/FULL

```
void lv_display_set_buffers_with_stride(lv_display_t *disp, void *buf1, void *buf2, uint32_t buf_size,
                                       uint32_t stride, lv_display_render_mode_t render_mode)
```

Set the frame buffers for a display, similarly to `lv_display_set_buffers`, but allow for a custom stride as required by a display controller. This allows the frame buffers to have a stride alignment different from the rest of the buffers @param disp pointer to a display @param buf1 first buffer @param buf2 second buffer (can be NULL)

Parametri

- **buf_size** -- buffer size in byte
- **stride** -- buffer stride in bytes
- **render_mode** -- LV_DISPLAY_RENDER_MODE_PARTIAL/DIRECT/FULL

```
void lv_display_set_draw_buffers(lv_display_t *disp, lv_draw_buf_t *buf1, lv_draw_buf_t *buf2)
```

Set the buffers for a display, accept a draw buffer pointer. Normally use `lv_display_set_buffers` is enough for most cases. Use this function when an existing `lv_draw_buf_t` is available.

Parametri

- **disp** -- pointer to a display
- **buf1** -- first buffer
- **buf2** -- second buffer (can be NULL)

```
void lv_display_set_3rd_draw_buffer(lv_display_t *disp, lv_draw_buf_t *buf3)
```

Set the third draw buffer for a display.

Parametri

- **disp** -- pointer to a display
- **buf3** -- third buffer

```
void lv_display_set_render_mode(lv_display_t *disp, lv_display_render_mode_t render_mode)
```

Set display render mode

Parametri

- **disp** -- pointer to a display

- **render_mode** -- LV_DISPLAY_RENDER_MODE_PARTIAL/DIRECT/FULL

void lv_display_set_flush_cb(lv_display_t *disp, *lv_display_flush_cb_t* flush_cb)

Set the flush callback which will be called to copy the rendered image to the display.

Parametri

- **disp** -- pointer to a display
- **flush_cb** -- the flush callback (px_map contains the rendered image as raw pixel map and it should be copied to area on the display)

void lv_display_set_flush_wait_cb(lv_display_t *disp, *lv_display_flush_wait_cb_t* wait_cb)

Set a callback to be used while LVGL is waiting flushing to be finished. It can do any complex logic to wait, including semaphores, mutexes, polling flags, etc. If not set the disp->flushing flag is used which can be cleared with *lv_display_flush_ready()*

Parametri

- **disp** -- pointer to a display
- **wait_cb** -- a callback to call while LVGL is waiting for flush ready. If NULL *lv_display_flush_ready()* can be used to signal that flushing is ready.

void lv_display_set_color_format(lv_display_t *disp, *lv_color_format_t* color_format)

Set the color format of the display.

Nota

To change the endianness of the rendered image in case of RGB565 format (i.e. swap the 2 bytes) call *lv_draw_sw_rgb565_swap* in the flush_cb

Parametri

- **disp** -- pointer to a display
- **color_format** -- Possible values are
 - LV_COLOR_FORMAT_RGB565
 - LV_COLOR_FORMAT_RGB888
 - LV_COLOR_FORMAT_XRGB888
 - LV_COLOR_FORMAT_ARGB888

***lv_color_format_t* lv_display_get_color_format(lv_display_t *disp)**

Get the color format of the display

Parametri

disp -- pointer to a display

Ritorna

the color format

void lv_display_set_tile_cnt(lv_display_t *disp, uint32_t tile_cnt)

Set the number of tiles for parallel rendering.

Parametri

- **disp** -- pointer to a display
- **tile_cnt** -- number of tiles (1 =< tile_cnt < 256)

uint32_t lv_display_get_tile_cnt(lv_display_t *disp)

Get the number of tiles used for parallel rendering

Parametri

disp -- pointer to a display

Ritorna

number of tiles

void lv_display_set_antialiasing(lv_display_t *disp, bool en)

Disabling anti-aliasing is not supported since v9. This function will be removed. Enable anti-aliasing for the render engine

Parametri

- **disp** -- pointer to a display
- **en** -- true/false

bool lv_display_get_antialiasing(lv_display_t *disp)

Get if anti-aliasing is enabled for a display or not

Parametri

- **disp** -- pointer to a display (NULL to use the default display)

Ritorna

true/false

void lv_display_flush_ready(lv_display_t *disp)

Call from the display driver when the flushing is finished

Parametri

- **disp** -- pointer to display whose `flush_cb` was called

bool lv_display_flush_is_last(lv_display_t *disp)

Tell if it's the last area of the refreshing process. Can be called from `flush_cb` to execute some special display refreshing if needed when all areas area flushed.

Parametri

- **disp** -- pointer to display

Ritorna

true: it's the last area to flush; false: there are other areas too which will be refreshed soon

bool lv_display_is_double_buffered(lv_display_t *disp)

lv_display_render_mode_t **lv_display_get_render_mode(lv_display_t *disp)**

Get display render mode

Parametri

- **disp** -- pointer to a display

Ritorna

display's render mode (LV_DISPLAY_RENDER_MODE_PARTIAL/DIRECT/FULL)

lv_obj_t *lv_display_get_screen_active(lv_display_t *disp)

Return a pointer to the active screen on a display

Parametri

- **disp** -- pointer to display which active screen should be get. (NULL to use the default screen)

Ritorna

pointer to the active screen object (loaded by '`lv_screen_load()`')

lv_obj_t *lv_display_get_screen_prev(lv_display_t *disp)

Return with a pointer to the previous screen. Only used during screen transitions.

Parametri

- **disp** -- pointer to display which previous screen should be get. (NULL to use the default screen)

Ritorna

pointer to the previous screen object or NULL if not used now

lv_obj_t *lv_display_get_screen_loading(lv_display_t *disp)

Return the screen that is currently being loaded by the display

Parametri

- **disp** -- pointer to a display object (NULL to use the default screen)

Ritorna

pointer to the screen being loaded or NULL if no screen is currently being loaded

lv_obj_t *lv_display_get_layer_top(lv_display_t *disp)

Return the top layer. The top layer is the same on all screens and it is above the normal screen layer.

Parametri

- **disp** -- pointer to display which top layer should be get. (NULL to use the default screen)

Ritorna

pointer to the top layer object

```
lv_obj_t *lv_display_get_layer_sys(lv_display_t *disp)
```

Return the sys. layer. The system layer is the same on all screen and it is above the normal screen and the top layer.

Parametri

- **disp** -- pointer to display which sys. layer should be retrieved. (NULL to use the default screen)

Ritorna

pointer to the sys layer object

```
lv_obj_t *lv_display_get_layer_bottom(lv_display_t *disp)
```

Return the bottom layer. The bottom layer is the same on all screen and it is under the normal screen layer. It's visible only if the screen is transparent.

Parametri

- **disp** -- pointer to display (NULL to use the default screen)

Ritorna

pointer to the bottom layer object

```
lv_obj_t *lv_display_get_screen_by_name(const lv_display_t *disp, const char *screen_name)
```

Get screen by its name on a display. The name should be set by `lv_obj_set_name()` or `lv_obj_set_name_static()`.

Parametri

- **disp** -- pointer to a display or NULL to use default display
- **screen_name** -- name of the screen to get

Ritorna

pointer to the screen, or NULL if not found.

```
void lv_screen_load(struct _lv_obj_t *scr)
```

Load a screen on the default display

Parametri

- **scr** -- pointer to a screen

```
void lv_screen_load_anim(lv_obj_t *scr, lv_screen_load_anim_t anim_type, uint32_t time, uint32_t delay, bool auto_del)
```

Switch screen with animation

Parametri

- **scr** -- pointer to the new screen to load
- **anim_type** -- type of the animation from `lv_screen_load_anim_t`, e.g. `LV_SCREEN_LOAD_ANIM_MOVE_LEFT`
- **time** -- time of the animation
- **delay** -- delay before the transition
- **auto_del** -- true: automatically delete the old screen

```
lv_obj_t *lv_screen_active(void)
```

Get the active screen of the default display

Ritorna

pointer to the active screen

```
lv_obj_t *lv_layer_top(void)
```

Get the top layer of the default display

Ritorna

pointer to the top layer

```
lv_obj_t *lv_layer_sys(void)
```

Get the system layer of the default display

Ritorna

pointer to the sys layer

`lv_obj_t *lv_layer_bottom(void)`

Get the bottom layer of the default display

Ritorna

pointer to the bottom layer

`void lv_display_add_event_cb(lv_display_t *disp, lv_event_cb_t event_cb, lv_event_code_t filter, void *user_data)`

Add an event handler to the display

Parametri

- **disp** -- pointer to a display
- **event_cb** -- an event callback
- **filter** -- event code to react or LV_EVENT_ALL
- **user_data** -- optional user_data

`uint32_t lv_display_get_event_count(lv_display_t *disp)`

Get the number of event attached to a display

Parametri

disp -- pointer to a display

Ritorna

number of events

`lv_event_dsc_t *lv_display_get_event_dsc(lv_display_t *disp, uint32_t index)`

Get an event descriptor for an event

Parametri

- **disp** -- pointer to a display
- **index** -- the index of the event

Ritorna

the event descriptor

`bool lv_display_delete_event(lv_display_t *disp, uint32_t index)`

Remove an event

Parametri

- **disp** -- pointer to a display
- **index** -- the index of the event to remove

Ritorna

true: and event was removed; false: no event was removed

`uint32_t lv_display_remove_event_cb_with_user_data(lv_display_t *disp, lv_event_cb_t event_cb, void *user_data)`

Remove an event_cb with user_data

Parametri

- **disp** -- pointer to a display
- **event_cb** -- the event_cb of the event to remove
- **user_data** -- user_data

Ritorna

the count of the event removed

`lv_result_t lv_display_send_event(lv_display_t *disp, lv_event_code_t code, void *param)`

Send an event to a display

Parametri

- **disp** -- pointer to a display
- **code** -- an event code. LV_EVENT_...
- **param** -- optional param

Ritorna

LV_RESULT_OK: disp wasn't deleted in the event.

`lv_area_t *lv_event_get_invalidate_area(lv_event_t *e)`

Get the area to be invalidated. Can be used in LV_EVENT_INVALIDATE_AREA

Parametri

e -- pointer to an event

Ritorna
the area to invalidated (can be modified as required)

```
void lv_display_set_theme(lv_display_t *disp, lv_theme_t *th)
```

Set the theme of a display. If there are no user created widgets yet the screens' theme will be updated

Parametri

- **disp** -- pointer to a display
- **th** -- pointer to a theme

```
lv_theme_t *lv_display_get_theme(lv_display_t *disp)
```

Get the theme of a display

Parametri

- **disp** -- pointer to a display

Ritorna
the display's theme (can be NULL)

```
uint32_t lv_display_get_inactive_time(const lv_display_t *disp)
```

Get elapsed time since last user activity on a display (e.g. click)

Parametri

- **disp** -- pointer to a display (NULL to get the overall smallest inactivity)

Ritorna
elapsed ticks (milliseconds) since the last activity

```
void lv_display_trigger_activity(lv_display_t *disp)
```

Manually trigger an activity on a display

Parametri

- **disp** -- pointer to a display (NULL to use the default display)

```
void lv_display_enable_invalidation(lv_display_t *disp, bool en)
```

Temporarily enable and disable the invalidation of the display.

Parametri

- **disp** -- pointer to a display (NULL to use the default display)
- **en** -- true: enable invalidation; false: invalidation

```
bool lv_display_is_invalidation_enabled(lv_display_t *disp)
```

Get display invalidation is enabled.

Parametri

- **disp** -- pointer to a display (NULL to use the default display)

Ritorna
return true if invalidation is enabled

```
lv_timer_t *lv_display_get_refr_timer(lv_display_t *disp)
```

Get a pointer to the screen refresher timer to modify its parameters with `lv_timer_...` functions.

Parametri

- **disp** -- pointer to a display

Ritorna
pointer to the display refresher timer. (NULL on error)

```
void lv_display_delete_refr_timer(lv_display_t *disp)
```

Delete screen refresher timer

Parametri

- **disp** -- pointer to a display

```
bool lv_display_register_vsync_event(lv_display_t *disp, lv_event_cb_t event_cb, void *user_data)
```

Register vsync event of a display. LV_EVENT_VSYNC event will be sent periodically. Please don't use it in display event listeners, as it may cause memory leaks and illegal access issues.

Parametri

- **disp** -- pointer to a display
- **event_cb** -- an event callback
- **user_data** -- optional user_data

`bool lv_display_unregister_vsync_event(lv_display_t *disp, lv_event_cb_t event_cb, void *user_data)`

Unregister vsync event of a display. LV_EVENT_VSYNC event won't be sent periodically. Please don't use it in display event listeners, as it may cause memory leaks and illegal access issues.

Parametri

- **disp** -- pointer to a display
- **event_cb** -- an event callback
- **user_data** -- optional user_data

`lv_result_t lv_display_send_vsync_event(lv_display_t *disp, void *param)`

Send an vsync event to a display

Parametri

- **disp** -- pointer to a display
- **param** -- optional param

Ritorna

LV_RESULT_OK: disp wasn't deleted in the event.

`void lv_display_set_user_data(lv_display_t *disp, void *user_data)`

`void lv_display_set_driver_data(lv_display_t *disp, void *driver_data)`

`void *lv_display_get_user_data(lv_display_t *disp)`

`void *lv_display_get_driver_data(lv_display_t *disp)`

`lv_draw_buf_t *lv_display_get_buf_active(lv_display_t *disp)`

`void lv_display_rotate_area(lv_display_t *disp, lv_area_t *area)`

Rotate an area in-place according to the display's rotation

Parametri

- **disp** -- pointer to a display
- **area** -- pointer to an area to rotate

`uint32_t lv_display_get_draw_buf_size(lv_display_t *disp)`

Get the size of the draw buffers

Parametri

disp -- pointer to a display

Ritorna

the size of the draw buffer in bytes for valid display, 0 otherwise

`uint32_t lv_display_get_invalidate_buf_size(lv_display_t *disp, uint32_t width, uint32_t height)`

Get the size of the invalidated draw buffer. Can be used in the flush callback to get the number of bytes used in the current render buffer.

Parametri

- **disp** -- pointer to a display
- **width** -- the width of the invalidated area
- **height** -- the height of the invalidated area

Ritorna

the size of the invalidated draw buffer in bytes, not accounting for any preceding palette information for a valid display, 0 otherwise

`int32_t lv_dpx(int32_t n)`

For default display, computes the number of pixels (a distance or size) as if the display had 160 DPI. This allows you to specify 1/160-th fractions of an inch to get real distance on the display that will be consistent regardless of its current DPI. It ensures `lv_dpx(100)`, for example, will have the same physical size regardless to the DPI of the display.

Parametri

n -- number of 1/160-th-inch units to compute with

Ritorna

number of pixels to use to make that distance

```
int32_t lv_display_dpx(const lv_display_t *disp, int32_t n)
```

For specified display, computes the number of pixels (a distance or size) as if the display had 160 DPI. This allows you to specify 1/160-th fractions of an inch to get real distance on the display that will be consistent regardless of its current DPI. It ensures `lv_dpx(100)`, for example, will have the same physical size regardless to the DPI of the display.

Parametri

- **disp** -- pointer to display whose dpi should be considered
- **n** -- number of 1/160-th-inch units to compute with

Ritorna

number of pixels to use to make that distance

14.14.2 lv_display_private.h

Define

LV_INV_BUF_SIZE

Buffer size for invalid areas

struct _lv_display_t

Membri Pubblici

int32_t hor_res

Horizontal resolution.

int32_t ver_res

Vertical resolution.

int32_t physical_hor_res

Horizontal resolution of the full / physical display. Set to -1 for fullscreen mode.

int32_t physical_ver_res

Vertical resolution of the full / physical display. Set to -1 for fullscreen mode.

int32_t offset_x

Horizontal offset from the full / physical display. Set to 0 for fullscreen mode.

int32_t offset_y

Vertical offset from the full / physical display. Set to 0 for fullscreen mode.

uint32_t dpi

DPI (dot per inch) of the display. Default value is `LV_DPI_DEF`.

lv_draw_buf_t *buf_1

lv_draw_buf_t *buf_2

lv_draw_buf_t *buf_3

lv_draw_buf_t *buf_act

Internal, used by the library

lv_display_flush_cb_t flush_cb

MANDATORY: Write the internal buffer (draw_buf) to the display. '*lv_display_flush_ready()*' has to be called when finished

lv_display_flush_wait_cb_t flush_wait_cb

Used to wait while flushing is ready. It can do any complex logic to wait, including semaphores, mutexes, polling flags, etc. If not set flushing flag is used which can be cleared with *lv_display_flush_ready()*

volatile int **flushing**

1: flushing is in progress. (It can't be a bit field because when it's cleared from IRQ Read-Modify-Write issue might occur)

volatile int **flushing_last**

1: It was the last chunk to flush. (It can't be a bit field because when it's cleared from IRQ Read-Modify-Write issue might occur)

volatile uint32_t **last_area**

1: last area is being rendered

volatile uint32_t **last_part**

1: last part of the current area is being rendered

lv_display_render_mode_t render_mode**uint32_t **antialiasing****

1: anti-aliasing is enabled on this display.

uint32_t **tile_cnt**

Divide the display buffer into these number of tiles

uint32_t **stride_is_auto**

1: The stride of the buffers was not set explicitly.

uint32_t **rendering_in_progress**

1: The current screen rendering is in progress

lv_color_format_t color_format***lv_area_t inv_areas[32]***

Invalidate (marked to redraw) areas

uint8_t **inv_area_joined[32]****uint32_t **inv_p******int32_t **inv_en_cnt*******lv_ll_t sync_areas***

Double buffer sync areas (redrawn during last refresh)

lv_draw_buf_t _static_buf1

Used when user pass in a raw buffer as display draw buffer

lv_draw_buf_t _static_buf2**lv_layer_t *layer_head**

`void (*layer_init)(lv_display_t *disp, lv_layer_t *layer)`

`void (*layer_deinit)(lv_display_t *disp, lv_layer_t *layer)`

lv_obj_t **screens

Screens of the display Array of screen objects.

lv_obj_t *sys_layer
 **Vedi anche**

[*lv_display_get_layer_sys*](#)

lv_obj_t *top_layer
 **Vedi anche**

[*lv_display_get_layer_top*](#)

lv_obj_t *act_scr

Currently active screen on this display

lv_obj_t *bottom_layer
 **Vedi anche**

[*lv_display_get_layer_bottom*](#)

lv_obj_t *prev_scr

Previous screen. Used during screen animations

lv_obj_t *scr_to_load

The screen prepared to load in `lv_screen_load_anim`

uint32_t screen_cnt**uint8_t draw_prev_over_act**

```

uint8_t del_prev
    1: Draw previous screen over active screen

void *driver_data
    1: Automatically delete the previous screen when the screen load animation is ready Custom user data

void *user_data
    Custom user data

lv_event_list_t event_list

uint32_t rotation
    Element of lv_display_rotation_t

uint32_t matrix_rotation
    1: Use matrix for display rotation

lv_theme_t *theme
    The theme assigned to the screen

lv_timer_t *refr_timer
    A timer which periodically checks the dirty areas and refreshes them

uint32_t last_activity_time
    Last time when there was activity on this display

lv_area_t refreshed_area
    The area being refreshed

uint32_t vsync_count

lv_obj_t *perf_label

lv_sysmon_backend_data_t perf_sysmon_backend

lv_sysmon_perf_info_t perf_sysmon_info

lv_obj_t *mem_label

```

14.15 draw

14.15.1 lv_draw.h

Define

LV_DRAW_UNIT_NONE

Modified by NXP in 2024

LV_DRAW_UNIT_IDLE

The draw unit is idle, new dispatching might be requested to try again

Enum

enum **lv_draw_task_type_t**

Values:

enumerator **LV_DRAW_TASK_TYPE_NONE**

enumerator **LV_DRAW_TASK_TYPE_FILL**

enumerator **LV_DRAW_TASK_TYPE_BORDER**

enumerator **LV_DRAW_TASK_TYPE_BOX_SHADOW**

enumerator **LV_DRAW_TASK_TYPE LETTER**

enumerator **LV_DRAW_TASK_TYPE_LABEL**

enumerator **LV_DRAW_TASK_TYPE_IMAGE**

enumerator **LV_DRAW_TASK_TYPE_LAYER**

enumerator **LV_DRAW_TASK_TYPE_LINE**

enumerator **LV_DRAW_TASK_TYPE_ARC**

enumerator **LV_DRAW_TASK_TYPE_TRIANGLE**

enumerator **LV_DRAW_TASK_TYPE_MASK_RECTANGLE**

enumerator **LV_DRAW_TASK_TYPE_MASK_BITMAP**

enumerator **LV_DRAW_TASK_TYPE_VECTOR**

enumerator **LV_DRAW_TASK_TYPE_3D**

enum **lv_draw_task_state_t**

Values:

enumerator **LV_DRAW_TASK_STATE_BLOCKED**

Waiting for an other task to be finished. For example in case of **LV_DRAW_TASK_TYPE_LAYER** (used to blend a layer) is blocked until all the draw tasks of the layer is rendered.

enumerator **LV_DRAW_TASK_STATE_WAITING**

The draw task is added to the layers list and waits to be rendered.

enumerator LV_DRAW_TASK_STATE_QUEUED

The draw task is added to the command queue of the draw unit. As the queued task are executed in order it's possible to queue multiple draw task (for the same draw unit) even if they are depending on each other. Therefore `lv_draw_get_available_task` and `lv_draw_get_next_available_task` can return draw task for the same draw unit even if a dependent draw task is not finished ready yet.

enumerator LV_DRAW_TASK_STATE_IN_PROGRESS

The draw task is being rendered. This draw task needs to be finished before `lv_draw_get_available_task` and `lv_draw_get_next_available_task` would return any depending draw tasks.

enumerator LV_DRAW_TASK_STATE_FINISHED

The draw task is rendered. It will be removed from the draw task list of the layer and freed automatically.

Funzioni

void lv_draw_init(void)

Used internally to initialize the drawing module

void lv_draw_deinit(void)

Deinitialize the drawing module

void *lv_draw_create_unit(size_t size)

Allocate a new draw unit with the given size and appends it to the list of draw units

Parametri

- **size** -- the size to allocate. E.g. `sizeof(my_draw_unit_t)`, where the first element of `my_draw_unit_t` is `lv_draw_unit_t`.

lv_draw_task_t *lv_draw_add_task(lv_layer_t *layer, const lv_area_t *coords, lv_draw_task_type_t type)

Add an empty draw task to the draw task list of a layer.

Parametri

- **layer** -- pointer to a layer
- **coords** -- the coordinates of the draw task

Ritorna

the created draw task which needs to be further configured e.g. by added a draw descriptor

void lv_draw_finalize_task_creation(lv_layer_t *layer, lv_draw_task_t *t)

Needs to be called when a draw task is created and configured. It will send an event about the new draw task to the widget and assign it to a draw unit.

Parametri

- **layer** -- pointer to a layer
- **t** -- pointer to a draw task

void lv_draw_dispatch(void)

Try dispatching draw tasks to draw units

bool lv_draw_dispatch_layer(lv_display_t *disp, lv_layer_t *layer)

Used internally to try dispatching draw tasks of a specific layer

Parametri

- **disp** -- pointer to a display on which the dispatching was requested
- **layer** -- pointer to a layer

Ritorna

at least one draw task is being rendered (maybe it was taken earlier)

void lv_draw_dispatch_wait_for_request(void)

Wait for a new dispatch request. It's blocking if `LV_USE_OS == 0` else it yields

void lv_draw_wait_for_finish(void)

Wait for draw finish in case of asynchronous task execution. If `LV_USE_OS == 0` it just return.

```
void lv_draw_dispatch_request(void)
```

When a draw unit finished a draw task it needs to request dispatching to let LVGL assign a new draw task to it

```
uint32_t lv_draw_get_unit_count(void)
```

Get the total number of draw units.

```
lv_draw_task_t *lv_draw_get_available_task(lv_layer_t *layer, lv_draw_task_t *t_prev, uint8_t draw_unit_id)
```

If there is only one draw unit check the first draw task if it's available. If there are multiple draw units call `lv_draw_get_next_available_task` to find a task.

Parametri

- **layer** -- the draw layer to search in
- **t_prev** -- continue searching from this task
- **draw_unit_id** -- check the task where `preferred_draw_unit_id` equals this value or `LV_DRAW_UNIT_NONE`

Ritorna

an available draw task or NULL if there is not any

```
lv_draw_task_t *lv_draw_get_next_available_task(lv_layer_t *layer, lv_draw_task_t *t_prev, uint8_t draw_unit_id)
```

Find and available draw task

Parametri

- **layer** -- the draw layer to search in
- **t_prev** -- continue searching from this task
- **draw_unit_id** -- check the task where `preferred_draw_unit_id` equals this value or `LV_DRAW_UNIT_NONE`

Ritorna

an available draw task or NULL if there is not any

```
uint32_t lv_draw_get_dependent_count(lv_draw_task_t *t_check)
```

Tell how many draw task are waiting to be drawn on the area of `t_check`. It can be used to determine if a GPU shall combine many draw tasks into one or not. If a lot of tasks are waiting for the current ones it makes sense to draw them one-by-one to not block the dependent tasks' rendering

Parametri

`t_check` -- the task whose dependent tasks shall be counted

Ritorna

number of tasks depending on `t_check`

```
void lv_draw_unit_send_event(const char *name, lv_event_code_t code, void *param)
```

Send an event to the draw units

Parametri

- **name** -- the name of the draw unit to send the event to
- **code** -- the event code
- **param** -- the event parameter

```
void lv_layer_init(lv_layer_t *layer)
```

Initialize a layer

Parametri

`layer` -- pointer to a layer to initialize

```
void lv_layer_reset(lv_layer_t *layer)
```

Reset the layer to a drawable state

Parametri

`layer` -- pointer to a layer to reset

```
lv_layer_t *lv_draw_layer_create(lv_layer_t *parent_layer, lv_color_format_t color_format, const lv_area_t *area)
```

Create (allocate) a new layer on a parent layer

Parametri

- **parent_layer** -- the parent layer to which the layer will be merged when it's rendered
- **color_format** -- the color format of the layer

- **area** -- the areas of the layer (absolute coordinates)

Ritorna

the new target_layer or NULL on error

```
void lv_draw_layer_init(lv_layer_t *layer, lv_layer_t *parent_layer, lv_color_format_t color_format, const lv_area_t *area)
```

Initialize a layer which is allocated by the user

Parametri

- **layer** -- pointer the layer to initialize (its lifetime needs to be managed by the user)
- **parent_layer** -- the parent layer to which the layer will be merged when it's rendered
- **color_format** -- the color format of the layer
- **area** -- the areas of the layer (absolute coordinates)

Ritorna

the new target_layer or NULL on error

```
void *lv_draw_layer_alloc_buf(lv_layer_t *layer)
```

Try to allocate a buffer for the layer.

Parametri

layer -- pointer to a layer

Ritorna

pointer to the allocated aligned buffer or NULL on failure

```
void *lv_draw_layer_go_to_xy(lv_layer_t *layer, int32_t x, int32_t y)
```

Got to a pixel at X and Y coordinate on a layer

Parametri

- **layer** -- pointer to a layer
- **x** -- the target X coordinate
- **y** -- the target X coordinate

Ritorna

buf offset to point to the given X and Y coordinate

```
lv_draw_task_type_t lv_draw_task_get_type(const lv_draw_task_t *t)
```

Get the type of a draw task

Parametri

t -- the draw task to get the type of

Ritorna

the draw task type

```
void *lv_draw_task_get_draw_desc(const lv_draw_task_t *t)
```

Get the draw descriptor of a draw task

Parametri

t -- the draw task to get the draw descriptor of

Ritorna

a void pointer to the draw descriptor

```
void lv_draw_task_get_area(const lv_draw_task_t *t, lv_area_t *area)
```

Get the draw area of a draw task

Parametri

- **t** -- the draw task to get the draw area of
- **area** -- the destination where the draw area will be stored

struct _lv_layer_t

Membri Pubblici

lv_draw_buf_t *draw_buf

Target draw buffer of the layer

lv_draw_task_t *draw_task_head

Linked list of draw tasks

lv_layer_t *parent

Parent layer

lv_layer_t *next

Next layer

void *user_data

User data

lv_area_t buf_area

The absolute coordinates of the buffer

lv_area_t phy_clip_area

The physical clipping area relative to the display

lv_area_t _clip_area

NEVER USE IT DRAW UNITS. USED INTERNALLY DURING DRAW TASK CREATION. The current clip area with absolute coordinates, always the same or smaller than buf_area Can be set before new draw tasks are added to indicate the clip area of the draw tasks. Therefore `lv_draw_add_task()` always saves it in the new draw task to know the clip area when the draw task was added. During drawing the draw units also sees the saved clip_area and should use it during drawing. During drawing the layer's clip area shouldn't be used as it might be already changed for other draw tasks.

int32_t partial_y_offset

Partial y offset

lv_color32_t recolor

Recolor of the layer

lv_color_format_t color_format

The color format of the layer. LV_COLOR_FORMAT_...

bool all_tasks_added

Flag indicating all tasks are added

lv_opa_t opa

Opacity of the layer

struct **lv_draw_dsc_base_t**

Membri Pubblici

lv_obj_t *obj

The widget for which draw descriptor was created

uint32_t part

The widget part for which draw descriptor was created

uint32_t id1

A widget type specific ID (e.g. table row index). See the docs of the given widget.

uint32_t id2

A widget type specific ID (e.g. table column index). See the docs of the given widget.

lv_layer_t *layer

The target layer

size_t dsc_size

Size of the specific draw descriptor into which this base descriptor is embedded

void *user_data

Any custom user data

14.15.2 lv_draw_3d.h

Funzioni

void lv_draw_3d_dsc_init(lv_draw_3d_dsc_t *dsc)

Initialize a 3D draw descriptor

Parametri

dsc -- pointer to a draw descriptor

lv_draw_3d_dsc_t *lv_draw_task_get_3d_dsc(lv_draw_task_t *task)

Try to get a 3D draw descriptor from a draw task.

Parametri

task -- draw task

Ritorna

the task's draw descriptor or NULL if the task is not of type LV_DRAW_TASK_TYPE_3D

void lv_draw_3d(lv_layer_t *layer, const lv_draw_3d_dsc_t *dsc, const lv_area_t *coords)

Create a 3D draw task

Parametri

- **layer** -- pointer to a layer
- **dsc** -- pointer to an initialized *lv_draw_3d_dsc_t* variable

struct lv_draw_3d_dsc_t

Membri Pubblici

lv_draw_dsc_base_t base

lv_3dtexture_id_t tex_id

bool h_flip

bool v_flip

lv_opa_t opa

14.15.3 lv_draw_arc.h

Funzioni

`void lv_draw_arc_dsc_init(lv_draw_arc_dsc_t *dsc)`

Initialize an arc draw descriptor.

Parametri

`dsc` -- pointer to a draw descriptor

`lv_draw_arc_dsc_t *lv_draw_task_get_arc_dsc(lv_draw_task_t *task)`

Try to get an arc draw descriptor from a draw task.

Parametri

`task` -- draw task

Ritorna

the task's draw descriptor or NULL if the task is not of type LV_DRAW_TASK_TYPE_ARC

`void lv_draw_arc(lv_layer_t *layer, const lv_draw_arc_dsc_t *dsc)`

Create an arc draw task.

Parametri

- `layer` -- pointer to a layer
- `dsc` -- pointer to an initialized draw descriptor variable

`void lv_draw_arc_get_area(int32_t x, int32_t y, uint16_t radius, lv_value_precise_t start_angle, lv_value_precise_t end_angle, int32_t w, bool rounded, lv_area_t *area)`

Get an area the should be invalidated when the arcs angle changed between start_angle and end_angle

Parametri

- `x` -- the x coordinate of the center of the arc
- `y` -- the y coordinate of the center of the arc
- `radius` -- the radius of the arc
- `start_angle` -- the start angle of the arc (0 deg on the bottom, 90 deg on the right)
- `end_angle` -- the end angle of the arc
- `w` -- width of the arc
- `rounded` -- true: the arc is rounded
- `area` -- store the area to invalidate here

`struct lv_draw_arc_dsc_t`

Membri Pubblici

`lv_draw_dsc_base_t base`

`lv_color_t color`

The color of the arc

`int32_t width`

The width (thickness) of the arc

`lv_value_precise_t start_angle`

The start angle in 1 degree units (if LV_USE_FLOAT is enabled a float number can be also used) 0° is the 3 o'clock position, 90° is the 6 o'clock, etc.

`lv_value_precise_t end_angle`

The end angle, similarly to start_angle.

***lv_point_t* center**

The center point of the arc.

const void *img_src

An image source to be used instead of color. NULL if unused

uint16_t radius

The outer radius of the arc

lv_opa_t opa

Opacity of the arc in 0...255 range. LV_OPA_TRANSP, LV_OPA_10, LV_OPA_20, ..
LV_OPA_COVER can be used as well

uint8_t rounded

1: Make the arc ends rounded

14.15.4 lv_draw_buf.h

Define

LV_STRIDE_AUTO

Use this value to let LVGL calculate stride automatically

LV_DRAW_BUF_STRIDE(w, cf)

Stride alignment for draw buffers. It may vary between different color formats and hardware. Refine it to suit your needs.

LV_DRAW_BUF_SIZE(w, h, cf)

Allocate a slightly larger buffer, so we can adjust the start address to meet alignment

LV_DRAW_BUF_DEFINE_STATIC(name, _w, _h, _cf)

Define a static draw buffer with the given width, height, and color format. Stride alignment is set to LV_DRAW_BUF_STRIDE_ALIGN.

For platform that needs special buffer alignment, call LV_DRAW_BUF_INIT_STATIC.

LV_DRAW_BUF_INIT_STATIC(name)

Typedef

```
typedef void *(*lv_draw_buf_malloc_cb_t)(size_t size, lv_color_format_t color_format)
```

```
typedef void (*lv_draw_buf_free_cb_t)(void *draw_buf)
```

```
typedef void (*lv_draw_buf_copy_cb_t)(lv_draw_buf_t *dest, const lv_area_t *dest_area, const lv_draw_buf_t *src, const lv_area_t *src_area)
```

```
typedef void (*lv_draw_buf_align_cb_t)(void *buf, lv_color_format_t color_format)
```

```
typedef void (*lv_draw_buf_cache_operation_cb_t)(const lv_draw_buf_t *draw_buf, const lv_area_t *area)
```

```
typedef uint32_t (*lv_draw_buf_width_to_stride_cb_t)(uint32_t w, lv_color_format_t color_format)
```

Funzioni

void lv_draw_buf_init_with_default_handlers(lv_draw_buf_handlers_t *handlers)

Initialize the draw buffer with the default handlers.

Parametri

handlers -- the draw buffer handlers to set

void lv_draw_buf_handlers_init(lv_draw_buf_handlers_t *handlers, lv_draw_buf_malloc_cb_t buf_malloc_cb, lv_draw_buf_free_cb_t buf_free_cb, lv_draw_buf_copy_cb_t buf_copy_cb, lv_draw_buf_align_cb_t align_pointer_cb, lv_draw_buf_cache_operation_cb_t invalidate_cache_cb, lv_draw_buf_cache_operation_cb_t flush_cache_cb, lv_draw_buf_width_to_stride_cb_t width_to_stride_cb)

Initialize the draw buffer with given handlers.

Parametri

- **handlers** -- the draw buffer handlers to set
- **buf_malloc_cb** -- the callback to allocate memory for the buffer
- **buf_free_cb** -- the callback to free memory of the buffer
- **buf_copy_cb** -- the callback to copy a draw buffer to an other
- **align_pointer_cb** -- the callback to align the buffer
- **invalidate_cache_cb** -- the callback to invalidate the cache of the buffer
- **flush_cache_cb** -- the callback to flush buffer
- **width_to_stride_cb** -- the callback to calculate the stride based on the width and color format

lv_draw_buf_handlers_t *lv_draw_buf_get_handlers(void)

Get the struct which holds the callbacks for draw buf management. Custom callback can be set on the returned value

Ritorna

pointer to the struct of handlers

lv_draw_buf_handlers_t *lv_draw_buf_get_font_handlers(void)

lv_draw_buf_handlers_t *lv_draw_buf_get_image_handlers(void)

void *lv_draw_buf_align(void *buf, lv_color_format_t color_format)

Align the address of a buffer. The buffer needs to be large enough for the real data after alignment

Parametri

- **buf** -- the data to align
- **color_format** -- the color format of the buffer

Ritorna

the aligned buffer

void *lv_draw_buf_align_ex(const lv_draw_buf_handlers_t *handlers, void *buf, lv_color_format_t color_format)

Align the address of a buffer with custom draw buffer handlers. The buffer needs to be large enough for the real data after alignment

Parametri

- **handlers** -- the draw buffer handlers
- **buf** -- the data to align
- **color_format** -- the color format of the buffer

Ritorna

the aligned buffer

void lv_draw_buf_invalidate_cache(const lv_draw_buf_t *draw_buf, const lv_area_t *area)

Invalidate the cache of the buffer

Parametri

- **draw_buf** -- the draw buffer needs to be invalidated
- **area** -- the area to invalidate in the buffer, use NULL to invalidate the whole draw buffer address range

```
void lv_draw_buf_flush_cache(const lv_draw_buf_t *draw_buf, const lv_area_t *area)
```

Flush the cache of the buffer

Parametri

- **draw_buf** -- the draw buffer needs to be flushed
- **area** -- the area to flush in the buffer, use NULL to flush the whole draw buffer address range

```
uint32_t lv_draw_buf_width_to_stride(uint32_t w, lv_color_format_t color_format)
```

Calculate the stride in bytes based on a width and color format

Parametri

- **w** -- the width in pixels
- **color_format** -- the color format

Ritorna

the stride in bytes

```
uint32_t lv_draw_buf_width_to_stride_ex(const lv_draw_buf_handlers_t *handlers, uint32_t w,
                                         lv_color_format_t color_format)
```

Calculate the stride in bytes based on a width and color format

Parametri

- **handlers** -- the draw buffer handlers
- **w** -- the width in pixels
- **color_format** -- the color format

Ritorna

the stride in bytes

```
void lv_draw_buf_clear(lv_draw_buf_t *draw_buf, const lv_area_t *a)
```

Clear an area on the buffer

Parametri

- **draw_buf** -- pointer to draw buffer
- **a** -- the area to clear, or NULL to clear the whole buffer

```
lv_draw_buf_t *lv_draw_buf_create(uint32_t w, uint32_t h, lv_color_format_t cf, uint32_t stride)
```

Note: Eventually, lv_draw_buf_malloc/free will be kept as private. For now, we use `create` to distinguish with malloc.

Create an draw buf by allocating struct for `lv_draw_buf_t` and allocating a buffer for it that meets specified requirements.

Parametri

- **w** -- the buffer width in pixels
- **h** -- the buffer height in pixels
- **cf** -- the color format for image
- **stride** -- the stride in bytes for image. Use 0 for automatic calculation based on w, cf, and global stride alignment configuration.

```
lv_draw_buf_t *lv_draw_buf_create_ex(const lv_draw_buf_handlers_t *handlers, uint32_t w, uint32_t h,
                                    lv_color_format_t cf, uint32_t stride)
```

Note: Eventually, lv_draw_buf_malloc/free will be kept as private. For now, we use `create` to distinguish with malloc.

Create an draw buf by allocating struct for `lv_draw_buf_t` and allocating a buffer for it that meets specified requirements.

Parametri

- **handlers** -- the draw buffer handlers
- **w** -- the buffer width in pixels
- **h** -- the buffer height in pixels
- **cf** -- the color format for image
- **stride** -- the stride in bytes for image. Use 0 for automatic calculation based on w, cf, and global stride alignment configuration.

```
lv_draw_buf_t *lv_draw_buf_dup(const lv_draw_buf_t *draw_buf)
```

Duplicate a draw buf with same image size, stride and color format. Copy the image data too.

Parametri

- draw_buf** -- the draw buf to duplicate

Ritorna

the duplicated draw buf on success, NULL if failed

```
lv_draw_buf_t *lv_draw_buf_dup_ex(const lv_draw_buf_handlers_t *handlers, const lv_draw_buf_t *draw_buf)
```

Duplicate a draw buf with same image size, stride and color format. Copy the image data too.

Parametri

- **handlers** -- the draw buffer handlers
- **draw_buf** -- the draw buf to duplicate

Ritorna

the duplicated draw buf on success, NULL if failed

```
lv_result_t lv_draw_buf_init(lv_draw_buf_t *draw_buf, uint32_t w, uint32_t h, lv_color_format_t cf, uint32_t stride, void *data, uint32_t data_size)
```

Initialize a draw buf with the given buffer and parameters. Clear draw buffer flag to zero.

Parametri

- **draw_buf** -- the draw buf to initialize
- **w** -- the buffer width in pixels
- **h** -- the buffer height in pixels
- **cf** -- the color format
- **stride** -- the stride in bytes. Use 0 for automatic calculation
- **data** -- the buffer used for drawing. Unaligned data will be aligned internally
- **data_size** -- the size of the buffer in bytes

Ritorna

return LV_RESULT_OK on success, LV_RESULT_INVALID otherwise

```
lv_draw_buf_t *lv_draw_buf_reshape(lv_draw_buf_t *draw_buf, lv_color_format_t cf, uint32_t w, uint32_t h, uint32_t stride)
```

Keep using the existing memory, reshape the draw buffer to the given width and height. Return NULL if data_size is smaller than the required size.

Parametri

- **draw_buf** -- pointer to a draw buffer
- **cf** -- the new color format, use 0 or LV_COLOR_FORMAT_UNKNOWN to keep using the original color format.
- **w** -- the new width in pixels
- **h** -- the new height in pixels
- **stride** -- the stride in bytes for image. Use 0 for automatic calculation.

```
void lv_draw_buf_destroy(lv_draw_buf_t *draw_buf)
```

Destroy a draw buf by freeing the actual buffer if it's marked as LV_IMAGE_FLAGS_ALLOCATED in header. Then free the lv_draw_buf_t struct.

Parametri

- draw_buf** -- the draw buffer to destroy

```
void lv_draw_buf_copy(lv_draw_buf_t *dest, const lv_area_t *dest_area, const lv_draw_buf_t *src, const lv_area_t *src_area)
```

Copy an area from a buffer to another

Nota

`dest_area` and `src_area` should have the same width and height

Nota

The default copy function required `dest` and `src` to have the same color format. Overwriting `dest->handlers->buf_copy_cb` can resolve this limitation.

Parametri

- **dest** -- pointer to the destination draw buffer
- **dest_area** -- the area to copy from the destination buffer, if NULL, use the whole buffer
- **src** -- pointer to the source draw buffer
- **src_area** -- the area to copy from the source buffer, if NULL, use the whole buffer

`void *lv_draw_buf_goto_xy(const lv_draw_buf_t *buf, uint32_t x, uint32_t y)`

Return pointer to the buffer at the given coordinates

`lv_result_t lv_draw_buf_adjust_stride(lv_draw_buf_t *src, uint32_t stride)`

Adjust the stride of a draw buf in place.

Parametri

- **src** -- pointer to a draw buffer
- **stride** -- the new stride in bytes for image. Use LV_STRIDE_AUTO for automatic calculation.

Ritorna

LV_RESULT_OK: success or LV_RESULT_INVALID: failed

`lv_result_t lv_draw_buf_premultiply(lv_draw_buf_t *draw_buf)`

Premultiply draw buffer color with alpha channel. If it's already premultiplied, return directly. Only color formats with alpha channel will be processed.

Ritorna

LV_RESULT_OK: premultiply success

`bool lv_draw_buf_has_flag(const lv_draw_buf_t *draw_buf, lv_image_flags_t flag)`

`void lv_draw_buf_set_flag(lv_draw_buf_t *draw_buf, lv_image_flags_t flag)`

`void lv_draw_buf_clear_flag(lv_draw_buf_t *draw_buf, lv_image_flags_t flag)`

`lv_result_t lv_draw_buf_from_image(lv_draw_buf_t *buf, const lv_image_dsc_t *img)`

As of now, draw buf share same definition as `lv_image_dsc_t`. And is interchangeable with `lv_image_dsc_t`.

`void lv_draw_buf_to_image(const lv_draw_buf_t *buf, lv_image_dsc_t *img)`

`void lv_draw_buf_set_palette(lv_draw_buf_t *draw_buf, uint8_t index, lv_color32_t color)`

Set the palette color of an indexed image. Valid only for LV_COLOR_FORMAT_I1/2/4/8

Parametri

- **draw_buf** -- pointer to an image descriptor
- **index** -- the palette color to set:
 - for LV_COLOR_FORMAT_I1: 0..1
 - for LV_COLOR_FORMAT_I2: 0..3
 - for LV_COLOR_FORMAT_I4: 0..15
 - for LV_COLOR_FORMAT_I8: 0..255
- **color** -- the color to set in `lv_color32_t` format

`void lv_image_buf_set_palette(lv_image_dsc_t *dsc, uint8_t id, lv_color32_t c)`

Deprecated:

Use `lv_draw_buf_set_palette` instead.

`void lv_image_buf_free(lv_image_dsc_t *dsc)`

Deprecated:

Use `lv_draw_buffer_create/destroy` instead. Free the data pointer and dsc struct of an image.

`struct _lv_draw_buf_t`

Membri Pubblici

lv_image_header_t **header**

uint32_t **data_size**
Total buf size in bytes

uint8_t ***data**

void ***unaligned_data**
Unaligned address of data, used internally by lvgl

const lv_draw_buf_handlers_t ***handlers**
draw buffer alloc/free ops.

14.15.5 lv_draw_buf_private.h**Funzioni**

void **lv_draw_buf_init_handlers**(void)
Called internally to initialize the draw_buf_handlers in lv_global

struct **lv_draw_buf_handlers_t**

Membri Pubblici

lv_draw_buf_malloc_cb_t **buf_malloc_cb**

lv_draw_buf_free_cb_t **buf_free_cb**

lv_draw_buf_copy_cb_t **buf_copy_cb**

lv_draw_buf_align_cb_t **align_pointer_cb**

lv_draw_buf_cache_operation_cb_t **invalidate_cache_cb**

lv_draw_buf_cache_operation_cb_t **flush_cache_cb**

lv_draw_buf_width_to_stride_cb_t **width_to_stride_cb**

14.15.6 lv_draw_image.h**Typedef**

typedef void (***lv_draw_image_core_cb**)(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_image_decoder_dsc_t *decoder_dsc, lv_draw_image_sup_t *sup, const *lv_area_t* *img_coords, const *lv_area_t* *clipped_img_area)

PErform the actual rendering of a decoded image

Param t

pointer to a draw task

Param draw_dsc

the draw descriptor of the image

Param decoder_dsc

pointer to the decoded image's descriptor

Param sup

supplementary data

Param img_coords

the absolute coordinates of the image

Param clipped_img_area

the absolute clip coordinates

Funzioni

void lv_draw_image_dsc_init(lv_draw_image_dsc_t *dsc)

Initialize an image draw descriptor.

Parametri

dsc -- pointer to a draw descriptor

lv_draw_image_dsc_t *lv_draw_task_get_image_dsc(lv_draw_task_t *task)

Try to get an image draw descriptor from a draw task.

Parametri

task -- draw task

Ritorna

the task's draw descriptor or NULL if the task is not of type LV_DRAW_TASK_TYPE_IMAGE

void lv_draw_image(lv_layer_t *layer, const lv_draw_image_dsc_t *dsc, const lv_area_t *coords)

Create an image draw task

i Nota

coords can be small than the real image area (if only a part of the image is rendered) or can be larger (in case of tiled images). .

Parametri

- **layer** -- pointer to a layer
- **dsc** -- pointer to an initialized draw descriptor
- **coords** -- the coordinates of the image

void lv_draw_layer(lv_layer_t *layer, const lv_draw_image_dsc_t *dsc, const lv_area_t *coords)

Create a draw task to blend a layer to another layer

i Nota

coords can be small than the total widget area from which the layer is created (if only a part of the widget was rendered to a layer)

Parametri

- **layer** -- pointer to a layer
- **dsc** -- pointer to an initialized draw descriptor. **src** must be set to the layer to blend
- **coords** -- the coordinates of the layer.

lv_image_src_t lv_image_src_get_type(const void *src)

Get the type of an image source

Parametri

src -- pointer to an image source:

- pointer to an 'lv_image_t' variable (image stored internally and compiled into the code)

- a path to a file (e.g. "S:/folder/image.bin")
- or a symbol (e.g. LV_SYMBOL_CLOSE)

Ritorna

type of the image source LV_IMAGE_SRC_VARIABLE/FILE/SYMBOL/UNKNOWN

```
struct _lv_draw_image_dsc_t
```

Membri Pubblici

lv_draw_dsc_base_t **base**

const void *src

The image source: pointer to *lv_image_dsc_t* or a path to a file

lv_image_header_t **header**

The header of the image. Initialized internally in *lv_draw_image*

int32_t clip_radius

Clip the corner of the image with this radius. Use LV_RADIUS_CIRCLE for max. radius

int32_t rotation

The rotation of the image in 0.1 degree unit. E.g. 234 means 23.4°

int32_t scale_x

Horizontal scale (zoom) of the image. 256 (LV_SCALE_NONE): means no zoom, 512 double size, 128 half size.

int32_t scale_y

Same as *scale_y* but vertically

int32_t skew_x

Parallelogram like transformation of the image horizontally in 0.1 degree unit. E.g. 456 means 45.6°.

int32_t skew_y

Same as *skew_x* but vertically

lv_point_t **pivot**

The pivot point of transformation (scale and rotation). 0;0 is the top left corner of the image. Can be outside of the image too.

lv_color_t **recolor**

Mix this color to the images. In case of LV_COLOR_FORMAT_A8 it will be the color of the visible pixels

lv_opa_t **recolor_opa**

The intensity of recoloring. 0 means, no recolor, 255 means full cover (transparent pixels remain transparent)

lv_opa_t **opa**

Opacity in 0...255 range. LV_OPA_TRANSP, LV_OPA_10, LV_OPA_20, .. LV_OPA_COVER can be used as well

lv_blend_mode_t blend_mode

Describes how to blend the pixels of the image to the background. See [*lv_blend_mode_t*](#) for more details.

uint16_t antialias

1: perform the transformation with anti-aliasing

uint16_t tile

If the image is smaller than the `image_area` field of `lv_draw_image_dsc_t` tile the image (repeat is both horizontally and vertically) to fill the `image_area` area

const lv_image_colorkey_t *colorkey***lv_draw_image_sup_t *sup***

Used internally to store some information about the palette or the color of A8 images

lv_area_t image_area

Used to indicate the entire original, non-clipped area where the image is to be drawn. This is important for:

- i. Layer rendering, where it might happen that only a smaller area of the layer is rendered and e.g. `clip_radius` needs to know what the original image was.
- ii. Tiled images, where the target draw area is larger than the image to be tiled.

const lv_image_dsc_t *bitmap_mask_src

Pointer to an A8 or L8 image descriptor to mask the image with. The mask is always center aligned.

14.15.7 `lv_draw_image_private.h`

Funzioni

```
void lv_draw_image_normal_helper(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t
                                *coords, lv_draw_image_core_cb draw_core_cb)
```

Can be used by draw units to handle the decoding and prepare everything for the actual image rendering

Parametri

- `t` -- pointer to a draw task
- `draw_dsc` -- the draw descriptor of the image
- `coords` -- the absolute coordinates of the image
- `draw_core_cb` -- a callback to perform the actual rendering

```
void lv_draw_image_tiled_helper(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t
                                *coords, lv_draw_image_core_cb draw_core_cb)
```

Can be used by draw units for TILED images to handle the decoding and prepare everything for the actual image rendering

Parametri

- `t` -- pointer to a draw task
- `draw_dsc` -- the draw descriptor of the image
- `coords` -- the absolute coordinates of the image
- `draw_core_cb` -- a callback to perform the actual rendering

```
void lv_image_buf_get_transformed_area(lv_area_t *res, int32_t w, int32_t h, int32_t angle, uint16_t scale_x,
                                      uint16_t scale_y, const lv_point_t *pivot)
```

Get the area of a rectangle if its rotated and scaled

Parametri

- `res` -- store the coordinates here

- **w** -- width of the rectangle to transform
- **h** -- height of the rectangle to transform
- **angle** -- angle of rotation
- **scale_x** -- zoom in x direction, (256 no zoom)
- **scale_y** -- zoom in y direction, (256 no zoom)
- **pivot** -- x,y pivot coordinates of rotation

```
struct _lv_draw_image_sup_t
```

Membri Pubblici

lv_color_t alpha_color

const *lv_color32_t *palette*

uint32_t **palette_size**

14.15.8 lv_draw_label.h

Define

LV_DRAW_LABEL_NO_TXT_SEL

TypeDef

typedef void (***lv_draw_glyph_cb_t**)(lv_draw_task_t *t, lv_draw_glyph_dsc_t *dsc, *lv_draw_fill_dsc_t *fill_dsc*, const *lv_area_t *fill_area*)

Passed as a parameter to *lv_draw_label_iterate_characters* to draw the characters one by one

Param t

pointer to a draw task

Param dsc

pointer to *lv_draw_glyph_dsc_t* to describe the character to draw if NULL don't draw character

Param fill_dsc

pointer to a fill descriptor to draw a background for the character or underline or strike through if NULL do not fill anything

Param fill_area

the area to fill if NULL do not fill anything

Funzioni

void **lv_draw_letter_dsc_init**(*lv_draw_letter_dsc_t *dsc*)

void **lv_draw_label_dsc_init**(*lv_draw_label_dsc_t *dsc*)

Initialize a label draw descriptor

Parametri

dsc -- pointer to a draw descriptor

*lv_draw_label_dsc_t *lv_draw_task_get_label_dsc(lv_draw_task_t *task)*

Try to get a label draw descriptor from a draw task.

Parametri

task -- draw task

Ritorna

the task's draw descriptor or NULL if the task is not of type LV_DRAW_TASK_TYPE_LABEL

```
void lv_draw_glyph_dsc_init(lv_draw_glyph_dsc_t *dsc)
```

Initialize a glyph draw descriptor. Used internally.

Parametri

- **dsc** -- pointer to a draw descriptor

```
void lv_draw_label(lv_layer_t *layer, const lv_draw_label_dsc_t *dsc, const lv_area_t *coords)
```

Create a draw task to render a text

Parametri

- **layer** -- pointer to a layer
- **dsc** -- pointer to draw descriptor
- **coords** -- coordinates of the character

```
void lv_draw_character(lv_layer_t *layer, lv_draw_label_dsc_t *dsc, const lv_point_t *point, uint32_t unicode_letter)
```

Create a draw task to render a single character

Parametri

- **layer** -- pointer to a layer
- **dsc** -- pointer to draw descriptor
- **point** -- position of the label
- **unicode_letter** -- the letter to draw

```
void lv_draw_letter(lv_layer_t *layer, lv_draw_label_dsc_t *dsc, const lv_point_t *point)
```

Draw a single letter

Parametri

- **layer** -- pointer to a layer
- **dsc** -- pointer to draw descriptor
- **point** -- position of the label

```
void lv_draw_label_iterate_characters(lv_draw_task_t *t, const lv_draw_label_dsc_t *dsc, const lv_area_t *coords, lv_draw_glyph_cb_t cb)
```

Should be used during rendering the characters to get the position and other parameters of the characters

Parametri

- **t** -- pointer to a draw task
- **dsc** -- pointer to draw descriptor
- **coords** -- coordinates of the label
- **cb** -- a callback to call to draw each glyphs one by one

```
void lv_draw_unit_draw_letter(lv_draw_task_t *t, lv_draw_glyph_dsc_t *dsc, const lv_point_t *pos, const lv_font_t *font, uint32_t letter, lv_draw_glyph_cb_t cb)
```

Draw a single letter using the provided draw unit, glyph descriptor, position, font, and callback.

This function is responsible for rendering a single character from a text string, applying the necessary styling described by the glyph descriptor (dsc). It handles the retrieval of the glyph's description, checks its visibility within the clipping area, and invokes the callback (cb) to render the glyph at the specified position (pos) using the given font (font).

Parametri

- **t** -- Pointer to the drawing task.
- **dsc** -- Pointer to the descriptor containing styling for the glyph to be drawn.
- **pos** -- Pointer to the point coordinates where the letter should be drawn.
- **font** -- Pointer to the font containing the glyph.
- **letter** -- The Unicode code point of the letter to be drawn.
- **cb** -- Callback function to execute the actual rendering of the glyph.

```
struct lv_draw_label_dsc_t
```

Membri Pubblici***lv_draw_dsc_base_t* base**const char ***text**

The text to draw

***lv_point_t* text_size**

The size of the text

const lv_font_t ***font**

The font to use. Fallback fonts are also handled.

***lv_color_t* color**

Color of the text

int32_t **line_space**

Extra space between the lines

int32_t **letter_space**

Extra space between the characters

int32_t **ofs_x**

Offset the text with this value horizontally

int32_t **ofs_y**

Offset the text with this value vertically

int32_t **rotation**

Rotation of the letters in 0.1 degree unit

uint32_t **sel_start**

The first characters index for selection (not byte index). LV_DRAW_LABEL_NO_TXT_SEL for no selection

uint32_t **sel_end**

The last characters's index for selection (not byte index). LV_DRAW_LABEL_NO_TXT_SEL for no selection

***lv_color_t* sel_color**

Color of the selected characters

***lv_color_t* sel_bg_color**

Background color of the selected characters

uint32_t **text_length**

The number of characters to render. 0: means render until reaching the \0 termination.

***lv_text_align_t* align**

The alignment of the text LV_TEXT_ALIGN_LEFT/RIGHT/CENTER

lv_base_dir_t* **bidi_dir*

The base direction. Used when type setting Right-to-left (e.g. Arabic) texts

lv_opa_t* **opa*

Opacity of the text in 0...255 range. LV_OPA_TRANSP, LV_OPA_10, LV_OPA_20, .. LV_OPA_COVER can be used as well

lv_opa_t* **outline_stroke_opa*

Letter outline stroke opacity

lv_text_decor_t* **decor*

Text decoration, e.g. underline

lv_text_flag_t* **flag*

Some flags to control type setting

uint8_t* **text_local*

1: malloc a buffer and copy text there. 0: text will be valid during rendering.

uint8_t* **text_static*

Indicate that the text is constant and its pointer can be safely saved e.g. in a cache.

uint8_t* **has_bided*

1: already executed lv_bidi_process_paragraph. 0: has not been executed lv_bidi_process_paragraph.

lv_draw_label_hint_t* ***hint*

Pointer to an externally stored struct where some data can be cached to speed up rendering

lv_color_t* **outline_stroke_color****int32_t* **outline_stroke_width******struct *lv_draw_letter_dsc_t*****Membri Pubblici*****lv_draw_dsc_base_t* **base*******uint32_t* **unicode******const *lv_font_t* ***font*******lv_color_t* **color*******int32_t* **rotation*******int32_t* **scale_x****

```

int32_t scale_y

int32_t skew_x

int32_t skew_y

lv_point_t pivot

lv_opa_t opa

lv_text_decor_t decor

lv_blend_mode_t blend_mode

lv_opa_t outline_stroke_opa

int32_t outline_stroke_width

lv_color_t outline_stroke_color

```

14.15.9 lv_draw_label_private.h

struct _lv_draw_label_hint_t

#include <lv_draw_label_private.h> Store some info to speed up drawing of very large texts It takes a lot of time to get the first visible character because all the previous characters needs to be checked to calculate the positions. This structure stores an earlier (e.g. at -1000 px) coordinate and the index of that line. Therefore the calculations can start from here.

Membri Pubblici

int32_t line_start

Index of the line at y coordinate

int32_t y

Give the y coordinate of the first letter at line_start index. Relative to the label's coordinates

int32_t coord_y

The 'y1' coordinate of the label when the hint was saved. Used to invalidate the hint if the label has moved too much.

struct _lv_draw_glyph_dsc_t

Membri Pubblici

const void *glyph_data

Depends on format field, it could be image source or draw buf of bitmap or vector data.

```
lv_font_glyph_format_t format

const lv_area_t *letter_coords

const lv_area_t *bg_coords

lv_font_glyph_dsc_t *g

lv_color_t color

lv_opa_t opa

lv_color_t outline_stroke_color

lv_opa_t outline_stroke_opa

int32_t outline_stroke_width

int32_t rotation

lv_point_t pivot
```

Rotation pivot point associated with total glyph including line_height

```
lv_draw_buf_t *_draw_buf
```

a shared draw buf for get_bitmap, do not use it directly, use glyph_data instead

14.15.10 lv_draw_line.h

Funzioni

```
void lv_draw_line_dsc_init(lv_draw_line_dsc_t *dsc)
```

Initialize a line draw descriptor

Parametri

dsc -- pointer to a draw descriptor

```
lv_draw_line_dsc_t *lv_draw_task_get_line_dsc(lv_draw_task_t *task)
```

Try to get a line draw descriptor from a draw task.

Parametri

task -- draw task

Ritorna

the task's draw descriptor or NULL if the task is not of type LV_DRAW_TASK_TYPE_LINE

```
void lv_draw_line(lv_layer_t *layer, const lv_draw_line_dsc_t *dsc)
```

Create a line draw task

Parametri

- **layer** -- pointer to a layer
- **dsc** -- pointer to an initialized *lv_draw_line_dsc_t* variable

```
struct lv_draw_line_dsc_t
```

Membri Pubblici

lv_draw_dsc_base_t **base**

lv_point_precise_t **p1**

The first point of the line. If LV_USE_FLOAT is enabled float number can be also used

lv_point_precise_t **p2**

The second point of the line. If LV_USE_FLOAT is enabled float number can be also used

lv_color_t **color**

The color of the line

int32_t **width**

The width (thickness) of the line

int32_t **dash_width**

The length of a dash (0: don't dash)

int32_t **dash_gap**

The length of the gaps between dashes (0: don't dash)

lv_opa_t **opa**

Opacity of the line in 0...255 range. LV_OPA_TRANSP, LV_OPA_10, LV_OPA_20, ..
LV_OPA_COVER can be used as well

uint8_t **round_start**

Make the line start rounded

uint8_t **round_end**

Make the line end rounded

uint8_t **raw_end**

1: Do not bother with line ending (if it's not visible for any reason)

14.15.11 lv_draw_mask.h

Funzioni

void **lv_draw_mask_rect_dsc_init**(lv_draw_mask_rect_dsc_t *dsc)

Initialize a rectangle mask draw descriptor.

Parametri

dsc -- pointer to a draw descriptor

lv_draw_mask_rect_dsc_t ***lv_draw_task_get_mask_rect_dsc**(lv_draw_task_t *task)

Try to get a rectangle mask draw descriptor from a draw task.

Parametri

task -- draw task

Ritorna

the task's draw descriptor or NULL if the task is not of type LV_DRAW_TASK_TYPE_MASK_RECTANGLE

```
void lv_draw_mask_rect(lv_layer_t *layer, const lv_draw_mask_rect_dsc_t *dsc)
```

Create a draw task to mask a rectangle from the buffer

Parametri

- **layer** -- pointer to a layer
- **dsc** -- pointer to a draw descriptor

14.15.12 lv_draw_mask_private.h

```
struct _lv_draw_mask_rect_dsc_t
```

Membri Pubblici

lv_draw_dsc_base_t **base**

lv_area_t **area**

The area t mask.

int32_t radius

The radius of masking

uint32_t keep_outside

0: clear the content out of the area. 1: don't touch the area out of area

14.15.13 lv_draw_private.h

```
struct _lv_draw_task_t
```

#include <lv_draw_private.h> Modified by NXP in 2024

Membri Pubblici

lv_draw_task_t ***next**

lv_draw_task_type_t **type**

lv_area_t **area**

The area where to draw

lv_area_t **_real_area**

The real draw area. E.g. for shadow, outline, or transformed images it's different from area

lv_area_t **clip_area_original**

The original area which is updated

lv_area_t **clip_area**

The clip area of the layer is saved here when the draw task is created. As the clip area of the layer can be changed as new draw tasks are added its current value needs to be saved. Therefore during drawing the layer's clip area shouldn't be used as it might be already changed for other draw tasks.

lv_layer_t ***target_layer**

```
lv_draw_unit_t *draw_unit

volatile int state

void *draw_dsc
    int instead of lv_draw_task_state_t to be sure its atomic

lv_opa_t opa
    Opacity of the layer

uint8_t preferred_draw_unit_id
    The ID of the draw_unit which should take this task

uint8_t preference_score
    Set to which extent preferred_draw_unit_id is good at this task. 80: means 20% better (faster) than software rendering 100: the default value 110: means 10% worse (slower) than software rendering

struct _lv_draw_mask_t

Membri Pubblici

void *user_data

struct _lv_draw_unit_t

Membri Pubblici

lv_draw_unit_t *next

const char *name
    Name and ID of the draw unit, for debugging purposes only.

int32_t idx

int32_t (*dispatch_cb)(lv_draw_unit_t *draw_unit, lv_layer_t *layer)
    Called to try to assign a draw task to itself. lv_draw_get_next_available_task can be used to get an independent draw task. A draw task should be assigned only if the draw unit can draw it too
        Param draw_unit
            pointer to the draw unit
        Param layer
            pointer to a layer on which the draw task should be drawn
    Return
        >=0: The number of taken draw task: 0 means the task has not yet been completed.  

        1 means a new task has been accepted. -1: The draw unit wanted to work on a task but couldn't do that due to some errors (e.g. out of memory). It signals that LVGL should call the dispatcher later again to let draw unit try to start the rendering again.

int32_t (*evaluate_cb)(lv_draw_unit_t *draw_unit, lv_draw_task_t *task)
    Param draw_unit
    Param task
```

Return

```
int32_t (*wait_for_finish_cb)(lv_draw_unit_t *draw_unit)
```

Called to signal the unit to complete all tasks in order to return their ready status. This callback can be implemented in case of asynchronous task processing. Below is an example to show the difference between synchronous and asynchronous:

Synchronous: LVGL thread DRAW thread HW

```
task1 --> submit --> Receive task1 wait_for_finish() <&#8212; task1->state = READY <&#8212; Complete task1 task2 --> submit --> Receive task2 wait_for_finish() task2->state = READY <&#8212; Complete task2 task3 --> submit --> Receive task3 wait_for_finish() <&#8212; task3->state = READY <&#8212; Complete task3 task4 --> submit --> Receive task4 wait_for_finish() <&#8212; task4->state = READY <&#8212; Complete task4 NO MORE TASKS
```

Asynchronous: LVGL thread DRAW thread HW is IDLE task1 --> queue task1 submit --> Receive task1 task2 --> queue task2 is BUSY (with task1) task3 --> queue task3 still BUSY (with task1) task4 --> queue task4 becomes IDLE <— task1->state = READY <— Complete task1 submit --> Receive task2, task3, task4 NO MORE TASKS *wait_for_finish_cb()* wait_for_finish() <— Complete task2, task3, task4 <— task2->state = READY <— <— task3->state = READY <— <— task4->state = READY <—

Param draw_unit

Return

```
int32_t (*delete_cb)(lv_draw_unit_t *draw_unit)
```

Called to delete draw unit.

Param draw_unit

Return

```
void (*event_cb)(lv_event_t *event)
```

Called when an event is sent to the draw unit.

Param event

pointer to the event descriptor

```
struct lv_draw_global_info_t
```

Membri Pubblici

```
lv_draw_unit_t *unit_head
```

```
uint32_t unit_cnt
```

```
uint32_t used_memory_for_layers
```

```
volatile int dispatch_req
```

```
lv_mutex_t circle_cache_mutex
```

```
bool task_running
```

14.15.14 lv_draw_rect.h

Define

LV_RADIUS_CIRCLE

A very big radius to always draw as circle

Funzioni

void lv_draw_rect_dsc_init(*lv_draw_rect_dsc_t* *dsc)

Initialize a rectangle draw descriptor.

Parametri

dsc -- pointer to a draw descriptor

void lv_draw_fill_dsc_init(*lv_draw_fill_dsc_t* *dsc)

Initialize a fill draw descriptor.

Parametri

dsc -- pointer to a draw descriptor

***lv_draw_fill_dsc_t* *lv_draw_task_get_fill_dsc(*lv_draw_task_t* *task)**

Try to get a fill draw descriptor from a draw task.

Parametri

task -- draw task

Ritorna

the task's draw descriptor or NULL if the task is not of type LV_DRAW_TASK_TYPE_FILL

void lv_draw_fill(*lv_layer_t* *layer, const *lv_draw_fill_dsc_t* *dsc, const *lv_area_t* *coords)

Fill an area

Parametri

- **layer** -- pointer to a layer
- **dsc** -- pointer to an initialized draw descriptor variable
- **coords** -- the coordinates of the rectangle

void lv_draw_border_dsc_init(*lv_draw_border_dsc_t* *dsc)

Initialize a border draw descriptor.

Parametri

dsc -- pointer to a draw descriptor

***lv_draw_border_dsc_t* *lv_draw_task_get_border_dsc(*lv_draw_task_t* *task)**

Try to get a border draw descriptor from a draw task.

Parametri

task -- draw task

Ritorna

the task's draw descriptor or NULL if the task is not of type LV_DRAW_TASK_TYPE_BORDER

void lv_draw_border(*lv_layer_t* *layer, const *lv_draw_border_dsc_t* *dsc, const *lv_area_t* *coords)

Draw a border

Parametri

- **layer** -- pointer to a layer
- **dsc** -- pointer to an initialized draw descriptor variable
- **coords** -- the coordinates of the rectangle

void lv_draw_box_shadow_dsc_init(*lv_draw_box_shadow_dsc_t* *dsc)

Initialize a box shadow draw descriptor.

Parametri

dsc -- pointer to a draw descriptor

lv_draw_box_shadow_dsc_t ***lv_draw_task_get_box_shadow_dsc**(lv_draw_task_t *task)

Try to get a box shadow draw descriptor from a draw task.

Parametri

task -- draw task

Ritorna

the task's draw descriptor or NULL if the task is not of type LV_DRAW_TASK_TYPE_BOX_SHADOW

void **lv_draw_box_shadow**(lv_layer_t *layer, const *lv_draw_box_shadow_dsc_t* *dsc, const *lv_area_t* *coords)

Draw a box shadow

Parametri

- **layer** -- pointer to a layer
- **dsc** -- pointer to an initialized draw descriptor variable
- **coords** -- the coordinates of the rectangle

void **lv_draw_rect**(lv_layer_t *layer, const *lv_draw_rect_dsc_t* *dsc, const *lv_area_t* *coords)

The rectangle is a wrapper for fill, border, bg. image and box shadow. Internally fill, border, image and box shadow draw tasks will be created.

Parametri

- **layer** -- pointer to a layer
- **dsc** -- pointer to an initialized draw descriptor variable
- **coords** -- the coordinates of the rectangle

struct **lv_draw_rect_dsc_t**

Membri Pubblici

lv_draw_dsc_base_t **base**

int32_t **radius**

const void ***bg_image_src**

const void ***bg_image_symbol_font**

lv_color_t **bg_image_recolor**

lv_opa_t **bg_image_opa**

lv_opa_t **bg_image_recolor_opa**

uint8_t **bg_image_tiled**

lv_opa_t **bg_opa**

lv_opa_t **border_opa**

lv_opa_t **outline_opa**

lv_opa_t **shadow_opa**

lv_color_t* **bg_color*

First element of a gradient is a color, so it maps well here

lv_grad_dsc_t* **bg_grad*

```
const lv_image_colorkey_t *bg_image_colorkey
```

lv_color_t* **border_color*

```
int32_t border_width
```

lv_border_side_t* **border_side*

```
uint8_t border_post
```

lv_color_t* **outline_color*

```
int32_t outline_width
```

```
int32_t outline_pad
```

lv_color_t* **shadow_color*

```
int32_t shadow_width
```

```
int32_t shadow_offset_x
```

```
int32_t shadow_offset_y
```

```
int32_t shadow_spread
```

```
struct lv_draw_fill_dsc_t
```

Membri Pubblici***lv_draw_dsc_base_t* **base****

```
int32_t radius
```

Radius, LV_RADIUS_CIRCLE for max. radius

lv_opa_t* **opa*

Opacity in 0...255 range. LV_OPA_TRANSPI, LV_OPA_10, LV_OPA_20, .. LV_OPA_COVER can be used as well

lv_color_t* **color*

The color of the rectangle. If the gradient is set (grad.dir!=LV_GRAD_DIR_NONE) it's ignored.

***lv_grad_dsc_t* grad**

Describe a gradient. If `grad.dir` is not `LV_GRAD_DIR_NONE` color will be ignored

```
struct lv_draw_border_dsc_t
```

Membri Pubblici***lv_draw_dsc_base_t* base****int32_t radius**

Radius, `LV_RADIUS_CIRCLE` for max. radius

***lv_color_t* color**

The color of the border.

int32_t width

The width of the border in pixels

***lv_opa_t* opa**

Opacity in 0...255 range. `LV_OPA_TRANSP`, `LV_OPA_10`, `LV_OPA_20`, .. `LV_OPA_COVER` can be used as well

***lv_border_side_t* side**

`LV_BORDER_SIDE_NONE/LEFT/RIGHT/TOP/BOTTOM/FULL`. `LV_BORDER_SIDE_INTERNAL` is an information for upper layers and shouldn't be used here.

```
struct lv_draw_box_shadow_dsc_t
```

Membri Pubblici***lv_draw_dsc_base_t* base****int32_t radius**

Radius, `LV_RADIUS_CIRCLE` for max. radius

***lv_color_t* color**

Color of shadow

int32_t width

Width of the shadow. (radius of the blur)

int32_t spread

Make the rectangle larger with this value in all directions. Can be negative too.

int32_t ofs_x

Offset the rectangle horizontally.

int32_t ofs_y

Offset the rectangle vertically.

lv_opa_t opa

Opacity in 0...255 range. LV_OPA_TRANSP, LV_OPA_10, LV_OPA_20, .. LV_OPA_COVER can be used as well

uint8_t bg_cover

Set **bg_cover** to 1 if the background will cover the shadow. It's a hint to the renderer about it might skip some masking.

14.15.15 lv_draw_rect_private.h**14.15.16 lv_draw_triangle.h****Funzioni**

void lv_draw_triangle_dsc_init(lv_draw_triangle_dsc_t *draw_dsc)

Initialize a triangle draw descriptor

Parametri

draw_dsc -- pointer to a draw descriptor

lv_draw_triangle_dsc_t *lv_draw_task_get_triangle_dsc(lv_draw_task_t *task)

Try to get a triangle draw descriptor from a draw task.

Parametri

task -- draw task

Ritorna

the task's draw descriptor or NULL if the task is not of type LV_DRAW_TASK_TYPE_TRIANGLE

void lv_draw_triangle(lv_layer_t *layer, const lv_draw_triangle_dsc_t *draw_dsc)

Create a triangle draw task

Parametri

- **layer** -- pointer to a layer
- **draw_dsc** -- pointer to an initialized **lv_draw_triangle_dsc_t** object

struct lv_draw_triangle_dsc_t

Membri Pubblici

lv_draw_dsc_base_t base

lv_point_precise_t p[3]

Points of the triangle. If LV_USE_FLOAT is enabled floats can be used here

lv_color_t color

Color of the triangle

lv_opa_t opa

Opacity of the arc in 0...255 range. LV_OPA_TRANSP, LV_OPA_10, LV_OPA_20, .. LV_OPA_COVER can be used as well

lv_grad_dsc_t grad

Describe a gradient. If **grad.dir** is not LV_GRAD_DIR_NONE color will be ignored

14.15.17 lv_draw_triangle_private.h

14.15.18 lv_draw_vector.h

Typedef

```
typedef void (*vector_draw_task_cb)(void *ctx, const lv_vector_path_t *path, const lv_vector_path_ctx_t *dsc)
```

Enum

```
enum lv_vector_fill_t
```

Values:

```
enumerator LV_VECTOR_FILL_NONZERO
```

```
enumerator LV_VECTOR_FILL_EVENODD
```

```
enum lv_vector_stroke_cap_t
```

Values:

```
enumerator LV_VECTOR_STROKE_CAP_BUTT
```

```
enumerator LV_VECTOR_STROKE_CAP_SQUARE
```

```
enumerator LV_VECTOR_STROKE_CAP_ROUND
```

```
enum lv_vector_stroke_join_t
```

Values:

```
enumerator LV_VECTOR_STROKE_JOIN_MITER
```

```
enumerator LV_VECTOR_STROKE_JOIN_BEVEL
```

```
enumerator LV_VECTOR_STROKE_JOIN_ROUND
```

```
enum lv_vector_path_quality_t
```

Values:

```
enumerator LV_VECTOR_PATH_QUALITY_MEDIUM
```

```
enumerator LV_VECTOR_PATH_QUALITY_HIGH
```

```
enumerator LV_VECTOR_PATH_QUALITY_LOW
```

```
enum lv_vector_blend_t
```

Values:

```
enumerator LV_VECTOR_BLEND_SRC_OVER
```

enumerator **LV_VECTOR_BLEND_SRC_IN**

enumerator **LV_VECTOR_BLEND_DST_OVER**

enumerator **LV_VECTOR_BLEND_DST_IN**

enumerator **LV_VECTOR_BLEND_SCREEN**

enumerator **LV_VECTOR_BLEND_MULTIPLY**

enumerator **LV_VECTOR_BLEND_NONE**

enumerator **LV_VECTOR_BLEND_ADDITIVE**

enumerator **LV_VECTOR_BLEND_SUBTRACTIVE**

enum **lv_vector_path_op_t**

Values:

enumerator **LV_VECTOR_PATH_OP_MOVE_TO**

enumerator **LV_VECTOR_PATH_OP_LINE_TO**

enumerator **LV_VECTOR_PATH_OP_QUAD_TO**

enumerator **LV_VECTOR_PATH_OP_CUBIC_TO**

enumerator **LV_VECTOR_PATH_OP_CLOSE**

enum **lv_vector_draw_style_t**

Values:

enumerator **LV_VECTOR_DRAW_STYLE_SOLID**

enumerator **LV_VECTOR_DRAW_STYLE_PATTERN**

enumerator **LV_VECTOR_DRAW_STYLE_GRADIENT**

enum **lv_vector_gradient_spread_t**

Values:

enumerator **LV_VECTOR_GRADIENT_SPREAD_PAD**

enumerator **LV_VECTOR_GRADIENT_SPREAD_REPEAT**

enumerator **LV_VECTOR_GRADIENT_SPREAD_REFLECT**

enum **lv_vector_gradient_style_t**

Values:

enumerator **LV_VECTOR_GRADIENT_STYLE_LINEAR**

enumerator **LV_VECTOR_GRADIENT_STYLE_RADIAL**

enum **lv_vector_fill_units_t**

Values:

enumerator **LV_VECTOR_FILL_UNITS_OBJECT_BOUNDING_BOX**

enumerator **LV_VECTOR_FILL_UNITS_USER_SPACE_ON_USE**

Relative coordinates relative to the object bounding box.

Funzioni

void **lv_matrix_transform_point**(const lv_matrix_t *matrix, lv_fpoint_t *point)

Transform the coordinates of a point using given matrix

Parametri

- **matrix** -- pointer to a matrix
- **point** -- pointer to a point

void **lv_matrix_transform_path**(const lv_matrix_t *matrix, lv_vector_path_t *path)

Transform all the coordinates of a path using given matrix

Parametri

- **matrix** -- pointer to a matrix
- **path** -- pointer to a path

lv_vector_path_t ***lv_vector_path_create**(*lv_vector_path_quality_t* quality)

Create a vector graphic path object

Parametri

quality -- the quality hint of path

Ritorna

pointer to the created path object

void **lv_vector_path_copy**(lv_vector_path_t *target_path, const lv_vector_path_t *path)

Copy a path data to another

Parametri

- **target_path** -- pointer to a path
- **path** -- pointer to source path

void **lv_vector_path_clear**(lv_vector_path_t *path)

Clear path data

Parametri

path -- pointer to a path

void **lv_vector_path_delete**(lv_vector_path_t *path)

Delete the graphic path object

Parametri

path -- pointer to a path

void **lv_vector_path_move_to**(lv_vector_path_t *path, const lv_fpoint_t *p)

Begin a new sub path and set a point to path

Parametri

- **path** -- pointer to a path

- **p** -- pointer to a `lv_fpoint_t` variable

`void lv_vector_path_line_to(lv_vector_path_t *path, const lv_fpoint_t *p)`

Add a line to the path from last point to the point

Parametri

- **path** -- pointer to a path
- **p** -- pointer to a `lv_fpoint_t` variable

`void lv_vector_path_quad_to(lv_vector_path_t *path, const lv_fpoint_t *p1, const lv_fpoint_t *p2)`

Add a quadratic bezier line to the path from last point to the point

Parametri

- **path** -- pointer to a path
- **p1** -- pointer to a `lv_fpoint_t` variable for control point
- **p2** -- pointer to a `lv_fpoint_t` variable for end point

`void lv_vector_path_cubic_to(lv_vector_path_t *path, const lv_fpoint_t *p1, const lv_fpoint_t *p2, const lv_fpoint_t *p3)`

Add a cubic bezier line to the path from last point to the point

Parametri

- **path** -- pointer to a path
- **p1** -- pointer to a `lv_fpoint_t` variable for first control point
- **p2** -- pointer to a `lv_fpoint_t` variable for second control point
- **p3** -- pointer to a `lv_fpoint_t` variable for end point

`void lv_vector_path_arc_to(lv_vector_path_t *path, float radius_x, float radius_y, float rotate_angle, bool large_arc, bool clockwise, const lv_fpoint_t *p)`

Add ellipse arc to the path from last point to the point

Parametri

- **path** -- pointer to a path
- **radius_x** -- the x radius for ellipse arc
- **radius_y** -- the y radius for ellipse arc
- **rotate_angle** -- the rotate angle for arc
- **large_arc** -- true for large arc, otherwise small
- **clockwise** -- true for clockwise, otherwise anticlockwise
- **p** -- pointer to a `lv_fpoint_t` variable for end point

`void lv_vector_path_close(lv_vector_path_t *path)`

Close the sub path

Parametri

path -- pointer to a path

`void lv_vector_path_get_bounding(const lv_vector_path_t *path, lv_area_t *area)`

Get the bounding box of a path

Parametri

- **path** -- pointer to a path
- **area** -- pointer to a `lv_area_t` variable for bounding box

`void lv_vector_path_append_rectangle(lv_vector_path_t *path, float x, float y, float w, float h, float rx, float ry)`

Add a rectangle to the path by x/y/w/h. rx/ry are corner radii

Parametri

- **path** -- pointer to a path
- **x** -- the x coordinate of the top-left corner of the rectangle
- **y** -- the y coordinate of the top-left corner of the rectangle
- **w** -- the width of the rectangle
- **h** -- the height of the rectangle
- **rx** -- the horizontal radius for rounded rectangle
- **ry** -- the vertical radius for rounded rectangle

`static inline void lv_vector_path_append_rect(lv_vector_path_t *path, const lv_area_t *rect, float rx, float ry)`

Add a rectangle to the path (legacy api, recommend use `lv_vector_path_append_rectangle` instead)

Parametri

- **path** -- pointer to a path
- **rect** -- pointer to a *lv_area_t* variable
- **rx** -- the horizontal radius for rounded rectangle
- **ry** -- the vertical radius for rounded rectangle

void lv_vector_path_append_circle(lv_vector_path_t *path, const lv_fpoint_t *c, float rx, float ry)

Add a circle to the path

Parametri

- **path** -- pointer to a path
- **c** -- pointer to a *lv_fpoint_t* variable for center of the circle
- **rx** -- the horizontal radius for circle
- **ry** -- the vertical radius for circle

void lv_vector_path_append_arc(lv_vector_path_t *path, const lv_fpoint_t *c, float radius, float start_angle, float sweep, bool pie)

Add a arc to the path

Parametri

- **path** -- pointer to a path
- **c** -- pointer to a *lv_fpoint_t* variable for center of the circle
- **radius** -- the radius for arc
- **start_angle** -- the start angle for arc
- **sweep** -- the sweep angle for arc, could be negative
- **pie** -- true: draw a pie, false: draw a arc

void lv_vector_path_append_path(lv_vector_path_t *path, const lv_vector_path_t *subpath)

Add an sub path to the path

Parametri

- **path** -- pointer to a path
- **subpath** -- pointer to another path which will be added

lv_draw_vector_dsc_t *lv_draw_vector_dsc_create(lv_layer_t *layer)

Create a vector graphic descriptor

Parametri

layer -- pointer to a layer

Ritorna

pointer to the created descriptor

void lv_draw_vector_dsc_delete(lv_draw_vector_dsc_t *dsc)

Delete the vector graphic descriptor

Parametri

dsc -- pointer to a vector graphic descriptor

void lv_draw_vector_dsc_set_transform(lv_draw_vector_dsc_t *dsc, const lv_matrix_t *matrix)

Set a matrix to current transformation matrix. The new path shapes added by *lv_draw_vector_dsc_add_path* will use this matrix.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **matrix** -- pointer to a matrix

void lv_draw_vector_dsc_set_blend_mode(lv_draw_vector_dsc_t *dsc, lv_vector_blend_t blend)

Set blend mode for descriptor. The new path shapes added by *lv_draw_vector_dsc_add_path* will use this blend mode.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **blend** -- the blend mode to be set in *lv_vector_blend_t*

void lv_draw_vector_dsc_set_fill_color32(lv_draw_vector_dsc_t *dsc, lv_color32_t color)

Set the fill color for descriptor. The new path shapes added by *lv_draw_vector_dsc_add_path* will use this color.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **color** -- the color to be set in *lv_color32_t* format

```
void lv_draw_vector_dsc_set_fill_color(lv_draw_vector_dsc_t *dsc, lv_color_t color)
```

Set fill color for descriptor. The new path shapes added by `lv_draw_vector_dsc_add_path` will use this color.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **color** -- the color to be set in *lv_color_t* format

```
void lv_draw_vector_dsc_set_fill_opa(lv_draw_vector_dsc_t *dsc, lv_opa_t opa)
```

Set fill opacity for descriptor. The new path shapes added by `lv_draw_vector_dsc_add_path` will use this opacity.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **opa** -- the opacity to be set in *lv_opa_t* format

```
void lv_draw_vector_dsc_set_fill_rule(lv_draw_vector_dsc_t *dsc, lv_vector_fill_t rule)
```

Set fill rule for descriptor. The new path shapes added by `lv_draw_vector_dsc_add_path` will use this fill rule.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **rule** -- the fill rule to be set in *lv_vector_fill_t* format

```
void lv_draw_vector_dsc_set_fill_units(lv_draw_vector_dsc_t *dsc, const lv_vector_fill_units_t units)
```

Set the fill units for descriptor. The new path shapes added by `lv_draw_vector_dsc_add_path` will use this fill units.

 **Nota**

The units can be either relative to the object bounding box or absolute in user space. This API specifically affects the drawing position of the fill image and does not impact other elements.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **units** -- the units to be set in *lv_vector_fill_units_t* format

```
void lv_draw_vector_dsc_set_fill_image(lv_draw_vector_dsc_t *dsc, const lv_draw_image_dsc_t *img_dsc)
```

Set fill image for descriptor. The new path shapes added by `lv_draw_vector_dsc_add_path` will use this fill image.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **img_dsc** -- pointer to a *lv_draw_image_dsc_t* variable

```
void lv_draw_vector_dsc_set_fill_linear_gradient(lv_draw_vector_dsc_t *dsc, float x1, float y1, float x2, float y2)
```

Set fill linear gradient for descriptor. The new path shapes added by `lv_draw_vector_dsc_add_path` will use this gradient.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **x1** -- the x for start point
- **y1** -- the y for start point
- **x2** -- the x for end point
- **y2** -- the y for end point

```
void lv_draw_vector_dsc_set_fill_radial_gradient(lv_draw_vector_dsc_t *dsc, float cx, float cy, float radius)
```

Set fill radial gradient radius for descriptor. The new path shapes added by `lv_draw_vector_dsc_add_path` will use this gradient.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **cx** -- the x for center of the circle
- **cy** -- the y for center of the circle
- **radius** -- the radius for circle

```
void lv_draw_vector_dsc_set_fill_gradient_spread(lv_draw_vector_dsc_t *dsc,
                                                lv_vector_gradient_spread_t spread)
```

Set fill radial gradient spread for descriptor

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **spread** -- the gradient spread to be set in *lv_vector_gradient_spread_t* format

```
void lv_draw_vector_dsc_set_fill_gradient_color_stops(lv_draw_vector_dsc_t *dsc, const lv_grad_stop_t
                                                       *stops, uint16_t count)
```

Set fill gradient color stops for descriptor. The new path shapes added by *lv_draw_vector_dsc_add_path* will use this gradient color stops.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **stops** -- an array of *lv_grad_stop_t* variables
- **count** -- the number of stops in the array, range: 0..LV_GRADIENT_MAX_STOPS

```
void lv_draw_vector_dsc_set_fill_transform(lv_draw_vector_dsc_t *dsc, const lv_matrix_t *matrix)
```

Set a matrix to current fill transformation matrix. The new path shapes added by *lv_draw_vector_dsc_add_path* will use this matrix.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **matrix** -- pointer to a matrix

```
void lv_draw_vector_dsc_set_stroke_color32(lv_draw_vector_dsc_t *dsc, lv_color32_t color)
```

Set stroke color for descriptor. The new path shapes added by *lv_draw_vector_dsc_add_path* will use this color.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **color** -- the color to be set in *lv_color32_t* format

```
void lv_draw_vector_dsc_set_stroke_color(lv_draw_vector_dsc_t *dsc, lv_color_t color)
```

Set stroke color for descriptor. The new path shapes added by *lv_draw_vector_dsc_add_path* will use this color.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **color** -- the color to be set in *lv_color_t* format

```
void lv_draw_vector_dsc_set_stroke_opa(lv_draw_vector_dsc_t *dsc, lv_opa_t opa)
```

Set stroke opacity for descriptor

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **opa** -- the opacity to be set in *lv_opa_t* format

```
void lv_draw_vector_dsc_set_stroke_width(lv_draw_vector_dsc_t *dsc, float width)
```

Set stroke line width for descriptor. The new path shapes added by *lv_draw_vector_dsc_add_path* will use this stroke width.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **width** -- the stroke line width

```
void lv_draw_vector_dsc_set_stroke_dash(lv_draw_vector_dsc_t *dsc, float *dash_pattern, uint16_t
                                         dash_count)
```

Set stroke line dash pattern for descriptor. The new path shapes added by *lv_draw_vector_dsc_add_path* will use this dash.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **dash_pattern** -- an array of values that specify the segments of dash line
- **dash_count** -- the length of dash pattern array

void lv_draw_vector_dsc_set_stroke_cap(lv_draw_vector_dsc_t *dsc, lv_vector_stroke_cap_t cap)

Set stroke line cap style for descriptor The new path shapes added by `lv_draw_vector_dsc_add_path` will use this line cap.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **cap** -- the line cap to be set in `lv_vector_stroke_cap_t` format

void lv_draw_vector_dsc_set_stroke_join(lv_draw_vector_dsc_t *dsc, lv_vector_stroke_join_t join)

Set stroke line join style for descriptor

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **join** -- the line join to be set in `lv_vector_stroke_join_t` format

void lv_draw_vector_dsc_set_stroke_miter_limit(lv_draw_vector_dsc_t *dsc, uint16_t miter_limit)

Set stroke miter limit for descriptor The new path shapes added by `lv_draw_vector_dsc_add_path` will use this miter limit.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **miter_limit** -- the stroke miter_limit

void lv_draw_vector_dsc_set_stroke_linear_gradient(lv_draw_vector_dsc_t *dsc, float x1, float y1, float x2, float y2)

Set stroke linear gradient for descriptor The new path shapes added by `lv_draw_vector_dsc_add_path` will use this gradient.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **x1** -- the x for start point
- **y1** -- the y for start point
- **x2** -- the x for end point
- **y2** -- the y for end point

void lv_draw_vector_dsc_set_stroke_radial_gradient(lv_draw_vector_dsc_t *dsc, float cx, float cy, float radius)

Set stroke radial gradient for descriptor The new path shapes added by `lv_draw_vector_dsc_add_path` will use this gradient.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **cx** -- the x for center of the circle
- **cy** -- the y for center of the circle
- **radius** -- the radius for circle

void lv_draw_vector_dsc_set_stroke_gradient_spread(lv_draw_vector_dsc_t *dsc, lv_vector_gradient_spread_t spread)

Set stroke color stops for descriptor The new path shapes added by `lv_draw_vector_dsc_add_path` will use this gradient spread.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **spread** -- the gradient spread to be set in `lv_vector_gradient_spread_t` format

void lv_draw_vector_dsc_set_stroke_gradient_color_stops(lv_draw_vector_dsc_t *dsc, const lv_grad_stop_t *stops, uint16_t count)

Set stroke color stops for descriptor The new path shapes added by `lv_draw_vector_dsc_add_path` will use this color stops.

Parametri

- **dsc** -- pointer to a vector graphic descriptor

- **stops** -- an array of `lv_grad_stop_t` variables
- **count** -- the number of stops in the array

`void lv_draw_vector_dsc_set_stroke_transform(lv_draw_vector_dsc_t *dsc, const lv_matrix_t *matrix)`

Set a matrix to current stroke transformation matrix. The new path shapes added by `lv_draw_vector_dsc_add_path` will use this matrix.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **matrix** -- pointer to a matrix

`void lv_draw_vector_dsc_identity(lv_draw_vector_dsc_t *dsc)`

Set current transformation matrix to identity matrix The new path shapes added by `lv_draw_vector_dsc_add_path` will use this matrix.

Parametri

dsc -- pointer to a vector graphic descriptor

`void lv_draw_vector_dsc_scale(lv_draw_vector_dsc_t *dsc, float scale_x, float scale_y)`

Change the scale factor of current transformation matrix The new path shapes added by `lv_draw_vector_dsc_add_path` will use this scale.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **scale_x** -- the scale factor for the X direction
- **scale_y** -- the scale factor for the Y direction

`void lv_draw_vector_dsc_rotate(lv_draw_vector_dsc_t *dsc, float degree)`

Rotate current transformation matrix with origin The new path shapes added by `lv_draw_vector_dsc_add_path` will use this rotation.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **degree** -- angle to rotate

`void lv_draw_vector_dsc_translate(lv_draw_vector_dsc_t *dsc, float tx, float ty)`

Translate current transformation matrix to new position The new path shapes added by `lv_draw_vector_dsc_add_path` will use this rotation.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **tx** -- the amount of translate in x direction
- **ty** -- the amount of translate in y direction

`void lv_draw_vector_dsc_skew(lv_draw_vector_dsc_t *dsc, float skew_x, float skew_y)`

Change the skew factor of current transformation matrix The new path shapes added by `lv_draw_vector_dsc_add_path` will use this skew.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **skew_x** -- the skew factor for x direction
- **skew_y** -- the skew factor for y direction

`void lv_draw_vector_dsc_add_path(lv_draw_vector_dsc_t *dsc, const lv_vector_path_t *path)`

Add a graphic path to the draw list. It will use colors, opacity, matrix and other parameters set by `lv_draw_vector_dsc_set_fill_color()` and similar functions.

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **path** -- pointer to a path

`void lv_draw_vector_dsc_clear_area(lv_draw_vector_dsc_t *dsc, const lv_area_t *rect)`

Clear a rectangle area use current fill color

Parametri

- **dsc** -- pointer to a vector graphic descriptor
- **rect** -- the area to clear in the buffer

```
void lv_draw_vector(lv_draw_vector_dsc_t *dsc)
```

Draw all the vector graphic paths

Parametri

dsc -- pointer to a vector graphic descriptor

```
lv_draw_vector_dsc_t *lv_draw_task_get_vector_dsc(lv_draw_task_t *task)
```

Try to get a vector draw descriptor from a draw task.

Parametri

task -- draw task

Ritorna

the task's draw descriptor or NULL if the task is not of type LV_DRAW_TASK_TYPE_VECTOR

```
struct _lv_fpoint_t
```

Membri Pubblici

float x

float y

14.15.19 lv_draw_vector_private.h

Funzioni

```
void lv_vector_for_each_destroy_tasks(lv_ll_t *task_list, vector_draw_task_cb cb, void *used_data)
```

This is the main function to draw the accumulated vector tasks by passing them to a vector renderer callback. When the callback returns the processed vector task will be destroyed.

Parametri

- **task_list** -- pointer to the linked list in `lv_draw_vector_dsc_t` that stores the path shapes and their attributes.
- **cb** -- the callback used to iterate through the task
- **user_data** -- a custom pointer that will be passed to the callback

```
struct _lv_vector_path_t
```

#include <lv_draw_vector_private.h> Stores the shape of the path as arrays of operations and points. For example move to 10;20 then draw a line to 30;40 and draw an arc with 30 radius and 70° sweep.

`lv_vector_path_ctx_t` is also required to describe how to fill and stroke the path.

Membri Pubblici

`lv_vector_path_quality_t quality`

`lv_array_t ops`

`lv_array_t points`

```
struct _lv_vector_gradient_t
```

Membri Pubblici

lv_vector_gradient_style_t **style**

lv_grad_stop_t **stops[2]**

A gradient stop array

uint16_t stops_count

The number of used stops in the array

float x1

float y1

float x2

float y2

float cx

float cy

float cr

lv_vector_gradient_spread_t **spread**

struct _lv_vector_fill_dsc_t

Membri Pubblici

lv_vector_draw_style_t **style**

lv_color32_t **color**

lv_opa_t opa

lv_vector_fill_t **fill_rule**

lv_vector_fill_units_t **fill_units**

lv_draw_image_dsc_t **img_dsc**

lv_vector_gradient_t **gradient**

lv_matrix_t **matrix**

struct _lv_vector_stroke_dsc_t

Membri Pubblici

```
lv_vector_draw_style_t style
```

```
lv_color32_t color
```

```
lv_opa_t opa
```

```
float width
```

```
lv_array_t dash_pattern
```

```
lv_vector_stroke_cap_t cap
```

```
lv_vector_stroke_join_t join
```

```
uint16_t miter_limit
```

```
lv_vector_gradient_t gradient
```

```
lv_matrix_t matrix
```

```
struct _lv_vector_path_ctx_t
```

#include <lv_draw_vector_private.h> Stores how to fill, stroke, transform etc a given path

Membri Pubblici

```
lv_vector_fill_dsc_t fill_dsc
```

```
lv_vector_stroke_dsc_t stroke_dsc
```

```
lv_matrix_t matrix
```

```
lv_vector_blend_t blend_mode
```

```
lv_area_t scissor_area
```

```
struct _lv_draw_vector_dsc_t
```

Membri Pubblici

```
lv_draw_dsc_base_t base
```

```
lv_vector_path_ctx_t *ctx
```

The current colors, opacities, matrix, etc for the next task to be added by

`lv_ll_t *task_list`

Store path shapes and their attributes in a list as `lv_draw_vector_subtask_t`.

struct `lv_draw_vector_subtask_t`

`#include <lv_draw_vector_private.h>` Contains a path shape and its attributes together. It's a task that will be passed to the vector rendering engine. It's used in the `task_list` of `lv_draw_vector_dsc_t`.

Membri Pubblici

`lv_vector_path_t *path`

`lv_vector_path_ctx_t ctx`

14.15.20 lv_image_decoder.h

Typedef

`typedef lv_result_t (*lv_image_decoder_info_f_t)(lv_image_decoder_t *decoder, lv_image_decoder_dsc_t *dsc, lv_image_header_t *header)`

Get info from an image and store in the header

Param decoder

pointer to decoder object

Param dsc

pointer to decoder descriptor

Param header

store the info here

Return

`LV_RESULT_OK`: info written correctly; `LV_RESULT_INVALID`: failed

`typedef lv_result_t (*lv_image_decoder_open_f_t)(lv_image_decoder_t *decoder, lv_image_decoder_dsc_t *dsc)`

Open an image for decoding. Prepare it as it is required to read it later

Param decoder

pointer to the decoder the function associated with

Param dsc

pointer to decoder descriptor. `src`, `color` are already initialized in it.

`typedef lv_result_t (*lv_image_decoder_get_area_cb_t)(lv_image_decoder_t *decoder, lv_image_decoder_dsc_t *dsc, const lv_area_t *full_area, lv_area_t *decoded_area)`

Decode `full_area` pixels incrementally by calling in a loop. Set `decoded_area` values to `LV_COORD_MIN` on first call. Required only if the "open" function can't return with the whole decoded pixel array.

Param decoder

pointer to the decoder the function associated with

Param dsc

pointer to decoder descriptor

Param full_area

input parameter. the full area to decode after enough subsequent calls

Param decoded_area

input+output parameter. set the values to `LV_COORD_MIN` for the first call and to reset decoding. the decoded area is stored here after each call.

Return

`LV_RESULT_OK`: ok; `LV_RESULT_INVALID`: failed or there is nothing left to decode

```
typedef void (*lv_image_decoder_close_f_t)(lv_image_decoder_t *decoder, lv_image_decoder_dsc_t *dsc)
```

Close the pending decoding. Free resources etc.

Param decoder

pointer to the decoder the function associated with

Param dsc

pointer to decoder descriptor

```
typedef void (*lv_image_decoder_custom_draw_t)(lv_layer_t *layer, const lv_image_decoder_dsc_t *dsc, const lv_area_t *coords, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t *clip_area)
```

Custom drawing functions for special image formats.

Param layer

pointer to a layer

Param dsc

pointer to decoder descriptor

Param coords

the coordinates of the image

Param draw_dsc

the draw image descriptor

Param clip_area

the clip area of the image

Enum

enum **lv_image_src_t**

Source of image.

Values:

enumerator **LV_IMAGE_SRC_VARIABLE**

enumerator **LV_IMAGE_SRC_FILE**

Binary/C variable

enumerator **LV_IMAGE_SRC_SYMBOL**

File in filesystem

enumerator **LV_IMAGE_SRC_UNKNOWN**

Symbol (lv_symbol_def.h)

Funzioni

lv_result_t lv_image_decoder_get_info(const void *src, lv_image_header_t *header)

Get information about an image. Try the created image decoder one by one. Once one is able to get info that info will be used.

Parametri

- **src** -- the image source. Can be 1) File name: E.g. "S:folder/img1.png" (The drivers needs to registered via `lv_fs_drv_register()`) 2) Variable: Pointer to an `lv_image_dsc_t` variable 3) Symbol: E.g. `LV_SYMBOL_OK`
- **header** -- the image info will be stored here

Ritorna

`LV_RESULT_OK`: success; `LV_RESULT_INVALID`: wasn't able to get info about the image

```
lv_result_t lv_image_decoder_open(lv_image_decoder_dsc_t *dsc, const void *src, const
                                lv_image_decoder_args_t *args)
```

Open an image. Try the created image decoders one by one. Once one is able to open the image that decoder is saved in **dsc**

Parametri

- **dsc** -- describes a decoding session. Simply a pointer to an `lv_image_decoder_dsc_t` variable.
- **src** -- the image source. Can be 1) File name: E.g. "S:folder/img1.png" (The drivers needs to registered via `lv_fs_drv_register()`) 2) Variable: Pointer to an `lv_image_dsc_t` variable 3) Symbol: E.g. `LV_SYMBOL_OK`
- **args** -- args about how the image should be opened.

Ritorna

`LV_RESULT_OK`: opened the image. `dsc->decoded` and `dsc->header` are set.

`LV_RESULT_INVALID`: none of the registered image decoders were able to open the image.

```
lv_result_t lv_image_decoder_get_area(lv_image_decoder_dsc_t *dsc, const lv_area_t *full_area, lv_area_t
                                      *decoded_area)
```

Decode `full_area` pixels incrementally by calling in a loop. Set `decoded_area` to `LV_COORD_MIN` on first call.

Parametri

- **dsc** -- image decoder descriptor
- **full_area** -- input parameter. the full area to decode after enough subsequent calls
- **decoded_area** -- input+output parameter. set the values to `LV_COORD_MIN` for the first call and to reset decoding. the decoded area is stored here after each call.

Ritorna

`LV_RESULT_OK`: success; `LV_RESULT_INVALID`: an error occurred or there is nothing left to decode

```
void lv_image_decoder_close(lv_image_decoder_dsc_t *dsc)
```

Close a decoding session

Parametri

`dsc` -- pointer to `lv_image_decoder_dsc_t` used in `lv_image_decoder_open`

```
lv_image_decoder_t *lv_image_decoder_create(void)
```

Create a new image decoder

Ritorna

pointer to the new image decoder

```
void lv_image_decoder_delete(lv_image_decoder_t *decoder)
```

Delete an image decoder

Parametri

`decoder` -- pointer to an image decoder

```
lv_image_decoder_t *lv_image_decoder_get_next(lv_image_decoder_t *decoder)
```

Get the next image decoder in the linked list of image decoders

Parametri

`decoder` -- pointer to an image decoder or `NULL` to get the first one

Ritorna

the next image decoder or `NULL` if no more image decoder exists

```
void lv_image_decoder_set_info_cb(lv_image_decoder_t *decoder, lv_image_decoder_info_f_t info_cb)
```

Set a callback to get information about the image

Parametri

- **decoder** -- pointer to an image decoder
- **info_cb** -- a function to collect info about an image (fill an `lv_image_header_t` struct)

```
void lv_image_decoder_set_open_cb(lv_image_decoder_t *decoder, lv_image_decoder_open_f_t open_cb)
```

Set a callback to open an image

Parametri

- **decoder** -- pointer to an image decoder
- **open_cb** -- a function to open an image

```
void lv_image_decoder_set_get_area_cb(lv_image_decoder_t *decoder, lv_image_decoder_get_area_cb_t
                                      read_line_cb)
```

Set a callback to a decoded line of an image

Parametri

- **decoder** -- pointer to an image decoder
- **read_line_cb** -- a function to read a line of an image

```
void lv_image_decoder_set_close_cb(lv_image_decoder_t *decoder, lv_image_decoder_close_f_t close_cb)
```

Set a callback to close a decoding session. E.g. close files and free other resources.

Parametri

- **decoder** -- pointer to an image decoder
- **close_cb** -- a function to close a decoding session

```
lv_cache_entry_t *lv_image_decoder_add_to_cache(lv_image_decoder_t *decoder, lv_image_cache_data_t
                                                *search_key, const lv_draw_buf_t *decoded, void
                                                *user_data)
```

```
lv_draw_buf_t *lv_image_decoder_post_process(lv_image_decoder_dsc_t *dsc, lv_draw_buf_t *decoded)
```

Check the decoded image, make any modification if decoder args requires.

Nota

A new draw buf will be allocated if provided decoded is not modifiable or stride mismatch etc.

Parametri

- **dsc** -- pointer to a decoder descriptor
- **decoded** -- pointer to a decoded image to post process to meet dsc->args requirement.

Ritorna

post processed draw buffer, when it differs with decoded, it's newly allocated.

14.15.21 lv_image_decoder_private.h

Funzioni

```
void lv_image_decoder_init(uint32_t image_cache_size, uint32_t image_header_count)
```

Initialize the image decoder module

Parametri

- **image_cache_size** -- Image cache size in bytes. 0 to disable cache.
- **image_header_count** -- Number of header cache entries. 0 to disable header cache.

```
void lv_image_decoder_deinit(void)
```

Deinitialize the image decoder module

```
struct _lv_image_decoder_args_t
```

```
#include <lv_image_decoder_private.h> Image decoder args. It determines how to decoder an image, e.g.  
whether to premultiply the alpha or not. It should be passed to lv_img_decoder_open() function. If NULL is  
provided, default args are used.
```

Default args: all field are zero or false.

Membri Pubblici

```
bool stride_align
```

Whether stride should be aligned

bool premultiply

Whether image should be premultiplied or not after decoding

bool no_cache

When set, decoded image won't be put to cache, and decoder open will also ignore cache.

bool use_indexed

Decoded indexed image as is. Convert to ARGB8888 if false.

bool flush_cache

Whether to flush the data cache after decoding

struct **_lv_image_decoder_t**

Membri Pubblici

lv_image_decoder_info_f_t **info_cb**

lv_image_decoder_open_f_t **open_cb**

lv_image_decoder_get_area_cb_t **get_area_cb**

lv_image_decoder_close_f_t **close_cb**

lv_image_decoder_custom_draw_t **custom_draw_cb**

const char ***name**

void ***user_data**

struct **_lv_image_cache_data_t**

Membri Pubblici

lv_cache_slot_size_t **slot**

const void ***src**

lv_image_src_t **src_type**

const lv_draw_buf_t ***decoded**

const lv_image_decoder_t ***decoder**

void ***user_data**

struct **_lv_image_header_cache_data_t**

Membri Pubblici

```
const void *src
```

lv_image_src_t src_type

lv_image_header_t header

lv_image_decoder_t *decoder

```
struct _lv_image_decoder_dsc_t
```

#include <lv_image_decoder_private.h> Describe an image decoding session. Stores data about the decoding

Membri Pubblici

lv_image_decoder_t *decoder

The decoder which was able to open the image source

lv_image_decoder_args_t args

A copy of parameters of how this image is decoded

```
const void *src
```

The image source. A file path like "S:my_img.png" or pointer to an *lv_image_dsc_t* variable

lv_image_src_t src_type

Type of the source: file or variable. Can be set in open function if required

lv_fs_file_t file

lv_image_header_t header

Info about the opened image: color format, size, etc. MUST be set in open function

const lv_draw_buf_t *decoded

Pointer to a draw buffer where the image's data (pixels) are stored in a decoded, plain format. MUST be set in open or get_area_cbfunction

const lv_color32_t *palette

uint32_t palette_size

uint32_t time_to_open

How much time did it take to open the image. [ms] If not set *lv_image_cache* will measure and set the time to open

const char *error_msg

A text to display instead of the image when the image can't be opened. Can be set in open function or set NULL.

lv_cache_t *cache

`lv_cache_entry_t *cache_entry`

Point to cache entry information

`void *user_data`

Store any custom data here is required

14.15.22 lv_image_dsc.h

Define

`LV_IMAGE_HEADER_MAGIC`

Magic number for lvgl image, 9 means lvgl version 9 It must be neither a valid ASCII character nor larger than 0x80. See `lv_image_src_get_type`.

`LV_IMAGE_FLAGS_USER_MASK`

Flags reserved for user, lvgl won't use these bits.

Typedef

`typedef enum _lvimage_flags_t lv_image_flags_t`

Enum

`enum _lvimage_flags_t`

Values:

enumerator `LV_IMAGE_FLAGS_PREMULTIPLIED`

For RGB map of the image data, mark if it's pre-multiplied with alpha. For indexed image, this bit indicated palette data is pre-multiplied with alpha.

enumerator `LV_IMAGE_FLAGS_COMPRESSED`

The image data is compressed, so decoder needs to decode image firstly. If this flag is set, the whole image will be decompressed upon decode, and `get_area_cb` won't be necessary.

enumerator `LV_IMAGE_FLAGS_ALLOCATED`

The image is allocated from heap, thus should be freed after use.

enumerator `LV_IMAGE_FLAGS_MODIFIABLE`

If the image data is malloced and can be processed in place. In image decoder post processing, this flag means we modify it in-place.

enumerator `LV_IMAGE_FLAGS_CUSTOM_DRAW`

The image has custom drawing methods.

enumerator `LV_IMAGE_FLAGS_USER1`

Flags reserved for user, lvgl won't use these bits.

enumerator `LV_IMAGE_FLAGS_USER2`

enumerator **LV_IMAGE_FLAGS_USER3**

enumerator **LV_IMAGE_FLAGS_USER4**

enumerator **LV_IMAGE_FLAGS_USER5**

enumerator **LV_IMAGE_FLAGS_USER6**

enumerator **LV_IMAGE_FLAGS_USER7**

enumerator **LV_IMAGE_FLAGS_USER8**

enum **lv_image_compress_t**

Values:

enumerator **LV_IMAGE_COMPRESS_NONE**

enumerator **LV_IMAGE_COMPRESS_RLE**

LVGL custom RLE compression

enumerator **LV_IMAGE_COMPRESS_LZ4**

struct **lv_image_header_t**

Membri Pubblici

uint32_t magic

Magic number. Must be LV_IMAGE_HEADER_MAGIC

uint32_t cf

Color format: See [*lv_color_format_t*](#)

uint32_t flags

Image flags, see [*lv_image_flags_t*](#)

uint32_t w

uint32_t h

uint32_t stride

Number of bytes in a row

uint32_t reserved_2

Reserved to be used later

struct **lv_yuv_plane_t**

Membri Pubblici

`void *buf`

`uint32_t stride`

Number of bytes in a row

`union lv_yuv_buf_t`

Membri Pubblici

`lv_yuv_plane_t yuv`

packed format

`lv_yuv_plane_t y`

`lv_yuv_plane_t u`

`lv_yuv_plane_t v`

`struct lv_yuv_buf_t planar`

planar format with 3 plane

`lv_yuv_plane_t uv`

`struct lv_yuv_buf_t semi_planar`

planar format with 2 plane

`struct lv_image_dsc_t`

`#include <lv_image_dsc.h>` Struct to describe a constant image resource. It's similar to lv_draw_buf_t, but the data is constant.

Membri Pubblici

`lv_image_header_t header`

A header describing the basics of the image

`uint32_t data_size`

Size of the image in bytes

`const uint8_t *data`

Pointer to the data of the image

`const void *reserved`

A reserved field to make it has same size as lv_draw_buf_t

`const void *reserved_2`

A reserved field to make it has same size as lv_draw_buf_t

14.15.23 convert

`lv_draw_buf_convert.h`

Funzioni

`lv_result_t lv_draw_buf_convert_premultiply(lv_draw_buf_t *buf)`

Convert draw_buf to premultiplied format

Parametri

`buf` -- pointer to a draw buf

helium

`lv_draw_buf_convert_helium.h`

neon

`lv_draw_buf_convert_neon.h`

14.15.24 dma2d

`lv_draw_dma2d.h`

Funzioni

`void lv_draw_dma2d_init(void)`

`void lv_draw_dma2d_deinit(void)`

`void lv_draw_dma2d_transfer_complete_interrupt_handler(void)`

`lv_draw_dma2d_private.h`

Define

`LV_DRAW_DMA2D_ASYNC`

`LV_DRAW_DMA2D_CACHE`

Enum

enum `lv_draw_dma2d_output_cf_t`

Values:

enumerator `LV_DRAW_DMA2D_OUTPUT_CF_ARGB888`

enumerator `LV_DRAW_DMA2D_OUTPUT_CF_RGB888`

enumerator `LV_DRAW_DMA2D_OUTPUT_CF_RGB565`

enumerator `LV_DRAW_DMA2D_OUTPUT_CF_ARGB1555`

enumerator `LV_DRAW_DMA2D_OUTPUT_CF_ARGB4444`

enum `lv_draw_dma2d_fgbg_cf_t`

Values:

enumerator `LV_DRAW_DMA2D_FGBG_CF_ARGB8888`

enumerator `LV_DRAW_DMA2D_FGBG_CF_RGB888`

enumerator `LV_DRAW_DMA2D_FGBG_CF_RGB565`

enumerator `LV_DRAW_DMA2D_FGBG_CF_ARGB1555`

enumerator `LV_DRAW_DMA2D_FGBG_CF_ARGB4444`

enumerator `LV_DRAW_DMA2D_FGBG_CF_L8`

enumerator `LV_DRAW_DMA2D_FGBG_CF_AL44`

enumerator `LV_DRAW_DMA2D_FGBG_CF_AL88`

enumerator `LV_DRAW_DMA2D_FGBG_CF_L4`

enumerator `LV_DRAW_DMA2D_FGBG_CF_A8`

enumerator `LV_DRAW_DMA2D_FGBG_CF_A4`

enumerator `LV_DRAW_DMA2D_FGBG_CF_YCBCR`

enum `lv_draw_dma2d_mode_t`

Values:

enumerator `LV_DRAW_DMA2D_MODE_MEMORY_TO_MEMORY`

enumerator `LV_DRAW_DMA2D_MODE_MEMORY_TO_MEMORY_WITH_PFC`

enumerator `LV_DRAW_DMA2D_MODE_MEMORY_TO_MEMORY_WITH_BLENDING`

enumerator `LV_DRAW_DMA2D_MODE_REGISTER_TO_MEMORY`

enumerator `LV_DRAW_DMA2D_MODE_MEMORY_TO_MEMORY_WITH_BLENDING_AND_FIXED_COLOR_FG`

enumerator `LV_DRAW_DMA2D_MODE_MEMORY_TO_MEMORY_WITH_BLENDING_AND_FIXED_COLOR_BG`

enum `lv_draw_dma2d_alpha_mode_t`

Values:

enumerator `LV_DRAW_DMA2D_ALPHA_MODE_NO MODIFY IMAGE ALPHA CHANNEL`

enumerator `LV_DRAW_DMA2D_ALPHA_MODE_REPLACE_ALPHA_CHANNEL`

enumerator `LV_DRAW_DMA2D_ALPHA_MODE_MULTIPLY_IMAGE_ALPHA_CHANNEL`

Funzioni

```
void lv_draw_dma2d_opaque_fill(lv_draw_task_t *t, void *first_pixel, int32_t w, int32_t h, int32_t stride)
void lv_draw_dma2d_fill(lv_draw_task_t *t, void *first_pixel, int32_t w, int32_t h, int32_t stride)
void lv_draw_dma2d_opaque_image(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t *coords)
void lv_draw_dma2d_image(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t *coords)
lv_draw_dma2d_output_cf_t lv_draw_dma2d_cf_to_dma2d_output_cf(lv_color_format_t cf)
uint32_t lv_draw_dma2d_color_to_dma2d_color(lv_draw_dma2d_output_of_t cf, lv_color_t color)
void lv_draw_dma2d_configure_and_start_transfer(const lv_draw_dma2d_configuration_t *conf)

struct lv_draw_dma2d_configuration_t
```

Membri Pubblici

```
lv_draw_dma2d_mode_t mode
uint32_t w
uint32_t h
void *output_address
uint32_t output_offset
lv_draw_dma2d_output_cf_t output_cf
uint32_t reg_to_mem_mode_color
const void *fg_address
uint32_t fg_offset
lv_draw_dma2d_fgbg_cf_t fg_cf
uint32_t fg_color
uint32_t fg_alpha_mode
uint32_t fg_alpha
const void *bg_address
uint32_t bg_offset
```

```
lv_draw_dma2d_fgbg_cf_t bg_cf

uint32_t bg_color

uint32_t bg_alpha_mode

uint32_t bg_alpha
```

```
struct lv_draw_dma2d_cache_area_t
```

Membri Pubblici

```
const void *first_byte

uint32_t width_bytes

uint32_t height

uint32_t stride
```

```
struct lv_draw_dma2d_unit_t
```

Membri Pubblici

```
lv_draw_unit_t base_unit

lv_draw_task_t *volatile task_act
```

14.15.25 espressif

ppa

lv_draw_ppa.h

Funzioni

```
void lv_draw_ppa_init(void)

void lv_draw_ppa_deinit(void)

void lv_draw_buf_ppa_init_handlers(void)

void lv_draw_ppa_fill(lv_draw_task_t *t, const lv_draw_fill_dsc_t *dsc, const lv_area_t *coords)

void lv_draw_ppa_img(lv_draw_task_t *t, const lv_draw_image_dsc_t *dsc, const lv_area_t *coords)
```

lv_draw_ppa_private.h

Define

LV_PPA_NONBLOCKING_OPS

```
PPA_ALIGN_UP(x, align)
PPA_PTR_ALIGN_UP(p, align)
PPA_ALIGN_DOWN(x, align)
PPA_PTR_ALIGN_DOWN(p, align)
```

Typedef

```
typedef struct lv_draw_ppa_unit lv_draw_ppa_unit_t
```

Funzioni

```
static inline bool ppa_src_cf_supported(lv_color_format_t cf)
static inline bool ppa_dest_cf_supported(lv_color_format_t cf)
static inline ppa_fill_color_mode_t lv_color_format_to_ppa_fill(lv_color_format_t lv_fmt)
static inline ppa_blend_color_mode_t lv_color_format_to_ppa_blend(lv_color_format_t lv_fmt)
static inline ppa_srm_color_mode_t lv_color_format_to_ppa_srm(lv_color_format_t lv_fmt)

struct lv_draw_ppa_unit
```

Membri Pubblici

```
lv_draw_unit_t base_unit
lv_draw_task_t *task_act
ppa_client_handle_t srm_client
ppa_client_handle_t fill_client
ppa_client_handle_t blend_client
uint8_t *buf
```

14.15.26 eve

lv_draw_eve.h

Funzioni

```
void lv_draw_eve_init(void)
void lv_draw_eve_set_display_data(lv_display_t *disp, const lv_draw_eve_parameters_t *params,
                                 lv_draw_eve_operation_cb_t op_cb)
```

lv_draw_eve_private.h**Define**

```
LV_DRAW_EVE_WRITE_BUFFER_SIZE_INTERNAL
```

```
DEGREES(n)
```

```
F16(x)
```

lv_draw_eve_unit_g**Funzioni**

```
void lv_draw_eve_image(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t *coords)
bool lv_draw_eve_image_src_check(const void *src)
uint32_t lv_draw_eve_upload_image(bool burst_is_active, const lv_image_dsc_t *img_dsc)
void lv_draw_eve_fill(lv_draw_task_t *t, const lv_draw_fill_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_eve_border(lv_draw_task_t *t, const lv_draw_border_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_eve_line(lv_draw_task_t *t, const lv_draw_line_dsc_t *dsc)
void lv_draw_eve_label(lv_draw_task_t *t, const lv_draw_label_dsc_t *dsc, const lv_area_t *coords)
bool lv_draw_eve_label_font_check(const lv_font_t *font)
uint32_t lv_draw_eve_label_upload_glyph(bool burst_is_active, const lv_font_fmt_txt_dsc_t *font_dsc,
                                       uint32_t gid_index)
void lv_draw_eve_arc(lv_draw_task_t *t, const lv_draw_arc_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_eve_triangle(lv_draw_task_t *t, const lv_draw_triangle_dsc_t *dsc)
```

```
struct lv_draw_eve_ramg_hash_table_cell_t
```

Membri Pubblici

```
uintptr_t key
```

```
uint32_t addr
```

```
struct lv_draw_eve_ramg_t
```

Membri Pubblici

```
uint32_t ramg_addr_end
```

```
uint32_t hash_table_cell_count
```

```
uint32_t hash_table_cells_occupied
```

`lv_draw_eve_ramg_hash_table_cell_t *hash_table`

struct `_lv_draw_eve_unit_t`

Membri Pubblici

`lv_draw_unit_t base_unit`

`lv_draw_task_t *task_act`

`lv_display_t *disp`

`lv_draw_eve_ramg_t ramg`

`lv_draw_eve_parameters_t params`

`lv_draw_eve_operation_cb_t op_cb`

`uint32_t lv_eve_write_buf_len`

`uint8_t lv_eve_write_buf[2048]`

`lv_draw_eve_ram_g.h`

Define

`LV_DRAW_EVE_RAMG_OUT_OF_RAMG`

Funzioni

bool `lv_draw_eve_ramg_get_addr(uint32_t *addr_dst, uintptr_t key, uint32_t addr_size, uint32_t addr_align)`

`lv_draw_eve_target.h`

Typedef

typedef void (*`lv_draw_eve_operation_cb_t`)(`lv_display_t *disp, lv_draw_eve_operation_t operation, void *data, uint32_t length`)

Enum

enum `lv_draw_eve_operation_t`

Values:

enumerator `LV_DRAW_EVE_OPERATION_POWERDOWN_SET`

set the "PD_N" pin low

enumerator `LV_DRAW_EVE_OPERATION_POWERDOWN_CLEAR`

set the "PD_N" pin high

enumerator **LV_DRAW_EVE_OPERATION_CS_ASSERT**

set the "CS_N" pin low

enumerator **LV_DRAW_EVE_OPERATION_CS_DEASSERT**

set the "CS_N" pin high

enumerator **LV_DRAW_EVE_OPERATION_SPI_SEND**

send length bytes of data over SPI

enumerator **LV_DRAW_EVE_OPERATION_SPI_RECEIVE**

receive length bytes into data from SPI

struct **lv_draw_eve_parameters_t**

Membri Pubblici

uint16_t hor_res

active display width

uint16_t ver_res

active display height

uint16_t hcycle

total number of clocks per line, incl front/back porch

uint16_t hoffset

start of active line

uint16_t hsync0

start of horizontal sync pulse

uint16_t hsync1

end of horizontal sync pulse

uint16_t vcycle

total number of lines per screen, including pre/post

uint16_t voffset

start of active screen

uint16_t vsync0

start of vertical sync pulse

uint16_t vsync1

end of vertical sync pulse

uint8_t swizzle

FT8xx output to LCD - pin order

uint8_t pclkpol

LCD data is clocked in on this PCLK edge

uint8_t cspread

helps with noise, when set to 1 fewer signals are changed simultaneously, reset-default: 1

uint8_t pclk

60MHz / pclk = pclk frequency

bool has_crystal

has an external clock crystal

bool has_gt911

has a touch controller

uint8_t backlight_pwm

backlight PWM duty cycle 0 = off, 128 = max

uint16_t backlight_freq

backlight PWM frequency. try 4000 if unsure

lv_eve.h

Enum

enum lv_eve_primitive_t*Values:*enumerator **LV_EVE_PRIMITIVE_ZERO_VALUE**enumerator **LV_EVE_PRIMITIVE_BITMAPS**enumerator **LV_EVE_PRIMITIVE_POINTS**enumerator **LV_EVE_PRIMITIVE_LINES**enumerator **LV_EVE_PRIMITIVE_LINE_STRIP**enumerator **LV_EVE_PRIMITIVE_EDGE_STRIP_R**enumerator **LV_EVE_PRIMITIVE_EDGE_STRIP_L**enumerator **LV_EVE_PRIMITIVE_EDGE_STRIP_A**enumerator **LV_EVE_PRIMITIVE_EDGE_STRIP_B**enumerator **LV_EVE_PRIMITIVE_RECTS**

Funzioni

```
void lv_eve_save_context(void)
void lv_eve_restore_context(void)
void lv_eve_scissor(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2)
void lv_eve_primitive(uint8_t context)
void lv_eve_color(lv_color_t color)
void lv_eve_color_opa(lv_opa_t opa)
void lv_eve_line_width(int32_t width)
void lv_eve_point_size(uint16_t radius)
void lv_eve_vertex_2f(int16_t x, int16_t y)
void lv_eve_color_mask(uint8_t r, uint8_t g, uint8_t b, uint8_t a)
void lv_eve_stencil_func(uint8_t func, uint8_t ref, uint8_t mask)
void lv_eve_stencil_op(uint8_t sfail, uint8_t spass)
void lv_eve_blend_func(uint8_t src, uint8_t dst)
void lv_eve_draw_circle_simple(int16_t coord_x1, int16_t coord_y1, uint16_t radius_t)
void lv_eve_draw_rect_simple(int16_t coord_x1, int16_t coord_y1, int16_t coord_x2, int16_t coord_y2,
                             uint16_t radius)
void lv_eve_mask_round(int16_t coord_x1, int16_t coord_y1, int16_t coord_x2, int16_t coord_y2, int16_t
                      radius)
```

void **lv_eve_bitmap_source**(uint32_t addr)

Set the bitmap source to **addr**. SPI transmission will occur unless it is already set to this value. The bitmap source is not part of the saved and restored context.

Parametri

addr -- the remote EVE memory address to set as the bitmap source

void **lv_eve_bitmap_size**(uint8_t filter, uint8_t wrapx, uint8_t wrapy, uint16_t width, uint16_t height)

Set the bitmap size and sampling parameters. SPI transmission will occur unless the currently set parameters are already these. The bitmap size is not part of the saved and restored context.

Parametri

- **filter** -- the sampling method. Either EVE_NEAREST or EVE_BILINEAR
- **wrapx** -- the out of bounds sampling behavior in the X direction. Either EVE_BORDER or EVE_REPEAT
- **wrapy** -- the out of bounds sampling behavior in the Y direction. Either EVE_BORDER or EVE_REPEAT
- **width** -- the width of the bitmap in pixels
- **height** -- the height of the bitmap in pixels

void **lv_eve_bitmap_layout**(uint8_t format, uint16_t linestride, uint16_t height)

Set the bitmap format/layout parameters. SPI transmission will occur unless the currently set parameters are already these. The bitmap layout is not part of the saved and restored context.

Parametri

- **format** -- an eve color format value like EVE_RGB565
- **linestride** -- the stride of the bitmap rows in bytes
- **height** -- the number of rows in the bitmap

struct **lv_eve_drawing_context_t**

Membri Pubblici

```
lv_eve_primitive_t primitive
```

```
lv_color_t color
```

```
lv_opa_t opa
```

```
int32_t line_width
```

```
uint16_t point_size
```

```
uint8_t color_mask[4]
```

```
uint8_t stencil_func[3]
```

```
uint8_t stencil_op[2]
```

```
uint8_t blend_func[2]
```

```
uint16_t scx
```

```
uint16_t scy
```

```
struct lv_eve_drawing_state_t
```

Membri Pubblici

```
uint32_t bitmap_source
```

```
uint32_t bitmap_size
```

```
uint32_t bitmap_size_h
```

```
uint32_t bitmap_layout
```

```
uint32_t bitmap_layout_h
```

14.15.27 nema_gfx

lv_draw_nema_gfx.h

MIT License

Copyright (c) 2008-24 Think Silicon Single Member PC

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice (including the next paragraph) shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Funzioni

```
void lv_draw_nema_gfx_init(void)
void lv_draw_nema_gfx_deinit(void)
void lv_draw_nema_gfx_fill(lv_draw_task_t *t, const lv_draw_fill_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_nema_gfx_triangle(lv_draw_task_t *t, const lv_draw_triangle_dsc_t *dsc)
void lv_draw_nema_gfx_img(lv_draw_task_t *t, const lv_draw_image_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_nema_gfx_label(lv_draw_task_t *t, const lv_draw_label_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_nema_gfx_label_init(lv_draw_unit_t *draw_unit)
void lv_draw_nema_gfx_layer(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t
                           *coords)
void lv_draw_nema_gfx_line(lv_draw_task_t *t, const lv_draw_line_dsc_t *dsc)
void lv_draw_nema_gfx_border(lv_draw_task_t *t, const lv_draw_border_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_nema_gfx_arc(lv_draw_task_t *t, const lv_draw_arc_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_nema_gfx_vector(lv_draw_task_t *t, const lv_draw_vector_dsc_t *dsc, const lv_area_t *coords)

struct lv_draw_nema_gfx_unit_t
```

Membri Pubblici

```
lv_draw_unit_t base_unit
lv_draw_task_t *task_act
uint32_t idx
nema_cmdlist_t cl
NEMA_VG_PAINT_HANDLE paint
NEMA_VG_GRAD_HANDLE gradient
NEMA_VG_PATH_HANDLE path
```

lv_draw_nema_gfx_utils.h

MIT License

Copyright (c) 2008-24 Think Silicon Single Member PC

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice (including the next paragraph) shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Define**NEMA_VIRT2PHYS****LV_NEMA_GFX_COLOR_FORMAT****LV_NEMA_GFX_FORMAT_MULTIPLIER****Funzioni**bool **lv_nemagfx_is_cf_supported**(*lv_color_format_t* cf)Check if *lv_color_format_t* is supported.**Parametri**

cf -- The LVGL color format

Ritorna

True/false

uint32_t **lv_nemagfx_cf_to_nema**(*lv_color_format_t* cf)Convert a *lv_color_format_t* to a Nema color format.**Parametri**

cf -- The LVGL color format

Ritorna

The Nema color format

uint32_t **lv_nemagfx_blending_mode**(*lv_blend_mode_t* lv_blend_mode)

Get NemaGFX blending mode

Parametri*lv_blend_mode* -- [in] The LVGL blend mode**Ritorna**

NemaGFX blending mode

void **lv_nemagfx_grad_set**(NEMA_VG_GRAD_HANDLE gradient, *lv_grad_desc_t* lv_grad, *lv_opa_t* opa)

Get NemaGFX blending mode

Parametri

- **gradient** -- [in] NemaGFX Gradient Buffer
- **lv_grad** -- [in] Gradient descriptor
- **opa** -- [in] Descriptor's opacity

lv_nema_gfx_path.h

MIT License

Copyright (c) 2008-24 Think Silicon Single Member PC

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice (including the next paragraph) shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Funzioni

```
lv_nema_gfx_path_t *lv_nema_gfx_path_create(void)
void lv_nema_gfx_path_alloc(lv_nema_gfx_path_t *nema_gfx_path)
void lv_nema_gfx_path_destroy(lv_nema_gfx_path_t *nema_gfx_path)
void lv_nema_gfx_path_move_to(lv_nema_gfx_path_t *nema_gfx_path, float x, float y)
void lv_nema_gfx_path_line_to(lv_nema_gfx_path_t *nema_gfx_path, float x, float y)
void lv_nema_gfx_path_quad_to(lv_nema_gfx_path_t *nema_gfx_path, float cx, float cy, float x, float y)
void lv_nema_gfx_path_cubic_to(lv_nema_gfx_path_t *nema_gfx_path, float cx1, float cy1, float cx2, float cy2,
float x, float y)

void lv_nema_gfx_path_end(lv_nema_gfx_path_t *nema_gfx_path)
```

struct **lv_nema_gfx_path_t****Membri Pubblici**NEMA_VG_PATH_HANDLE **path**NEMA_VG_PAINT_HANDLE **paint**float ***data**uint8_t ***seg**uint32_t **data_size**uint32_t **seg_size**

14.15.28 nxp

g2d

lv_draw_g2d.h

Typedef

```
typedef struct lv_draw_g2d_unit lv_draw_g2d_unit_t
```

Copyright 2024 NXP

SPDX-License-Identifier: MIT

Funzioni

```
void lv_draw_g2d_init(void)
```

```
void lv_draw_g2d_deinit(void)
```

```
void lv_draw_buf_g2d_init_handlers(void)
```

```
void lv_draw_g2d_fill(lv_draw_task_t *t)
```

```
void lv_draw_g2d_img(lv_draw_task_t *t)
```

```
struct lv_draw_g2d_unit
```

#include <lv_draw_g2d.h> Copyright 2024 NXP

SPDX-License-Identifier: MIT

Membri Pubblici

lv_draw_unit_t **base_unit**

lv_draw_task_t ***task_act**

lv_g2d_buf_map.h

Typedef

```
typedef struct lv_map_item lv_map_item_t
```

Copyright 2024 NXP

SPDX-License-Identifier: MIT

```
typedef struct lv_buf_map lv_buf_map_t
```

Funzioni

```
void g2d_create_buf_map(void)
```

```
void g2d_free_buf_map(void)
```

```
void g2d_insert_buf_map(void *key, struct g2d_buf *value)
```

```
struct g2d_buf *g2d_search_buf_map(void *key)
```

```
void g2d_free_item(void *key)
```

```
void g2d_print_table(void)

struct lv_map_item
#include <lv_g2d_buf_map.h> Copyright 2024 NXP
SPDX-License-Identifier: MIT
```

Membri Pubblici

```
void *key

struct g2d_buf *value

struct lv_buf_map
```

Membri Pubblici

```
lv_map_item_t **items

lv_array_t **overflow_list

int size

int count
```

lv_g2d_utils.h**Define**

```
G2D_ASSERT(expr)
Copyright 2024 NXP
SPDX-License-Identifier: MIT

G2D_ASSERT_MSG(expr, msg)
```

Typedef

```
typedef enum g2d_format g2d_format_t
```

Funzioni

```
g2d_format_t g2d_get_buf_format(lv_color_format_t cf)

uint32_t g2d_rgba_to_u32(lv_color_t color)

int32_t g2d_get_buf_fd(const lv_draw_buf_t *draw_buf)

void g2d_set_handle(void *handle)

void *g2d_get_handle(void)

void g2d_rotate(lv_draw_buf_t *buf1, lv_draw_buf_t *buf2, int32_t width, int32_t height, uint32_t rotation,
               lv_color_format_t cf)
```

pxp**lv_draw_pxp.h****Typedef**

```
typedef struct _lv_draw_pxp_unit_t lv_draw_pxp_unit_t
```

Copyright 2022-2024 NXP

SPDX-License-Identifier: MIT

Funzioni

```
void lv_draw_pxp_init(void)
```

```
void lv_draw_pxp_deinit(void)
```

```
void lv_draw_pxp_rotate(const void *src_buf, void *dest_buf, int32_t src_width, int32_t src_height, int32_t src_stride, int32_t dest_stride, lv_display_rotation_t rotation, lv_color_format_t cf)
```

```
void lv_draw_buf_pxp_init_handlers(void)
```

```
void lv_draw_pxp_fill(lv_draw_task_t *t)
```

```
void lv_draw_pxp_img(lv_draw_task_t *t)
```

```
void lv_draw_pxp_layer(lv_draw_task_t *t)
```

```
struct _lv_draw_pxp_unit_t
```

#include <lv_draw_pxp.h> Copyright 2022-2024 NXP

SPDX-License-Identifier: MIT

Membri Pubblici

lv_draw_unit_t **base_unit**

lv_draw_task_t ***task_act**

lv_pxp_cfg.h**Define****PXP_ID**

Copyright 2020-2023 NXP

SPDX-License-Identifier: MIT PXP module instance to use

PXP_IRQ_ID

PXP interrupt line ID

Funzioni

```
void lv_pxp_init(void)
```

Reset and initialize PXP device. This function should be called as a part of display init sequence.

```
void lv_pxp_deinit(void)
```

Disable PXP device. Should be called during display deinit sequence.

```
void lv_pxpx_reset(void)
    Reset PXP device.

void lv_pxpx_run(void)
    Clear cache and start PXP.

void lv_pxpx_wait(void)
    Wait for PXP completion.

struct ppxp_cfg_t
    #include <lv_pxpx_cfg.h> NXP PXP device configuration.
```

Membri Pubblici

```
void (*ppx_interrupt_init)(void)
    Callback for PXP interrupt initialization

void (*ppx_interrupt_deinit)(void)
    Callback for PXP interrupt de-initialization

void (*ppx_run)(void)
    Callback for PXP start

void (*ppx_wait)(void)
    Callback for waiting of PXP completion
```

lv_pxpx_osa.h

Funzioni

```
void PXP_IRQHandler(void)
    Copyright 2020, 2022-2023 NXP
    SPDX-License-Identifier: MIT PXP device interrupt handler. Used to check PXP task completion status.

ppx_cfg_t *ppx_get_default_cfg(void)
    Get the PXP default configuration.
```

lv_pxpx_utils.h

Define

```
PXP_ASSERT(expr)
    Copyright 2023-2024 NXP
    SPDX-License-Identifier: MIT

PXP_ASSERT_MSG(expr, msg)
```

Funzioni

```
ppx_output_pixel_format_t ppx_get_out_px_format(lv_color_format_t cf)
ppx_as_pixel_format_t ppx_get_as_px_format(lv_color_format_t cf)
ppx_ps_pixel_format_t ppx_get_ps_px_format(lv_color_format_t cf)

bool ppx_buf_aligned(const void *buf, uint32_t stride)
```

14.15.29 opengles

lv_draw_opengles.h

Funzioni

```
void lv_draw_opengles_init(void)
void lv_draw_opengles_deinit(void)
```

14.15.30 renesas

dave2d

lv_draw_dave2d.h

Define

```
D2_FIX4(x)
D2_INT4(x)
D2_FLOOR4(x)
D2_CEIL4(x)
D2_FRAC4(x)
D2_FIX16(x)
D2_INT16(x)
D2_FLOOR16(x)
D2_CEIL16(x)
D2_FRAC16(x)
```

Funzioni

```
void lv_draw_dave2d_init(void)
void lv_draw_dave2d_image(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t *coords)
void lv_draw_dave2d_fill(lv_draw_task_t *t, const lv_draw_fill_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_dave2d_border(lv_draw_task_t *t, const lv_draw_border_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_dave2d_box_shadow(lv_draw_task_t *t, const lv_draw_box_shadow_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_dave2d_label(lv_draw_task_t *t, const lv_draw_label_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_dave2d_arc(lv_draw_task_t *t, const lv_draw_arc_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_dave2d_line(lv_draw_task_t *t, const lv_draw_line_dsc_t *dsc)
void lv_draw_dave2d_layer(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t *coords)
void lv_draw_dave2d_triangle(lv_draw_task_t *t, const lv_draw_triangle_dsc_t *dsc)
void lv_draw_dave2d_mask_rect(lv_draw_task_t *t, const lv_draw_mask_rect_dsc_t *dsc, const lv_area_t *coords)
```

```
void lv_draw_dave2d_transform(lv_draw_task_t *t, const lv_area_t *dest_area, const void *src_buf, int32_t
    src_w, int32_t src_h, int32_t src_stride, const lv_draw_image_dsc_t *draw_dsc,
    const lv_draw_image_sup_t *sup, lv_color_format_t cf, void *dest_buf)
```

```
struct lv_draw_dave2d_unit_t
```

Membri Pubblici

lv_draw_unit_t **base_unit**

lv_draw_task_t ***task_act**

uint32_t **idx**

d2_device ***d2_handle**

d2_renderbuffer ***renderbuffer**

d2_renderbuffer ***label_renderbuffer**

lv_draw_dave2d_utils.h

Funzioni

d2_color **lv_draw_dave2d_lv_colour_to_d2_colour**(lv_color_t color)

d2_u32 **lv_draw_dave2d_lv_colour_fmt_to_d2_fmt**(lv_color_format_t colour_format)

void **d2_framebuffer_from_layer**(d2_device *handle, lv_layer_t *layer)

bool **lv_draw_dave2d_is_dest_cf_supported**(lv_color_format_t cf)

14.15.31 sdl

lv_draw_sdl.h

Funzioni

void **lv_draw_sdl_init**(void)

void **lv_draw_sdl_image**(lv_draw_unit_t *draw_unit, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t *coords)

void **lv_draw_sdl_fill**(lv_draw_unit_t *draw_unit, const lv_draw_fill_dsc_t *dsc, const lv_area_t *coords)

void **lv_draw_sdl_border**(lv_draw_unit_t *draw_unit, const lv_draw_border_dsc_t *dsc, const lv_area_t *coords)

void **lv_draw_sdl_box_shadow**(lv_draw_unit_t *draw_unit, const lv_draw_box_shadow_dsc_t *dsc, const lv_area_t *coords)

void **lv_draw_sdl_label**(lv_draw_unit_t *draw_unit, const lv_draw_label_dsc_t *dsc, const lv_area_t *coords)

void **lv_draw_sdl_arc**(lv_draw_unit_t *draw_unit, const lv_draw_arc_dsc_t *dsc, const lv_area_t *coords)

void **lv_draw_sdl_line**(lv_draw_unit_t *draw_unit, const lv_draw_line_dsc_t *dsc)

```
void lv_draw_sdl_layer(lv_draw_unit_t *draw_unit, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t
                       *coords)

void lv_draw_sdl_triangle(lv_draw_unit_t *draw_unit, const lv_draw_triangle_dsc_t *dsc)

void lv_draw_sdl_mask_rect(lv_draw_unit_t *draw_unit, const lv_draw_mask_rect_dsc_t *dsc, const lv_area_t
                           *coords)
```

14.15.32 snapshot

lv_snapshot.h

Funzioni

`lv_draw_buf_t *lv_snapshot_take(lv_obj_t *obj, lv_color_format_t cf)`

Take snapshot for object with its children, create the draw buffer as needed.

Parametri

- **obj** -- the object to generate snapshot.
- **cf** -- color format for generated image.

Ritorna

a pointer to an draw buffer containing snapshot image, or NULL if failed.

`lv_draw_buf_t *lv_snapshot_create_draw_buf(lv_obj_t *obj, lv_color_format_t cf)`

Create a draw buffer to store the snapshot image for object.

Parametri

- **obj** -- the object to generate snapshot.
- **cf** -- color format for generated image.

Ritorna

a pointer to an draw buffer ready for taking snapshot, or NULL if failed.

`lv_result_t lv_snapshot_reshape_draw_buf(lv_obj_t *obj, lv_draw_buf_t *draw_buf)`

Reshape the draw buffer to prepare for taking snapshot for obj. This is usually used to check if the existing draw buffer is enough for obj snapshot. If return LV_RESULT_INVALID, you should create a new one.

Parametri

- **draw_buf** -- the draw buffer to reshape.
- **obj** -- the object to generate snapshot.

`lv_result_t lv_snapshot_take_to_draw_buf(lv_obj_t *obj, lv_color_format_t cf, lv_draw_buf_t *draw_buf)`

Take snapshot for object with its children, save image info to provided buffer.

Parametri

- **obj** -- the object to generate snapshot.
- **cf** -- color format for new snapshot image. It could differ with cf of **draw_buf** as long as the new cf will fit in.
- **draw_buf** -- the draw buffer to store the image result. It's reshaped automatically.

Ritorna

LV_RESULT_OK on success, LV_RESULT_INVALID on error.

`void lv_snapshot_free(lv_image_dsc_t *dsc)`

Deprecated:

Use `lv_draw_buf_destroy` instead.

Free the snapshot image returned by `lv_snapshot_take`

Parametri

- **dsc** -- the image descriptor generated by `lv_snapshot_take`.

`lv_result_t lv_snapshot_take_to_buf(lv_obj_t *obj, lv_color_format_t cf, lv_image_dsc_t *dsc, void *buf,
 uint32_t buf_size)`

Take snapshot for object with its children, save image info to provided buffer.

Deprecated:

Use lv_snapshot_take_to_draw_buf instead.

Parametri

- **obj** -- the object to generate snapshot.
- **cf** -- color format for generated image.
- **dsc** -- image descriptor to store the image result.
- **buf** -- the buffer to store image data. It must meet align requirement.
- **buf_size** -- provided buffer size in bytes.

Ritorna

LV_RESULT_OK on success, LV_RESULT_INVALID on error.

14.15.33 sw

lv_draw_sw.h

Funzioni

void **lv_draw_sw_init**(void)

Initialize the SW renderer. Called in internally. It creates as many SW renderers as defined in LV_DRAW_SW_DRAW_UNIT_CNT

void **lv_draw_sw_deinit**(void)

Deinitialize the SW renderers

void **lv_draw_sw_fill**(lv_draw_task_t *t, *lv_draw_fill_dsc_t* *dsc, const *lv_area_t* *coords)

Fill an area using SW render. Handle gradient and radius.

Parametri

- **t** -- pointer to a draw task
- **dsc** -- the draw descriptor
- **coords** -- the coordinates of the rectangle

void **lv_draw_sw_border**(lv_draw_task_t *t, const *lv_draw_border_dsc_t* *dsc, const *lv_area_t* *coords)

Draw border with SW render.

Parametri

- **t** -- pointer to a draw task
- **dsc** -- the draw descriptor
- **coords** -- the coordinates of the rectangle

void **lv_draw_sw_box_shadow**(lv_draw_task_t *t, const *lv_draw_box_shadow_dsc_t* *dsc, const *lv_area_t* *coords)

Draw box shadow with SW render.

Parametri

- **t** -- pointer to a draw task
- **dsc** -- the draw descriptor
- **coords** -- the coordinates of the rectangle for which the box shadow should be drawn

void **lv_draw_sw_image**(lv_draw_task_t *t, const *lv_draw_image_dsc_t* *draw_dsc, const *lv_area_t* *coords)

Draw an image with SW render. It handles image decoding, tiling, transformations, and recoloring.

Parametri

- **t** -- pointer to a draw task
- **draw_dsc** -- the draw descriptor
- **coords** -- the coordinates of the image

void **lv_draw_sw_letter**(lv_draw_task_t *t, const *lv_draw_letter_dsc_t* *dsc, const *lv_area_t* *coords)

void **lv_draw_sw_label**(lv_draw_task_t *t, const *lv_draw_label_dsc_t* *dsc, const *lv_area_t* *coords)

Draw a label with SW render.

Parametri

- **t** -- pointer to a draw task
- **dsc** -- the draw descriptor

- **coords** -- the coordinates of the label

`void lv_draw_sw_arc(lv_draw_task_t *t, const lv_draw_arc_dsc_t *dsc, const lv_area_t *coords)`

Draw an arc with SW render.

Parametri

- **t** -- pointer to a draw task
- **dsc** -- the draw descriptor
- **coords** -- the coordinates of the arc

`void lv_draw_sw_line(lv_draw_task_t *t, const lv_draw_line_dsc_t *dsc)`

Draw a line with SW render.

Parametri

- **t** -- pointer to a draw task
- **dsc** -- the draw descriptor

`void lv_draw_sw_layer(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t *coords)`

Blend a layer with SW render

Parametri

- **t** -- pointer to a draw task
- **draw_dsc** -- the draw descriptor
- **coords** -- the coordinates of the layer

`void lv_draw_sw_triangle(lv_draw_task_t *t, const lv_draw_triangle_dsc_t *dsc)`

Draw a triangle with SW render.

Parametri

- **t** -- pointer to a draw task
- **dsc** -- the draw descriptor

`void lv_draw_sw_mask_rect(lv_draw_task_t *t, const lv_draw_mask_rect_dsc_t *dsc)`

Mask out a rectangle with radius from a current layer

Parametri

- **t** -- pointer to a draw task
- **dsc** -- the draw descriptor
- **coords** -- the coordinates of the mask

`void lv_draw_sw_transform(const lv_area_t *dest_area, const void *src_buf, int32_t src_w, int32_t src_h, int32_t src_stride, const lv_draw_image_dsc_t *draw_dsc, const lv_draw_image_sup_t *sup, lv_color_format_t cf, void *dest_buf)`

Used internally to get a transformed area of an image

Parametri

- **dest_area** -- area to calculate, i.e. get this area from the transformed image
- **src_buf** -- source buffer
- **src_w** -- source buffer width in pixels
- **src_h** -- source buffer height in pixels
- **src_stride** -- source buffer stride in bytes
- **draw_dsc** -- draw descriptor
- **sup** -- supplementary data
- **cf** -- color format of the source buffer
- **dest_buf** -- the destination buffer

`void lv_draw_sw_vector(lv_draw_task_t *t, lv_draw_vector_dsc_t *dsc)`

Draw vector graphics with SW render.

Parametri

- **t** -- pointer to a draw task
- **dsc** -- the draw descriptor

`bool lv_draw_sw_register_blend_handler(lv_draw_sw_custom_blend_handler_t *handler)`

Register a custom blend handler for a color format. Handler will be called when blending a color or an image to a buffer with the given color format. At most one handler can be registered for a color format. Subsequent registrations will overwrite the previous handler.

Parametri

handler -- pointer to a blend handler

Ritorna

true if the handler was registered, false if the handler could not be registered

```
bool lv_draw_sw_unregister_blend_handler(lv_color_format_t dest_cf)
```

Unregister a custom blend handler for a color format.

Parametri

dest_cf -- color format

Ritorna

true if a handler was unregistered, false if no handler was registered

```
lv_draw_sw_blend_handler_t lv_draw_sw_get_blend_handler(lv_color_format_t dest_cf)
```

Get the blend handler for a color format.

Parametri

dest_cf -- color format

Ritorna

pointer to the blend handler or NULL if no handler is registered

lv_draw_sw_grad.h**Funzioni**

```
void lv_draw_sw_grad_color_calculate(const lv_grad_dsc_t *dsc, int32_t range, int32_t frac, lv_color_t *color_out, lv_opa_t *opa_out)
```

Compute the color in the given gradient and fraction Gradient are specified in a virtual [0-255] range, so this function scales the virtual range to the given range

Parametri

- **dsc** -- The gradient descriptor to use
- **range** -- The range to use in computation.
- **frac** -- The current part used in the range. frac is in [0; range]
- **color_out** -- Calculated gradient color
- **opa_out** -- Calculated opacity

```
lv_draw_sw_grad_calc_t *lv_draw_sw_grad_get(const lv_grad_dsc_t *gradient, int32_t w, int32_t h)
```

Get a gradient cache from the given parameters

```
void lv_draw_sw_grad_cleanup(lv_draw_sw_grad_calc_t *grad)
```

Clean up the gradient item after it was get with lv_grad_get_from_cache.

Parametri

grad -- pointer to a gradient

```
void lv_draw_sw_grad_linear_setup(lv_grad_dsc_t *dsc, const lv_area_t *coords)
```

Calculate constants from the given parameters that are used during rendering

Parametri

- **dsc** -- gradient descriptor
- **coords** -- the area where to draw the gradient

```
void lv_draw_sw_grad_linear_cleanup(lv_grad_dsc_t *dsc)
```

Free up the allocated memory for the gradient calculation

Parametri

dsc -- gradient descriptor

```
void lv_draw_sw_grad_linear_get_line(lv_grad_dsc_t *dsc, int32_t xp, int32_t yp, int32_t width,
                                     lv_draw_sw_grad_calc_t *result)
```

Calculate a line segment of a linear gradient

Parametri

- **dsc** -- gradient descriptor
- **xp** -- starting point x coordinate in gradient space
- **yp** -- starting point y coordinate in gradient space

- **width** -- width of the line segment in pixels
- **result** -- color buffer for the resulting line segment

`void lv_draw_sw_grad_radial_setup(lv_grad_dsc_t *dsc, const lv_area_t *coords)`

Calculate constants from the given parameters that are used during rendering

Parametri

- **dsc** -- gradient descriptor
- **coords** -- the area where to draw the gradient

`void lv_draw_sw_grad_radial_cleanup(lv_grad_dsc_t *dsc)`

Free up the allocated memory for the gradient calculation

Parametri

dsc -- gradient descriptor

`void lv_draw_sw_grad_radial_get_line(lv_grad_dsc_t *dsc, int32_t xp, int32_t yp, int32_t width, lv_draw_sw_grad_calc_t *result)`

Calculate a line segment of a radial gradient

Parametri

- **dsc** -- gradient descriptor
- **xp** -- starting point x coordinate in gradient space
- **yp** -- starting point y coordinate in gradient space
- **width** -- width of the line segment in pixels
- **result** -- color buffer for the resulting line segment

`void lv_draw_sw_grad_conical_setup(lv_grad_dsc_t *dsc, const lv_area_t *coords)`

Calculate constants from the given parameters that are used during rendering

Parametri

- **dsc** -- gradient descriptor
- **coords** -- the area where to draw the gradient

`void lv_draw_sw_grad_conical_cleanup(lv_grad_dsc_t *dsc)`

Free up the allocated memory for the gradient calculation

Parametri

dsc -- gradient descriptor

`void lv_draw_sw_grad_conical_get_line(lv_grad_dsc_t *dsc, int32_t xp, int32_t yp, int32_t width, lv_draw_sw_grad_calc_t *result)`

Calculate a line segment of a conical gradient

Parametri

- **dsc** -- gradient descriptor
- **xp** -- starting point x coordinate in gradient space
- **yp** -- starting point y coordinate in gradient space
- **width** -- width of the line segment in pixels
- **result** -- color buffer for the resulting line segment

`struct lv_draw_sw_grad_calc_t`

Membri Pubblici

`lv_color_t *color_map`

`lv_opa_t *opa_map`

`uint32_t size`

lv_draw_sw_mask.h**Define****LV_MASK_ID_INV****LV_MASK_MAX_NUM****Typedef**

```
typedef lv_draw_sw_mask_res_t (*lv_draw_sw_mask_xcb_t)(lv_opa_t *mask_buf, int32_t abs_x, int32_t abs_y, int32_t len, void *p)
```

A common callback type for every mask type. Used internally by the library.

Enum**enum lv_draw_sw_mask_res_t**

Values:

enumerator **LV_DRAW_SW_MASK_RES_TRANSPIRANT**

enumerator **LV_DRAW_SW_MASK_RES_FULL_COVER**

enumerator **LV_DRAW_SW_MASK_RES_CHANGED**

enumerator **LV_DRAW_SW_MASK_RES_UNKNOWN**

enum lv_draw_sw_mask_type_t

Values:

enumerator **LV_DRAW_SW_MASK_TYPE_LINE**

enumerator **LV_DRAW_SW_MASK_TYPE_ANGLE**

enumerator **LV_DRAW_SW_MASK_TYPE_RADIUS**

enumerator **LV_DRAW_SW_MASK_TYPE_FADE**

enumerator **LV_DRAW_SW_MASK_TYPE_MAP**

enum lv_draw_sw_mask_line_side_t

Values:

enumerator **LV_DRAW_SW_MASK_LINE_SIDE_LEFT**

enumerator **LV_DRAW_SW_MASK_LINE_SIDE_RIGHT**

enumerator **LV_DRAW_SW_MASK_LINE_SIDE_TOP**

enumerator **LV_DRAW_SW_MASK_LINE_SIDE_BOTTOM**

Funzioni

void lv_draw_sw_mask_init(void)

void lv_draw_sw_mask_deinit(void)

lv_draw_sw_mask_res_t lv_draw_sw_mask_apply(void *masks[], lv_opa_t *mask_buf, int32_t abs_x, int32_t abs_y, int32_t len)

Apply the added buffers on a line. Used internally by the library's drawing routines.

Parametri

- **masks** -- the masks list to apply, must be ended with NULL pointer in array.
- **mask_buf** -- store the result mask here. Has to be **len** byte long. Should be initialized with 0xFF.
- **abs_x** -- absolute X coordinate where the line to calculate start
- **abs_y** -- absolute Y coordinate where the line to calculate start
- **len** -- length of the line to calculate (in pixel count)

Ritorna

One of these values:

- **LV_DRAW_MASK_RES_FULL_TRANSP**: the whole line is transparent. **mask_buf** is not set to zero
- **LV_DRAW_MASK_RES_FULL_COVER**: the whole line is fully visible. **mask_buf** is unchanged
- **LV_DRAW_MASK_RES_CHANGED**: **mask_buf** has changed, it shows the desired opacity of each pixel in the given line

void lv_draw_sw_mask_free_param(void *p)

Free the data from the parameter. It's called inside **lv_draw_sw_mask_remove_id** and **lv_draw_sw_mask_remove_custom**. Needs to be called only in special cases when the mask is not added by **lv_draw_mask_add** and not removed by **lv_draw_mask_remove_id** or **lv_draw_mask_remove_custom**.

Parametri

p -- pointer to a mask parameter

void lv_draw_sw_mask_line_points_init(lv_draw_sw_mask_line_param_t *param, int32_t p1x, int32_t p1y, int32_t p2x, int32_t p2y, lv_draw_sw_mask_line_side_t side)

Initialize a line mask from two points.

Parametri

- **param** -- pointer to a **lv_draw_mask_param_t** to initialize
- **p1x** -- X coordinate of the first point of the line
- **p1y** -- Y coordinate of the first point of the line
- **p2x** -- X coordinate of the second point of the line
- **p2y** -- y coordinate of the second point of the line
- **side** -- an element of **lv_draw_mask_line_side_t** to describe which side to keep.
With **LV_DRAW_MASK_LINE_SIDE_LEFT/RIGHT** and horizontal line all pixels are kept
With **LV_DRAW_MASK_LINE_SIDE_TOP/BOTTOM** and vertical line all pixels are kept

void lv_draw_sw_mask_line_angle_init(lv_draw_sw_mask_line_param_t *param, int32_t px, int32_t py, int16_t angle, lv_draw_sw_mask_line_side_t side)

Initialize a line mask from a point and an angle.

Parametri

- **param** -- pointer to a **lv_draw_mask_param_t** to initialize
- **px** -- X coordinate of a point of the line
- **py** -- Y coordinate of a point of the line
- **angle** -- right 0 deg, bottom: 90
- **side** -- an element of **lv_draw_mask_line_side_t** to describe which side to keep.
With **LV_DRAW_MASK_LINE_SIDE_LEFT/RIGHT** and horizontal line all pixels are kept
With **LV_DRAW_MASK_LINE_SIDE_TOP/BOTTOM** and vertical line all pixels are kept

```
void lv_draw_sw_mask_angle_init(lv_draw_sw_mask_angle_param_t *param, int32_t vertex_x, int32_t
                                vertex_y, int32_t start_angle, int32_t end_angle)
```

Initialize an angle mask.

Parametri

- **param** -- pointer to a `lv_draw_mask_param_t` to initialize
- **vertex_x** -- X coordinate of the angle vertex (absolute coordinates)
- **vertex_y** -- Y coordinate of the angle vertex (absolute coordinates)
- **start_angle** -- start angle in degrees. 0 deg on the right, 90 deg, on the bottom
- **end_angle** -- end angle

```
void lv_draw_sw_mask_radius_init(lv_draw_sw_mask_radius_param_t *param, const lv_area_t *rect, int32_t
                                 radius, bool inv)
```

Initialize a fade mask.

Parametri

- **param** -- pointer to an `lv_draw_mask_radius_param_t` to initialize
- **rect** -- coordinates of the rectangle to affect (absolute coordinates)
- **radius** -- radius of the rectangle
- **inv** -- true: keep the pixels inside the rectangle; keep the pixels outside of the rectangle

```
void lv_draw_sw_mask_fade_init(lv_draw_sw_mask_fade_param_t *param, const lv_area_t *coords, lv_opa_t
                               opa_top, int32_t y_top, lv_opa_t opa_bottom, int32_t y_bottom)
```

Initialize a fade mask.

Parametri

- **param** -- pointer to a `lv_draw_mask_param_t` to initialize
- **coords** -- coordinates of the area to affect (absolute coordinates)
- **opa_top** -- opacity on the top
- **y_top** -- at which coordinate start to change to opacity to **opa_bottom**
- **opa_bottom** -- opacity at the bottom
- **y_bottom** -- at which coordinate reach **opa_bottom**.

```
void lv_draw_sw_mask_map_init(lv_draw_sw_mask_map_param_t *param, const lv_area_t *coords, const
                               lv_opa_t *map)
```

Initialize a map mask.

Parametri

- **param** -- pointer to a `lv_draw_mask_param_t` to initialize
- **coords** -- coordinates of the map (absolute coordinates)
- **map** -- array of bytes with the mask values

lv_draw_sw_mask_private.h

Typedef

```
typedef lv_draw_sw_mask_radius_circle_dsc_t lv_draw_sw_mask_radius_circle_dsc_arr_t[4]
```

Funzioni

```
void lv_draw_sw_mask_cleanup(void)
```

Called by LVGL the rendering of a screen is ready to clean up the temporal (cache) data of the masks

```
struct lv_draw_sw_mask_radius_circle_dsc_t
```

Membri Pubblici

```
uint8_t *buf
```

`lv_opa_t *cir_opa`

Opacity of values on the circumference of an 1/4 circle

`uint16_t *x_start_on_y`

The x coordinate of the circle for each y value

`uint16_t *opa_start_on_y`

The index of `cir_opa` for each y value

`int32_t life`

How many times the entry was used

`uint32_t used_cnt`

Like a semaphore to count the referencing masks

`int32_t radius`

The radius of the entry

struct `_lv_draw_sw_mask_common_dsc_t`

Membri Pubblici

`lv_draw_sw_mask_xcb_t cb`

`lv_draw_sw_mask_type_t type`

struct `_lv_draw_sw_mask_line_param_t`

Membri Pubblici

`lv_draw_sw_mask_common_dsc_t dsc`

The first element must be the common descriptor

`lv_point_t p1`

`lv_point_t p2`

`lv_draw_sw_mask_line_side_t side`

struct `_lv_draw_sw_mask_line_param_t cfg`

`lv_point_t origo`

A point of the line

`int32_t xy_stEEP`

X / (1024*Y) steepness (X is 0..1023 range). What is the change of X in 1024 Y?

int32_t yx_stEEP

$Y / (1024 * X)$ steepness (Y is 0..1023 range). What is the change of Y in 1024 X?

int32_t steep

Helper which stores yx_stEEP for flat lines and xy_stEEP for steep (non flat) lines

int32_t spx

Steepness in 1 px in 0..255 range. Used only by flat lines.

uint8_t flat

1: It's a flat line? (Near to horizontal)

uint8_t inv

Invert the mask. The default is: Keep the left part. It is used to select left/right/top/bottom

```
struct _lv_draw_sw_mask_angle_param_t
```

Membri Pubblici**lv_draw_sw_mask_common_dsc_t dsc**

The first element must be the common descriptor

lv_point_t* vertex_p*int32_t start_angle****int32_t end_angle****struct _lv_draw_sw_mask_angle_param_t cfg****lv_draw_sw_mask_line_param_t start_line****lv_draw_sw_mask_line_param_t end_line****uint16_t delta_deg**

```
struct _lv_draw_sw_mask_radius_param_t
```

Membri Pubblici**lv_draw_sw_mask_common_dsc_t dsc**

The first element must be the common descriptor

lv_area_t* rect*int32_t radius**

```
uint8_t outer
```

Invert the mask. 0: Keep the pixels inside.

```
struct _lv_draw_sw_mask_radius_param_t cfg
```

```
lv_draw_sw_mask_radius_circle_dsc_t *circle
```

```
struct _lv_draw_sw_mask_fade_param_t
```

Membri Pubblici

```
lv_draw_sw_mask_common_dsc_t dsc
```

The first element must be the common descriptor

```
lv_area_t coords
```

```
int32_t y_top
```

```
int32_t y_bottom
```

```
lv_opa_t opa_top
```

```
lv_opa_t opa_bottom
```

```
struct _lv_draw_sw_mask_fade_param_t cfg
```

```
struct _lv_draw_sw_mask_map_param_t
```

Membri Pubblici

```
lv_draw_sw_mask_common_dsc_t dsc
```

The first element must be the common descriptor

```
lv_area_t coords
```

```
const lv_opa_t *map
```

```
struct _lv_draw_sw_mask_map_param_t cfg
```

lv_draw_sw_private.h

```
struct lv_draw_sw_thread_dsc_t
```

Membri Pubblici

```
lv_draw_task_t *task_act
```

```
lv_thread_t thread
```

```
lv_thread_sync_t sync
```

```
lv_draw_unit_t *draw_unit
```

```
uint32_t idx
```

```
volatile bool inited
```

```
volatile bool exit_status
```

```
struct _lv_draw_sw_unit_t
```

Membri Pubblici

```
lv_draw_unit_t base_unit
```

```
lv_draw_task_t *task_act
```

lv_draw_sw_utils.h

Funzioni

```
void lv_draw_sw_i1_to_argb8888(const void *buf_i1, void *buf_argb8888, uint32_t width, uint32_t height,
                                 uint32_t buf_i1_stride, uint32_t buf_argb8888_stride, uint32_t index0_color,
                                 uint32_t index1_color)
```

Converts an I1 buffer to ARGB8888 format.

Parametri

- **buf_i1** -- pointer to buffer with I1 formatted render
- **buf_argb8888** -- pointer to buffer for ARGB8888 render
- **width** -- width in pixels of the area. must be a multiple of 8.
- **height** -- height in pixels of the area
- **buf_i1_stride** -- stride of i1 buffer in bytes
- **buf_argb8888_stride** -- stride of argb8888 buffer in bytes
- **index0_color** -- color of the 0 bits of i1 buf
- **index1_color** -- color of the 1 bits of i1 buf

```
void lv_draw_sw_rgb565_swap(void *buf, uint32_t buf_size_px)
```

Swap the upper and lower byte of an RGB565 buffer. Might be required if a 8bit parallel port or an SPI port send the bytes in the wrong order. The bytes will be swapped in place.

Parametri

- **buf** -- pointer to buffer
- **buf_size_px** -- number of pixels in the buffer

```
void lv_draw_sw_i1_invert(void *buf, uint32_t buf_size)
```

Invert a draw buffer in the I1 color format. Conventionally, a bit is set to 1 during blending if the luminance is greater than 127. Depending on the display controller used, you might want to have different behavior. The inversion will be performed in place.

Parametri

- **buf** -- pointer to the buffer to be inverted
- **buf_size** -- size of the buffer in bytes

```
void lv_draw_sw_i1_convert_to_vtiled(const void *buf, uint32_t buf_size, uint32_t width, uint32_t height,
                                     void *out_buf, uint32_t out_buf_size, bool bit_order_lsb)
```

Convert a draw buffer in I1 color format from htiled (row-wise) to vtiled (column-wise) buffer layout. The conversion assumes that the buffer width and height is rounded to a multiple of 8.

Parametri

- **buf** -- pointer to the buffer to be converted
- **buf_size** -- size of the buffer in bytes
- **width** -- width of the buffer
- **height** -- height of the buffer
- **out_buf** -- pointer to the output buffer
- **out_buf_size** -- size of the output buffer in bytes
- **bit_order_lsb** -- bit order of the resulting vtiled buffer

```
void lv_draw_sw_rotate(const void *src, void *dest, int32_t src_width, int32_t src_height, int32_t src_stride,
                      int32_t dest_stride, lv_display_rotation_t rotation, lv_color_format_t color_format)
```

Rotate a buffer into another buffer

Parametri

- **src** -- the source buffer
- **dest** -- the destination buffer
- **src_width** -- source width in pixels
- **src_height** -- source height in pixels
- **src_stride** -- source stride in bytes (number of bytes in a row)
- **dest_stride** -- destination stride in bytes (number of bytes in a row)
- **rotation** -- LV_DISPLAY_ROTATION_0/90/180/270
- **color_format** -- LV_COLOR_FORMAT_RGB565/RGB888/XRGB8888/ARGB8888

arm2d

[lv_draw_sw_arm2d.h](#)

[lv_draw_sw_helium.h](#)

blend

[lv_draw_sw_blend.h](#)

Typedef

```
typedef void (*lv_draw_sw_blend_handler_t)(lv_draw_task_t *t, const lv_draw_sw_blend_dsc_t *dsc)
```

Custom draw function for SW rendering.

Param t

pointer to a draw task

Param dsc

pointer to an initialized blend descriptor

Funzioni

```
void lv_draw_sw_blend(lv_draw_task_t *t, const lv_draw_sw_blend_dsc_t *dsc)
```

Call the blend function of the layer.

Parametri

- **draw_unit** -- pointer to a draw unit
- **dsc** -- pointer to an initialized blend descriptor

```
struct lv_draw_sw_custom_blend_handler_t
```

Membri Pubblici

lv_color_format_t dest_cf

lv_draw_sw_blend_handler_t handler

lv_draw_sw_blend_private.h

struct _lv_draw_sw_blend_dsc_t

Membri Pubblici

const *lv_area_t* *blend_area

The area with absolute coordinates to draw on layer->buf will be clipped to layer->clip_area

const void *src_buf

Pointer to an image to blend. If set fill_color is ignored

uint32_t src_stride

lv_color_format_t src_color_format

const *lv_area_t* *src_area

lv_opa_t opa

The overall opacity

lv_color_t color

Fill color

const *lv_opa_t* *mask_buf

NULL if ignored, or an alpha mask to apply on blend_area

lv_draw_sw_mask_res_t mask_res

The result of the previous mask operation

const *lv_area_t* *mask_area

The area of mask_buf with absolute coordinates

int32_t mask_stride

lv_blend_mode_t blend_mode

E.g. LV_BLEND_MODE_ADDITIVE

struct _lv_draw_sw_blend_fill_dsc_t

Membri Pubblici

```
void *dest_buf  
  
int32_t dest_w  
  
int32_t dest_h  
  
int32_t dest_stride  
  
const lv_opa_t *mask_buf  
  
int32_t mask_stride  
  
lv_color_t color  
  
lv_opa_t opa  
  
lv_area_t relative_area  
  
struct _lv_draw_sw_blend_image_dsc_t
```

Membri Pubblici

```
void *dest_buf  
  
int32_t dest_w  
  
int32_t dest_h  
  
int32_t dest_stride  
  
const lv_opa_t *mask_buf  
  
int32_t mask_stride  
  
const void *src_buf  
  
int32_t src_stride  
  
lv_color_format_t src_color_format  
  
lv_opa_t opa  
  
lv_blend_mode_t blend_mode
```

lv_area_t relative_area

The blend area relative to the layer's buffer area.

lv_area_t src_area

The original src area.

lv_draw_sw_blend_to_a188.h**Funzioni**

```
void lv_draw_sw_blend_color_to_a188(lv_draw_sw_blend_fill_dsc_t *dsc)
```

```
void lv_draw_sw_blend_image_to_a188(lv_draw_sw_blend_image_dsc_t *dsc)
```

lv_draw_sw_blend_to_argb8888.h**Funzioni**

```
void lv_draw_sw_blend_color_to_argb8888(lv_draw_sw_blend_fill_dsc_t *dsc)
```

```
void lv_draw_sw_blend_image_to_argb8888(lv_draw_sw_blend_image_dsc_t *dsc)
```

lv_draw_sw_blend_to_argb8888_premultiplied.h**Funzioni**

```
void lv_draw_sw_blend_color_to_argb8888_premultiplied(lv_draw_sw_blend_fill_dsc_t *dsc)
```

```
void lv_draw_sw_blend_image_to_argb8888_premultiplied(lv_draw_sw_blend_image_dsc_t *dsc)
```

lv_draw_sw_blend_to_i1.h**Funzioni**

```
void lv_draw_sw_blend_color_to_i1(lv_draw_sw_blend_fill_dsc_t *dsc)
```

```
void lv_draw_sw_blend_image_to_i1(lv_draw_sw_blend_image_dsc_t *dsc)
```

lv_draw_sw_blend_to_l8.h**Funzioni**

```
void lv_draw_sw_blend_color_to_l8(lv_draw_sw_blend_fill_dsc_t *dsc)
```

```
void lv_draw_sw_blend_image_to_l8(lv_draw_sw_blend_image_dsc_t *dsc)
```

lv_draw_sw_blend_to_rgb565.h**Funzioni**

```
void lv_draw_sw_blend_color_to_rgb565(lv_draw_sw_blend_fill_dsc_t *dsc)
```

```
void lv_draw_sw_blend_image_to_rgb565(lv_draw_sw_blend_image_dsc_t *dsc)
```

lv_draw_sw_blend_to_rgb565_swapped.h**Funzioni**

```
void lv_draw_sw_blend_color_to_rgb565_swapped(lv_draw_sw_blend_fill_dsc_t *dsc)
```

```
void lv_draw_sw_blend_image_to_rgb565_swapped(lv_draw_sw_blend_image_dsc_t *dsc)
```

lv_draw_sw_blend_to_rgb888.h**Funzioni**

```
void lv_draw_sw_blend_color_to_rgb888(lv_draw_sw_blend_fill_dsc_t *dsc, uint32_t dest_px_size)
void lv_draw_sw_blend_image_to_rgb888(lv_draw_sw_blend_image_dsc_t *dsc, uint32_t dest_px_size)
```

arm2d**lv_blend_arm2d.h****helium****lv_blend_helium.h****neon****lv_blend_neon.h****lv_draw_sw_blend_neon_to_rgb565.h****lv_draw_sw_blend_neon_to_rgb888.h****14.15.34 vg_lite****lv_draw_vg_lite.h****Funzioni**

```
void lv_draw_buf_vg_lite_init_handlers(void)
void lv_draw_vg_lite_init(void)
void lv_draw_vg_lite_deinit(void)
void lv_draw_vg_lite_arc(lv_draw_task_t *t, const lv_draw_arc_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_vg_lite_box_shadow(lv_draw_task_t *t, const lv_draw_box_shadow_dsc_t *dsc, const lv_area_t
                               *coords)
void lv_draw_vg_lite_border(lv_draw_task_t *t, const lv_draw_border_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_vg_lite_fill(lv_draw_task_t *t, const lv_draw_fill_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_vg_lite_img(lv_draw_task_t *t, const lv_draw_image_dsc_t *dsc, const lv_area_t *coords, bool
                        no_cache)
void lv_draw_vg_lite_label_init(struct _lv_draw_vg_lite_unit_t *u)
void lv_draw_vg_lite_label_deinit(struct _lv_draw_vg_lite_unit_t *u)
void lv_draw_vg_lite_letter(lv_draw_task_t *t, const lv_draw_letter_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_vg_lite_label(lv_draw_task_t *t, const lv_draw_label_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_vg_lite_layer(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t
                           *coords)
void lv_draw_vg_lite_line(lv_draw_task_t *t, const lv_draw_line_dsc_t *dsc)
void lv_draw_vg_lite_triangle(lv_draw_task_t *t, const lv_draw_triangle_dsc_t *dsc)
```

```
void lv_draw_vg_lite_mask_rect(lv_draw_task_t *t, const lv_draw_mask_rect_dsc_t *dsc, const lv_area_t
                               *coords)
```

```
void lv_draw_vg_lite_vector(lv_draw_task_t *t, const lv_draw_vector_dsc_t *dsc)
```

lv_draw_vg_lite_type.h

```
struct _lv_draw_vg_lite_unit_t
```

Membri Pubblici

```
lv_draw_unit_t base_unit
```

```
lv_draw_task_t *task_act
```

```
lv_area_t current_scissor_area
```

```
struct _lv_vg_lite_pending_t *image_dsc_pending
```

```
struct _lv_vg_lite_grad_ctx_t *grad_ctx
```

```
lv_cache_t *stroke_cache
```

```
struct _lv_vg_lite_pending_t *bitmap_font_pending
```

```
uint16_t flush_count
```

```
uint16_t letter_count
```

```
vg_lite_buffer_t target_buffer
```

```
vg_lite_matrix_t global_matrix
```

```
struct _lv_vg_lite_path_t *global_path
```

```
bool path_in_use
```

lv_vg_lite_decoder.h

Funzioni

```
void lv_vg_lite_decoder_init(void)
```

```
void lv_vg_lite_decoder_deinit(void)
```

lv_vg_lite_grad.h

Funzioni

```
struct _lv_vg_lite_grad_ctx_t *lv_vg_lite_grad_ctx_create(uint32_t cache_cnt, struct _lv_draw_vg_lite_unit_t
*unit)
```

Create a gradient context.

Parametri

- **cache_cnt** -- number of cache entries
- **unit** -- the draw unit

```
void lv_vg_lite_grad_ctx_delete(struct _lv_vg_lite_grad_ctx_t *ctx)
```

Delete a gradient context.

Parametri

ctx -- the gradient context to delete

```
struct _lv_vg_lite_pending_t *lv_vg_lite_grad_ctx_get_pending(struct _lv_vg_lite_grad_ctx_t *ctx)
```

Get the pending list of gradient items.

Parametri

ctx -- the gradient context

```
struct _lv_cache_t *lv_vg_lite_grad_ctx_get_cache(struct _lv_vg_lite_grad_ctx_t *ctx)
```

Get the cache of gradient items.

Parametri

ctx -- the gradient context

```
bool lv_vg_lite_draw_grad(struct _lv_vg_lite_grad_ctx_t *ctx, vg_lite_buffer_t *buffer, vg_lite_path_t *path,
const lv_vector_gradient_t *grad, const vg_lite_matrix_t *grad_matrix, const
vg_lite_matrix_t *matrix, vg_lite_fill_t fill, vg_lite_blend_t blend)
```

Draw a gradient.

Parametri

- **ctx** -- the gradient context
- **buffer** -- the target buffer
- **path** -- the path to draw the gradient on
- **grad** -- the gradient descriptor
- **grad_matrix** -- the gradient matrix
- **matrix** -- the matrix to apply to the gradient
- **fill** -- the fill rule
- **blend** -- the blend mode

Ritorna

true: success, false: failed

```
bool lv_vg_lite_draw_grad_helper(struct _lv_vg_lite_grad_ctx_t *ctx, vg_lite_buffer_t *buffer, vg_lite_path_t
*path, const lv_area_t *area, const lv_grad_dsc_t *grad_dsc, const
vg_lite_matrix_t *matrix, vg_lite_fill_t fill, vg_lite_blend_t blend)
```

Draw a gradient helper.

Parametri

- **ctx** -- the gradient context
- **buffer** -- the target buffer
- **path** -- the path to draw the gradient on
- **area** -- the area to draw the gradient on
- **grad_dsc** -- the gradient descriptor
- **matrix** -- the matrix to apply to the gradient
- **fill** -- the fill rule
- **blend** -- the blend mode

Ritorna

true: success, false: failed

lv_vg_lite_math.h

Define

MATH_PI

MATH_HALF_PI**MATH_TWO_PI****DEG_TO_RAD****RAD_TO_DEG****MATH_TANF(x)****MATH_SINF(x)****MATH_COSF(x)****MATH_ASINF(x)****MATH_ACOSF(x)****MATH_FABSF(x)****MATH_SQRTF(x)****MATH_RADIANS(deg)****MATH_DEGREES(rad)**

Funzioni

static inline bool **math_zero**(float a)static inline bool **math_equal**(float a, float b)float **math_fast_inv_sqrtf**(float number)

lv_vg_lite_path.h

Define

LV_VG_LITE_PATH_SET_OP_CODE(PTR, TYPE, OP_CODE)**LV_VG_LITE_PATH_GET_OP_CODE(PTR)**

Typedef

typedef struct _lv_vg_lite_path_t **lv_vg_lite_path_t**typedef struct _lv_draw_vg_lite_unit_t **lv_draw_vg_lite_unit_t**typedef void (***lv_vg_lite_path_iter_cb_t**)(void *user_data, uint8_t op_code, const float *data, uint32_t len)

Funzioni

void **lv_vg_lite_path_init**(*lv_draw_vg_lite_unit_t* *unit)void **lv_vg_lite_path_deinit**(*lv_draw_vg_lite_unit_t* *unit)*lv_vg_lite_path_t* ***lv_vg_lite_path_create**(vg_lite_format_t data_format)

```

void lv_vg_lite_path_destroy(lv_vg_lite_path_t *path)
lv_vg_lite_path_t *lv_vg_lite_path_get(lv_draw_vg_lite_unit_t *unit, vg_lite_format_t data_format)
void lv_vg_lite_path_drop(lv_draw_vg_lite_unit_t *unit, lv_vg_lite_path_t *path)
void lv_vg_lite_path_reset(lv_vg_lite_path_t *path, vg_lite_format_t data_format)
void lv_vg_lite_path_set_bounding_box_area(lv_vg_lite_path_t *path, const lv_area_t *area)
void lv_vg_lite_path_set_bounding_box(lv_vg_lite_path_t *path, float min_x, float min_y, float max_x, float
max_y)
void lv_vg_lite_path_get_bounding_box(lv_vg_lite_path_t *path, float *min_x, float *min_y, float *max_x,
float *max_y)
bool lv_vg_lite_path_update_bounding_box(lv_vg_lite_path_t *path)
void lv_vg_lite_path_set_transform(lv_vg_lite_path_t *path, const vg_lite_matrix_t *matrix)
void lv_vg_lite_path_set_quality(lv_vg_lite_path_t *path, vg_lite_quality_t quality)
vg_lite_path_t *lv_vg_lite_path_get_path(lv_vg_lite_path_t *path)
void lv_vg_lite_path_reserve_space(lv_vg_lite_path_t *path, size_t len)
void lv_vg_lite_path_move_to(lv_vg_lite_path_t *path, float x, float y)
void lv_vg_lite_path_line_to(lv_vg_lite_path_t *path, float x, float y)
void lv_vg_lite_path_quad_to(lv_vg_lite_path_t *path, float cx, float cy, float x, float y)
void lv_vg_lite_path_cubic_to(lv_vg_lite_path_t *path, float cx1, float cy1, float cx2, float cy2, float x, float y)
void lv_vg_lite_path_close(lv_vg_lite_path_t *path)
void lv_vg_lite_path_end(lv_vg_lite_path_t *path)
void lv_vg_lite_path_append_rect(lv_vg_lite_path_t *path, float x, float y, float w, float h, float r)
void lv_vg_lite_path_append_circle(lv_vg_lite_path_t *path, float cx, float cy, float rx, float ry)
void lv_vg_lite_path_append_arc_right_angle(lv_vg_lite_path_t *path, float start_x, float start_y, float
center_x, float center_y, float end_x, float end_y)
void lv_vg_lite_path_append_arc(lv_vg_lite_path_t *path, float cx, float cy, float radius, float start_angle, float
sweep, bool pie)
void lv_vg_lite_path_append_path(lv_vg_lite_path_t *dest, const lv_vg_lite_path_t *src)
uint8_t lv_vg_lite_vlc_op_arg_len(uint8_t vlc_op)
uint8_t lv_vg_lite_path_format_len(vg_lite_format_t format)
void lv_vg_lite_path_for_each_data(const vg_lite_path_t *path, lv_vg_lite_path_iter_cb_t cb, void
*user_data)

```

lv_vg_lite_pending.h

Typedef

```
typedef struct _lv_vg_lite_pending_t lv_vg_lite_pending_t
```

```
typedef void (*lv_vg_lite_pending_free_cb_t)(void *obj, void *user_data)
```

Funzioni

`lv_vg_lite_pending_t *lv_vg_lite_pending_create(size_t obj_size, uint32_t capacity_default)`

Create a pending list

Parametri

- **obj_size** -- the size of the objects in the list
- **capacity_default** -- the default capacity of the list

Ritorna

a pointer to the pending list

`void lv_vg_lite_pending_destroy(lv_vg_lite_pending_t *pending)`

Destroy a pending list

Parametri

pending -- pointer to the pending list

`void lv_vg_lite_pending_set_free_cb(lv_vg_lite_pending_t *pending, lv_vg_lite_pending_free_cb_t free_cb, void *user_data)`

Set a free callback for the pending list

Parametri

- **pending** -- pointer to the pending list
- **free_cb** -- the free callback
- **user_data** -- user data to pass to the free callback

`void lv_vg_lite_pending_add(lv_vg_lite_pending_t *pending, void *obj)`

Add an object to the pending list

Parametri

- **pending** -- pointer to the pending list
- **obj** -- pointer to the object to add

`void lv_vg_lite_pending_remove_all(lv_vg_lite_pending_t *pending)`

Remove all objects from the active pending list

Parametri

pending -- pointer to the pending list

`void lv_vg_lite_pending_swap(lv_vg_lite_pending_t *pending)`

Remove all old objects reference and swap new objects reference

Parametri

pending -- pointer to the pending list

lv_vg_lite_stroke.h

Funzioni

`void lv_vg_lite_stroke_init(struct _lv_draw_vg_lite_unit_t *unit, uint32_t cache_cnt)`

Initialize the stroke module

Parametri

unit -- pointer to the unit

`void lv_vg_lite_stroke_deinit(struct _lv_draw_vg_lite_unit_t *unit)`

Deinitialize the stroke module

Parametri

unit -- pointer to the unit

`lv_cache_entry_t *lv_vg_lite_stroke_get(struct _lv_draw_vg_lite_unit_t *unit, struct _lv_vg_lite_path_t *path, const lv_vector_stroke_dsc_t *dsc)`

Get the stroke cache entry

Parametri

- **unit** -- pointer to the unit
- **path** -- pointer to the path
- **dsc** -- pointer to the stroke descriptor

Ritorna

pointer to the stroke cache entry

```
struct _lv_vg_lite_path_t *lv_vg_lite_stroke_get_path(lv_cache_entry_t *cache_entry)
```

Get the path of a stroke

Parametri

cache_entry -- pointer to the stroke cache entry

Ritorna

pointer to the path

```
void lv_vg_lite_stroke_drop(struct _lv_draw_vg_lite_unit_t *unit, lv_cache_entry_t *cache_entry)
```

Drop the stroke cache entry

Parametri

- **unit** -- pointer to the unit
- **stroke** -- pointer to the stroke

lv_vg_lite_utils.h**Define**

```
LV_VG_LITE_ASSERT(expr)
```

```
LV_VG_LITE_CHECK_ERROR(expr, dump_param)
```

```
LV_VG_LITE_ASSERT_PATH(path)
```

```
LV_VG_LITE_ASSERT_SRC_BUFFER(buffer)
```

```
LV_VG_LITE_ASSERT_DEST_BUFFER(buffer)
```

```
LV_VG_LITE_ASSERT_MATRIX(matrix)
```

```
LV_VG_LITE_ALIGN(number, align_bytes)
```

```
LV_VG_LITE_IS_ALIGNED(num, align)
```

```
LV_VG_LITE_IS_INDEX_FMT(fmt)
```

Funzioni

```
void lv_vg_lite_dump_info(void)
```

```
void lv_vg_lite_error_dump_info(vg_lite_error_t error)
```

```
const char *lv_vg_lite_error_string(vg_lite_error_t error)
```

```
const char *lv_vg_lite_feature_string(vg_lite_feature_t feature)
```

```
const char *lv_vg_lite_buffer_format_string(vg_lite_buffer_format_t format)
```

```
const char *lv_vg_lite_vlc_op_string(uint8_t vlc_op)
```

```
void lv_vg_lite_path_dump_info(const vg_lite_path_t *path)
```

```
void lv_vg_lite_stroke_dump_info(const vg_lite_stroke_t *stroke)
```

```
void lv_vg_lite_buffer_dump_info(const vg_lite_buffer_t *buffer)
```

```
void lv_vg_lite_matrix_dump_info(const vg_lite_matrix_t *matrix)
```

```
void lv_vg_lite_color_dump_info(const vg_lite_color_t color)
```

```
bool lv_vg_lite_is_dest_cf_supported(lv_color_format_t cf)
```

```

bool lv_vg_lite_is_src_cf_supported(lv_color_format_t cf)
vg_lite_buffer_format_t lv_vg_lite_vg_fmt(lv_color_format_t cf)

void lv_vg_lite_buffer_format_bytes(vg_lite_buffer_format_t format, uint32_t *mul, uint32_t *div, uint32_t
                                    *bytes_align)

uint32_t lv_vg_lite_width_to_stride(uint32_t w, vg_lite_buffer_format_t color_format)

void lv_vg_lite_buffer_init(vg_lite_buffer_t *buffer, const void *ptr, int32_t width, int32_t height, uint32_t
                           stride, vg_lite_buffer_format_t format, bool tiled)

void lv_vg_lite_buffer_from_draw_buf(vg_lite_buffer_t *buffer, const lv_draw_buf_t *draw_buf)

void lv_vg_lite_image_matrix(vg_lite_matrix_t *matrix, int32_t x, int32_t y, const lv_draw_image_dsc_t *dsc)

vg_lite_color_t lv_vg_lite_recolor(vg_lite_buffer_t *buffer, const lv_draw_image_dsc_t *dsc)

bool lv_vg_lite_buffer_open_image(vg_lite_buffer_t *buffer, lv_image_decoder_dsc_t *decoder_dsc, const
                                  void *src, bool no_cache, bool premultiply)

void lv_vg_lite_image_dsc_init(struct lv_draw_vg_lite_unit_t *unit)
void lv_vg_lite_image_dsc_deinit(struct lv_draw_vg_lite_unit_t *unit)

vg_lite_blend_t lv_vg_lite_blend_mode(lv_blend_mode_t blend_mode, bool has_pre_mul)

uint32_t lv_vg_lite_get_palette_size(vg_lite_buffer_format_t format)

vg_lite_color_t lv_vg_lite_color(lv_color_t color, lv_opa_t opa, bool pre_mul)

void lv_vg_lite_rect(vg_lite_rectangle_t *rect, const lv_area_t *area)

static inline void lv_vg_lite_matrix(vg_lite_matrix_t *dest, const lv_matrix_t *src)

bool lv_vg_lite_buffer_check(const vg_lite_buffer_t *buffer, bool is_src)

bool lv_vg_lite_path_check(const vg_lite_path_t *path)

bool lv_vg_lite_matrix_check(const vg_lite_matrix_t *matrix)

bool lv_vg_lite_support_blend_normal(void)

void lv_vg_lite_matrix_multiply(vg_lite_matrix_t *matrix, const vg_lite_matrix_t *mult)

bool lv_vg_lite_matrix_inverse(vg_lite_matrix_t *result, const vg_lite_matrix_t *matrix)

lv_point_precise_t lv_vg_lite_matrix_transform_point(const vg_lite_matrix_t *matrix, const
                                                       lv_point_precise_t *point)

void lv_vg_lite_set_scissor_area(struct lv_draw_vg_lite_unit_t *u, const lv_area_t *area)

void lv_vg_lite_disable_scissor(void)

void lv_vg_lite_flush(struct lv_draw_vg_lite_unit_t *u)

void lv_vg_lite_finish(struct lv_draw_vg_lite_unit_t *u)

void lv_vg_lite_set_dump_param_enable(bool enable)

bool lv_vg_lite_is_dump_param_enabled(void)

static inline void lv_vg_lite_draw(vg_lite_buffer_t *target, vg_lite_path_t *path, vg_lite_fill_t fill_rule,
                                   vg_lite_matrix_t *matrix, vg_lite_blend_t blend, vg_lite_color_t color)

```

```
static inline void lv_vg_lite_draw_pattern(vg_lite_buffer_t *target, vg_lite_path_t *path, vg_lite_fill_t fill_rule,
                                         vg_lite_matrix_t *path_matrix, vg_lite_buffer_t *pattern_image,
                                         vg_lite_matrix_t *pattern_matrix, vg_lite_blend_t blend,
                                         vg_lite_pattern_mode_t pattern_mode, vg_lite_color_t
                                         pattern_color, vg_lite_color_t color, vg_lite_filter_t filter)

static inline void lv_vg_lite_blt_rect(vg_lite_buffer_t *target, vg_lite_buffer_t *source, vg_lite_rectangle_t
                                         *rect, vg_lite_matrix_t *matrix, vg_lite_blend_t blend, vg_lite_color_t
                                         color, vg_lite_filter_t filter)

void lv_vg_lite_set_color_key(const lv_image_colorkey_t *colorkey)

static inline void lv_vg_lite_clear(vg_lite_buffer_t *target, const lv_area_t *area, vg_lite_color_t color)
```

14.16 drivers

14.16.1 lv_drivers.h

14.16.2 display

drm

lv_linux_drm.h

Typedef

typedef drmModeModeInfo **lv_linux_drm_mode_t**

typedef size_t (***lv_linux_drm_select_mode_cb_t**)(lv_display_t *disp, const *lv_linux_drm_mode_t* *modes,
size_t mode_count)

Callback function type for selecting a DRM display mode

Param disp

pointer to the display object

Param modes

array of available DRM modes

Param mode_count

number of modes in the array

Return

index of the selected mode from the modes array

Funzioni

lv_display_t *lv_linux_drm_create(void)

Create a new Linux DRM display.

Creates and initializes a new LVGL display using the Linux DRM (Direct Rendering Manager) subsystem for hardware-accelerated graphics output.

Ritorna

Pointer to the created display object, or NULL on failure

void lv_linux_drm_set_file(lv_display_t *disp, const char *file, int64_t connector_id)

Configure the DRM device file and connector for a display.

Sets the DRM device file path and connector ID to use for the specified display. The DRM device file is typically located at /dev/dri/cardN where N is the card number. The connector ID specifies which physical output (HDMI, VGA, etc.) to use.

Parametri

- **disp** -- Pointer to the display object created with *lv_linux_drm_create()*
- **file** -- Path to the DRM device file (e.g., "/dev/dri/card0")

- **connector_id** -- ID of the DRM connector to use, or -1 to auto-select the first available

`char *lv_linux_drm_find_device_path(void)`

Automatically find a suitable DRM device path.

Scans the system for available DRM devices and returns the path to a suitable device file that can be used with `lv_linux_drm_set_file()`.

Ritorna

Dynamically allocated string containing the device path (must be freed with `lv_free()`), or NULL if no suitable device is found

`void lv_linux_drm_set_mode_cb(lv_display_t *disp, lv_linux_drm_select_mode_cb_t callback)`

Set a callback function for custom DRM mode selection to override the default mode selection behavior

The default mode selection behavior is selecting the native mode

Parametri

- **disp** -- pointer to the display object
- **callback** -- function to be called when a display mode needs to be selected, or NULL to use the default mode selection behavior

`int32_t lv_linux_drm_mode_get_horizontal_resolution(const lv_linux_drm_mode_t *mode)`

Get the horizontal resolution of a DRM mode

Parametri

`mode` -- pointer to the DRM mode object

Ritorna

horizontal resolution in pixels, or 0 if mode is invalid

`int32_t lv_linux_drm_mode_get_vertical_resolution(const lv_linux_drm_mode_t *mode)`

Get the vertical resolution of a DRM mode

Parametri

`mode` -- pointer to the DRM mode object

Ritorna

vertical resolution in pixels, or 0 if mode is invalid

`int32_t lv_linux_drm_mode_get_refresh_rate(const lv_linux_drm_mode_t *mode)`

Get the refresh rate of a DRM mode

Parametri

`mode` -- pointer to the DRM mode object

Ritorna

refresh rate in Hz, or 0 if mode is invalid

`bool lv_linux_drm_mode_is_preferred(const lv_linux_drm_mode_t *mode)`

Check if a DRM mode is the preferred mode for the display

Parametri

`mode` -- pointer to the DRM mode object

Ritorna

true if this is the preferred/native mode, false otherwise

lv_linux_drm_egl_private.h

fb

lv_linux_fbdev.h

Funzioni

`lv_display_t *lv_linux_fbdev_create(void)`

`void lv_linux_fbdev_set_file(lv_display_t *disp, const char *file)`

```
void lv_linux_fbdev_set_force_refresh(lv_display_t *disp, bool enabled)
```

Force the display to be refreshed on every change. Expected to be used with LV_DISPLAY_RENDER_MODE_DIRECT or LV_DISPLAY_RENDER_MODE_FULL.

ft81x

lv_ft81x.h

Typedef

```
typedef void (*lv_ft81x_spi_cb_t)(lv_display_t *disp, lv_ft81x_spi_operation_t operation, void *data, uint32_t length)
```

Enum

```
enum lv_ft81x_spi_operation_t
```

Values:

enumerator **LV_FT81X_SPI_OPERATION_CS_ASSERT**

enumerator **LV_FT81X_SPI_OPERATION_CS_DEASSERT**

enumerator **LV_FT81X_SPI_OPERATION_SEND**

enumerator **LV_FT81X_SPI_OPERATION_RECEIVE**

Funzioni

```
lv_display_t *lv_ft81x_create(const lv_ft81x_parameters_t *params, void *partial_buf, uint32_t buf_size,
                               lv_ft81x_spi_cb_t spi_cb, void *user_data)
```

Create a framebuffer-based ft81x driver display.

Parametri

- **params** -- pointer to a struct of display panel properties. does not need to be static.
- **partial_buf** -- a single partial buffer
- **buf_size** -- size of the partial buffer
- **spi_cb** -- a callback called by the driver to perform SPI operations
- **user_data** -- use `lv_ft81x_get_user_data` to get this pointer inside the SPI callback

Ritorna

pointer to the display

```
void *lv_ft81x_get_user_data(lv_display_t *disp)
```

Get the `user_data` parameter that was passed to `lv_ft81x_create`. Useful in the SPI callback.

Parametri

disp -- pointer to the ft81x display

Ritorna

the `user_data` pointer

```
struct lv_ft81x_parameters_t
```

Membri Pubblici**uint16_t hor_res**

active display width

uint16_t ver_res

active display height

uint16_t hcycle

total number of clocks per line, incl front/back porch

uint16_t hoffset

start of active line

uint16_t hsync0

start of horizontal sync pulse

uint16_t hsync1

end of horizontal sync pulse

uint16_t vcycle

total number of lines per screen, including pre/post

uint16_t voffset

start of active screen

uint16_t vsync0

start of vertical sync pulse

uint16_t vsync1

end of vertical sync pulse

uint8_t swizzle

FT8xx output to LCD - pin order

uint8_t pclkpol

LCD data is clocked in on this PCLK edge

uint8_t cspread

helps with noise, when set to 1 fewer signals are changed simultaneously, reset-default: 1

uint8_t pclk

60MHz / pclk = pclk frequency

bool has_crystal

has an external clock crystal

bool is_bt81x

is a BT series model, not FT

lv_ft81x_defines.h

Copied from https://github.com/lvgl/lvgl_esp32_drivers/blob/9fed1cc47b5a45fec6bae08b55d2147d3b50260c/lvgl_tft/EVE.h

Define**FT81X_ENABLE****DL_CLEAR****DL_CLEAR_RGB****DL_COLOR_RGB****DL_POINT_SIZE****DL_END****DL_BEGIN****DL_DISPLAY****CLR_COL****CLR_STN****CLR_TAG****SPI_WIDTH_SIO****SPI_WIDTH_DIO****SPI_WIDTH_QIO****EVE_ACTIVE****EVE_STANDBY****EVE_SLEEP****EVE_PWRDOWN****EVE_CLKEXT****EVE_CLKINT**

EVE_CORERST

EVE_CLK48M

EVE_CLK36M

EVE_NEVER

EVE_LESS

EVE_EQUAL

EVE_GREATER

EVE_GEQUAL

EVE_EQUAL

EVE_NOTEQUAL

EVE_ALWAYS

EVE_ARGB1555

EVE_L1

EVE_L4

EVE_L8

EVE_RGB332

EVE_ARGB2

EVE_ARGB4

EVE_RGB565

EVE_PALETTED

EVE_TEXT8X8

EVE_TEXTVGA

EVE_BARGRAPH

EVE_NEAREST

EVE_BILINEAR

EVE_BORDER

EVE_REPEAT

EVE_KEEP

EVE_REPLACE

EVE_INCR

EVE_DECR

EVE_INVERT

EVE_DL_SWAP_DONE

EVE_DL_SWAP_LINE

EVE_DL_SWAP_FRAME

EVE_INT_SWAP

EVE_INT_TOUCH

EVE_INT_TAG

EVE_INT_SOUND

EVE_INT_PLAYBACK

EVE_INT_CMDEMPTY

EVE_INT_CMDFLAG

EVE_INT_CONVCOMPLETE

EVE_TMODE_OFF

EVE_TMODE_ONESHOT

EVE_TMODE_FRAME

EVE_TMODE_CONTINUOUS

EVE_ZERO

EVE_ONE

EVE_SRC_ALPHA

EVE_DST_ALPHA

EVE_ONE_MINUS_SRC_ALPHA

EVE_ONE_MINUS_DST_ALPHA

EVE_BITMAPS

EVE_POINTS

EVE_LINES

EVE_LINE_STRIP

EVE_EDGE_STRIP_R

EVE_EDGE_STRIP_L

EVE_EDGE_STRIP_A

EVE_EDGE_STRIP_B

EVE_RECTS

EVE_OPT_MONO

EVE_OPT_NODL

EVE_OPT_FLAT

EVE_OPT_CENTERX

EVE_OPT_CENTERY

EVE_OPT_CENTER

EVE_OPT_NOBACK

EVE_OPT_NOTICKS

EVE_OPT_NOHM

EVE_OPT_NOPINTER

EVE_OPT_NOSECS

EVE_OPT_NOHANDS

EVE_OPT_RIGHTX

EVE_OPT_SIGNED

EVE_NUMCHAR_PERFONT

EVE_FONT_TABLE_SIZE

EVE_FONT_TABLE_POINTER

EVE_LINEAR_SAMPLES

EVE_ULAW_SAMPLES

EVE_ADPCM_SAMPLES

EVE_SILENCE

EVE_SQUAREWAVE

EVE_SINEWAVE

EVE_SAWTOOTH

EVE_TRIANGLE

EVE_BEEPING

EVE_ALARM

EVE_WARBLE

EVE_CAROUSEL

EVE_PIPS(n)

EVE_HARP

EVE_XYLOPHONE

EVE_TUBA

EVE_GLOCKENSPIEL

EVE_ORGAN

EVE_TRUMPET

EVE_PIANO

EVE_CHIMES

EVE_MUSICBOX

EVE_BELL

EVE_CLICK

EVE_SWITCH

EVE_COWBELL

EVE_NOTCH

EVE_HIHAT

EVE_KICKDRUM

EVE_POP

EVE_CLACK

EVE_CHACK

EVE_MUTE

EVE_UNMUTE

EVE_MIDI_A0

EVE_MIDI_A_0

EVE_MIDI_B0

EVE_MIDI_C1

EVE_MIDI_C_1

EVE_MIDI_D1

EVE_MIDI_D_1

EVE_MIDI_E1

EVE_MIDI_F1

EVE_MIDI_F_1

EVE_MIDI_G1

EVE_MIDI_G_1

EVE_MIDI_A1

EVE_MIDI_A_1

EVE_MIDI_B1

EVE_MIDI_C2

EVE_MIDI_C_2

EVE_MIDI_D2

EVE_MIDI_D_2

EVE_MIDI_E2

EVE_MIDI_F2

EVE_MIDI_F_2

EVE_MIDI_G2

EVE_MIDI_G_2

EVE_MIDI_A2

EVE_MIDI_A_2

EVE_MIDI_B2

EVE_MIDI_C3

EVE_MIDI_C_3

EVE_MIDI_D3

EVE_MIDI_D_3

EVE_MIDI_E3

EVE_MIDI_F3

EVE_MIDI_F_3

EVE_MIDI_G3

EVE_MIDI_G_3

EVE_MIDI_A3

EVE_MIDI_A_3

EVE_MIDI_B3

EVE_MIDI_C4

EVE_MIDI_C_4

EVE_MIDI_D4

EVE_MIDI_D_4

EVE_MIDI_E4

EVE_MIDI_F4

EVE_MIDI_F_4

EVE_MIDI_G4

EVE_MIDI_G_4

EVE_MIDI_A4

EVE_MIDI_A_4

EVE_MIDI_B4

EVE_MIDI_C5

EVE_MIDI_C_5

EVE_MIDI_D5

EVE_MIDI_D_5

EVE_MIDI_E5

EVE_MIDI_F5

EVE_MIDI_F_5

EVE_MIDI_G5

EVE_MIDI_G_5

EVE_MIDI_A5

EVE_MIDI_A_5

EVE_MIDI_B5

EVE_MIDI_C6

EVE_MIDI_C_6

EVE_MIDI_D6

EVE_MIDI_D_6

EVE_MIDI_E6

EVE_MIDI_F6

EVE_MIDI_F_6

EVE_MIDI_G6

EVE_MIDI_G_6

EVE_MIDI_A6

EVE_MIDI_A_6

EVE_MIDI_B6

EVE_MIDI_C7

EVE_MIDI_C_7

EVE_MIDI_D7

EVE_MIDI_D_7

EVE_MIDI_E7

EVE_MIDI_F7

EVE_MIDI_F_7

EVE_MIDI_G7

EVE_MIDI_G_7

EVE_MIDI_A7

EVE_MIDI_A_7

EVE_MIDI_B7

EVE_MIDI_C8

EVE_GPIO0

EVE_GPIO1

EVE_GPIO7

EVE_DISPLAY_0

EVE_DISPLAY_180

CMD_APPEND

CMD_BGCOLOR

CMD_BUTTON

CMD_CALIBRATE

CMD_CLOCK

CMD_COLDSTART

CMD_DIAL

CMD_DLSTART

CMD_FGCOLOR

CMD_GAUGE

CMD_GETMATRIX

CMD_GETPROPS

CMD_GETPTR

CMD_GRADCOLOR

CMD_GRADIENT

CMD_INFLATE

CMD_INTERRUPT

CMD_KEYS

CMD_LOADIDENTITY

CMD_LOADIMAGE

CMD_LOGO

CMD_MEMCPY

CMD_MEMCRC

CMD_MEMSET

CMD_MEMWRITE

CMD_MEMZERO

CMD_NUMBER

CMD_PROGRESS

CMD_REGREAD

CMD_ROTATE

CMD_SCALE

CMD_SCREENSAVER

CMD_SCROLLBAR

CMD_SETFONT

CMD_SETMATRIX

CMD_SKETCH

CMD_SLIDER

CMD_SNAPSHOT

CMD_SPINNER

CMD_STOP

CMD_SWAP

CMD_TEXT

CMD_TOGGLE

CMD_TRACK

CMD_TRANSLATE

ALPHA_FUNC(func, ref)

BEGIN(prim)

BITMAP_HANDLE(handle)

BITMAP_LAYOUT(format, linestride, height)
BITMAP_SIZE(filter, wrapx, wrapy, width, height)
BITMAP_TRANSFORM_A(a)
BITMAP_TRANSFORM_B(b)
BITMAP_TRANSFORM_C(c)
BITMAP_TRANSFORM_D(d)
BITMAP_TRANSFORM_E(e)
BITMAP_TRANSFORM_F(f)
BLEND_FUNC(src, dst)
CALL(dest)
CELL(cell)
CLEAR(c, s, t)
CLEAR_COLOR_A(alpha)
CLEAR_COLOR_RGB(red, green, blue)
CLEAR_STENCIL(s)
CLEAR_TAG(s)
COLOR_A(alpha)
COLOR_MASK(r, g, b, a)
COLOR_RGB(red, green, blue)
END()
JUMP(dest)
LINE_WIDTH(width)
MACRO(m)
POINT_SIZE(size)
RESTORE_CONTEXT()
RETURN()
SAVE_CONTEXT()
STENCIL_FUNC(func, ref, mask)
STENCIL_MASK(mask)
STENCIL_OP(sfail, spass)
TAG(s)
TAG_MASK(mask)
VERTEX2F(x, y)

VERTEX2II(x, y, handle, cell)

EVE_CLKSEL

EVE_RST_PULSE

EVE_PINDRIVE

EVE_PIN_PD_STATE

EVE_RAM_G

EVE_ROM_CHIPID

EVE_ROM_FONT

EVE_ROM_FONT_ADDR

EVE_RAM_DL

EVE_RAM_REG

EVE_RAM_CMD

EVE_RAM_G_SIZE

EVE_CMDFIFO_SIZE

EVE_RAM_DL_SIZE

EVE_ADC_DIFFERENTIAL

EVE_ADC_SINGLE_ENDED

EVE_INT_G8

EVE_INT_L8C

EVE_INT_VGA

EVE_OPT_MEDIAFIFO

EVE_OPT_FULLSCREEN

EVE_OPT_NOTEAR

EVE_OPT_SOUND

EVE_PALETTED565

EVE_PALETTED4444

EVE_PALETTED8

EVE_L2

CMD_MEDIAFIFO

CMD_PLAYVIDEO

CMD_ROMFONT

CMD_SETBASE

CMD_SETBITMAP

CMD_SETFONT2

CMD_SETROTATE

CMD_SETSCRATCH

CMD_SNAPSHOT2

CMD_VIDEOFRAME

CMD_VIDEOSTART

REG_ANA_COMP

REG_BIST_EN

REG_CLOCK

REG_CMDB_SPACE

REG_CMDB_WRITE

REG_CMD_DL

REG_CMD_READ

REG_CMD_WRITE

REG_CPURESET

REG_CSPREAD

REG_CTOUCH_EXTENDED

REG_CTOUCH_TOUCH0_XY

REG_CTOUCH_TOUCH4_X

REG_CTOUCH_TOUCH4_Y

REG_CTOUCH_TOUCH1_XY

REG_CTOUCH_TOUCH2_XY

REG_CTOUCH_TOUCH3_XY

REG_TOUCH_CONFIG

REG_DATESTAMP

REG_DITHER

REG_DL_SWAP

REG_FRAMES

REG_FREQUENCY

REG_GPIO

REG_GPIOX

REG_GPIOX_DIR

REG_GPIO_DIR

REG_HCYLE

REG_HOFFSET

REG_HSIZE

`REG_HSYNC0`

`REG_HSYNC1`

`REG_ID`

`REG_INT_EN`

`REG_INT_FLAGS`

`REG_INT_MASK`

`REG_MACRO_0`

`REG_MACRO_1`

`REG_MEDIAFIFO_READ`

`REG_MEDIAFIFO_WRITE`

`REG_OUTBITS`

`REG_PCLK`

`REG_PCLK_POL`

`REG_PLAY`

`REG_PLAYBACK_FORMAT`

`REG_PLAYBACK_FREQ`

`REG_PLAYBACK_LENGTH`

`REG_PLAYBACK_LOOP`

`REG_PLAYBACK_PLAY`

`REG_PLAYBACK_READPTR`

`REG_PLAYBACK_START`

`REG_PWM_DUTY`

`REG_PWM_HZ`

REG_RENDERMODE

REG_ROTATE

REG_SNAPFORMAT

REG_SNAPSHOT

REG_SNAPY

REG_SOUND

REG_SPI_WIDTH

REG_SWIZZLE

REG_TAG

REG_TAG_X

REG_TAG_Y

REG_TAP_CRC

REG_TAP_MASK

REG_TOUCH_ADC_MODE

REG_TOUCH_CHARGE

REG_TOUCH_DIRECT_XY

REG_TOUCH_DIRECT_Z1Z2

REG_TOUCH_MODE

REG_TOUCH_OVERSAMPLE

REG_TOUCH_RAW_XY

REG_TOUCH_RZ

REG_TOUCH_RZTHRESH

REG_TOUCH_SCREEN_XY

REG_TOUCH_SETTLE

REG_TOUCH_TAG

REG_TOUCH_TAG1

REG_TOUCH_TAG1_XY

REG_TOUCH_TAG2

REG_TOUCH_TAG2_XY

REG_TOUCH_TAG3

REG_TOUCH_TAG3_XY

REG_TOUCH_TAG4

REG_TOUCH_TAG4_XY

REG_TOUCH_TAG_XY

REG_TOUCH_TRANSFORM_A

REG_TOUCH_TRANSFORM_B

REG_TOUCH_TRANSFORM_C

REG_TOUCH_TRANSFORM_D

REG_TOUCH_TRANSFORM_E

REG_TOUCH_TRANSFORM_F

REG_TRACKER

REG_TRACKER_1

REG_TRACKER_2

REG_TRACKER_3

REG_TRACKER_4

REG_TRIM

REG_VCYCLE**REG_VOFFSET****REG_VOL_PB****REG_VOL_SOUND****REG_VSIZE****REG_VSYNC0****REG_VSYNC1****BITMAP_LAYOUT_H**(linestride, height)**BITMAP_SIZE_H**(width, height)**BITMAP_SOURCE**(addr)**PALETTE_SOURCE**(addr)**SCISSOR_SIZE**(width, height)**SCISSOR_XY**(x, y)**VERTEX_FORMAT**(frac)**VERTEX_TRANSLATE_X**(x)**VERTEX_TRANSLATE_Y**(y)

ili9341

lv_ili9341.h

This driver is just a wrapper around the generic MIPI compatible LCD controller driver

Typedef

```
typedef lv_lcd_send_cmd_cb_t lv_ili9341_send_cmd_cb_t
```

```
typedef lv_lcd_send_color_cb_t lv_ili9341_send_color_cb_t
```

Funzioni

```
lv_display_t *lv_ili9341_create(uint32_t hor_res, uint32_t ver_res, lv_lcd_flag_t flags,
                                lv_ili9341_send_cmd_cb_t send_cmd_cb, lv_ili9341_send_color_cb_t
                                send_color_cb)
```

Create an LCD display with ILI9341 driver

Parametri

- **hor_res** -- horizontal resolution
- **ver_res** -- vertical resolution
- **flags** -- default configuration settings (mirror, RGB ordering, etc.)
- **send_cmd** -- platform-dependent function to send a command to the LCD controller
(usually uses polling transfer)

- **send_color** -- platform-dependent function to send pixel data to the LCD controller
(usually uses DMA transfer: must implement a 'ready' callback)

Ritorna

pointer to the created display

void lv ili9341_set_gap(lv_display_t *disp, uint16_t x, uint16_t y)

Set gap, i.e., the offset of the (0,0) pixel in the VRAM

Parametri

- **disp** -- display object
- **x** -- x offset
- **y** -- y offset

void lv ili9341_set_invert(lv_display_t *disp, bool invert)

Set color inversion

Parametri

- **disp** -- display object
- **invert** -- false: normal, true: invert

void lv ili9341_set_gamma_curve(lv_display_t *disp, uint8_t gamma)

Set gamma curve

Parametri

- **disp** -- display object
- **gamma** -- gamma curve

void lv ili9341_send_cmd_list(lv_display_t *disp, const uint8_t *cmd_list)

Send list of commands.

Parametri

- **disp** -- display object
- **cmd_list** -- controller and panel-specific commands

lcd**lv_lcd_generic_mipi.h**

Generic driver for controllers adhering to the MIPI DBI/DCS specification

Works with:

ST7735 ST7789 ST7796 ILI9341 ILI9488 (NOTE: in SPI mode ILI9488 only supports RGB666 mode, which is currently not supported)

any probably many more

Define

LV_LCD_CMD_NOP

LV_LCD_CMD_SOFT_RESET

LV_LCD_CMD_GET_POWER_MODE

LV_LCD_CMD_GET_ADDRESS_MODE

LV_LCD_CMD_GET_PIXEL_FORMAT

LV_LCD_CMD_GET_DISPLAY_MODE

LV_LCD_CMD_GET_SIGNAL_MODE

LV_LCD_CMD_GET_DIAGNOSTIC_RESULT

LV_LCD_CMD_ENTER_SLEEP_MODE

LV_LCD_CMD_EXIT_SLEEP_MODE

LV_LCD_CMD_ENTER_PARTIAL_MODE

LV_LCD_CMD_ENTER_NORMAL_MODE

LV_LCD_CMD_EXIT_INVERT_MODE

LV_LCD_CMD_ENTER_INVERT_MODE

LV_LCD_CMD_SET_GAMMA_CURVE

LV_LCD_CMD_SET_DISPLAY_OFF

LV_LCD_CMD_SET_DISPLAY_ON

LV_LCD_CMD_SET_COLUMN_ADDRESS

LV_LCD_CMD_SET_PAGE_ADDRESS

LV_LCD_CMD_WRITE_MEMORY_START

LV_LCD_CMD_READ_MEMORY_START

LV_LCD_CMD_SET_PARTIAL_ROWS

LV_LCD_CMD_SET_PARTIAL_COLUMNS

LV_LCD_CMD_SET_SCROLL_AREA

LV_LCD_CMD_SET_TEAR_OFF

LV_LCD_CMD_SET_TEAR_ON

LV_LCD_CMD_SET_ADDRESS_MODE

LV_LCD_CMD_SET_SCROLL_START

LV_LCD_CMD_EXIT_IDLE_MODE

LV_LCD_CMD_ENTER_IDLE_MODE

LV_LCD_CMD_SET_PIXEL_FORMAT

LV_LCD_CMD_WRITE_MEMORY_CONTINUE

LV_LCD_CMD_READ_MEMORY_CONTINUE

LV_LCD_CMD_SET_TEAR_SCANLINE

LV_LCD_CMD_GET_SCANLINE

LV_LCD_CMD_READ_DDB_CONTINUE

LV_LCD_CMD_READ_DDB_START

LV_LCD_MASK_FLIP_VERTICAL

LV_LCD_MASK_FLIP_HORIZONTAL

LV_LCD_MASK_DATA_LATCH_DATA_ORDER

LV_LCD_MASK_RGB_ORDER

LV_LCD_MASK_LINE_ADDRESS_ORDER

LV_LCD_MASK_PAGE_COLUMN_ORDER

LV_LCD_MASK_COLUMN_ADDRESS_ORDER

LV_LCD_MASK_PAGE_ADDRESS_ORDER

LV_LCD_BIT_FLIP_VERTICAL__NOT_FLIPPED

LV_LCD_BIT_FLIP_VERTICAL__FLIPPED

LV_LCD_BIT_FLIP_HORIZONTAL__NOT_FLIPPED

LV_LCD_BIT_FLIP_HORIZONTAL__FLIPPED

LV_LCD_BIT_DATA_LATCH_DATA_ORDER__LTOR

LV_LCD_BIT_DATA_LATCH_DATA_ORDER__RTOL

LV_LCD_BIT_RGB_ORDER__RGB

`LV_LCD_BIT_RGB_ORDER_BGR`

`LV_LCD_BIT_LINE_ADDRESS_ORDER_TTOB`

`LV_LCD_BIT_LINE_ADDRESS_ORDER_BTOT`

`LV_LCD_BIT_PAGE_COLUMN_ORDER_NORMAL`

`LV_LCD_BIT_PAGE_COLUMN_ORDER_REVERSE`

`LV_LCD_BIT_COLUMN_ADDRESS_ORDER_LTOR`

`LV_LCD_BIT_COLUMN_ADDRESS_ORDER_RTOL`

`LV_LCD_BIT_PAGE_ADDRESS_ORDER_TTOB`

`LV_LCD_BIT_PAGE_ADDRESS_ORDER_BTOT`

`LV_LCD_GAMMA_2_2`

`LV_LCD_GAMMA_1_8`

`LV_LCD_GAMMA_2_5`

`LV_LCD_GAMMA_1_0`

`LV_LCD_PIXEL_FORMAT_RGB565`

`LV_LCD_PIXEL_FORMAT_RGB666`

`LV_LCD_FLAG_NONE`

`LV_LCD_FLAG_MIRROR_X`

`LV_LCD_FLAG_MIRROR_Y`

`LV_LCD_FLAG_BGR`

`LV_LCD_FLAG_RGB666`

`LV_LCD_CMD_DELAY_MS`

`LV_LCD_CMD_EOF`

Typedef

`typedef uint32_t lv_lcd_flag_t`

Configuration flags for lv_lcd_xxx_create()

`typedef void (*lv_lcd_send_cmd_cb_t)(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, const uint8_t *param, size_t param_size)`

Prototype of a platform-dependent callback to transfer commands and data to the LCD controller.

Param disp

display object

Param cmd

command buffer (can handle 16 bit commands as well)

Param cmd_size

number of bytes of the command

Param param

parameter buffer

Param param_size

number of bytes of the parameters

`typedef void (*lv_lcd_send_color_cb_t)(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, uint8_t *param, size_t param_size)`

Prototype of a platform-dependent callback to transfer pixel data to the LCD controller.

Param disp

display object

Param cmd

command buffer (can handle 16 bit commands as well)

Param cmd_size

number of bytes of the command

Param param

parameter buffer

Param param_size

number of bytes of the parameters

Funzioni

`lv_display_t *lv_lcd_generic_mipi_create(uint32_t hor_res, uint32_t ver_res, lv_lcd_flag_t flags, lv_lcd_send_cmd_cb_t send_cmd_cb, lv_lcd_send_color_cb_t send_color_cb)`

Create a MIPI DCS compatible LCD display

Parametri

- **hor_res** -- horizontal resolution
- **ver_res** -- vertical resolution
- **flags** -- default configuration settings (mirror, RGB ordering, etc.)
- **send_cmd** -- platform-dependent function to send a command to the LCD controller (usually uses polling transfer)
- **send_color** -- platform-dependent function to send pixel data to the LCD controller (usually uses DMA transfer). `lv_display_flush_ready` must be called after the transfer has finished.

Ritorna

pointer to the created display

`void lv_lcd_generic_mipi_set_gap(lv_display_t *disp, uint16_t x, uint16_t y)`

Set gap, i.e., the offset of the (0,0) pixel in the VRAM

Parametri

- **disp** -- display object
- **x** -- x offset
- **y** -- y offset

```
void lv_lcd_generic_mipi_set_invert(lv_display_t *disp, bool invert)
    Set color inversion
    Parametri
        • disp -- display object
        • invert -- false: normal, true: invert

void lv_lcd_generic_mipi_set_address_mode(lv_display_t *disp, bool mirror_x, bool mirror_y, bool swap_xy,
                                         bool bgr)
    Set address mode
    Parametri
        • disp -- display object
        • mirror_x -- horizontal mirror (false: normal, true: mirrored)
        • mirror_y -- vertical mirror (false: normal, true: mirrored)
        • swap_xy -- swap axes (false: normal, true: swap)
        • bgr -- RGB/BGR order (false: RGB, true: BGR)

void lv_lcd_generic_mipi_set_gamma_curve(lv_display_t *disp, uint8_t gamma)
    Set gamma curve
    Parametri
        • disp -- display object
        • gamma -- gamma curve

void lv_lcd_generic_mipi_send_cmd_list(lv_display_t *disp, const uint8_t *cmd_list)
    Send list of commands.
    Parametri
        • disp -- display object
        • cmd_list -- controller and panel-specific commands
```

```
struct lv_lcd_generic_mipi_driver_t
#include <lv_lcd_generic_mipi.h> Generic MIPI compatible LCD driver
```

Membri Pubblici

lv_display_t ***disp**

lv_lcd_send_cmd_cb_t **send_cmd**

lv_lcd_send_color_cb_t **send_color**

uint16_t **x_gap**

uint16_t **y_gap**

uint8_t **madctl_reg**

uint8_t **colmod_reg**

bool **mirror_x**

bool **mirror_y**

bool **swap_xy**

lovyan_gfx**lv_lgfx_user.hpp****lv_lovyan_gfx.h****Funzioni**

```
lv_display_t *lv_lovyan_gfx_create(uint32_t hor_res, uint32_t ver_res, void *buf, uint32_t buf_size_bytes,
                                    bool touch)
```

nv3007**lv_nv3007.h**

This driver is just a wrapper around the generic MIPI compatible LCD controller driver

Typedef

```
typedef lv_lcd_send_cmd_cb_t lv_nv3007_send_cmd_cb_t
```

```
typedef lv_lcd_send_color_cb_t lv_nv3007_send_color_cb_t
```

Funzioni

```
lv_display_t *lv_nv3007_create(uint32_t hor_res, uint32_t ver_res, lv_flag_t flags,
                               lv_nv3007_send_cmd_cb_t send_cmd_cb, lv_nv3007_send_color_cb_t
                               send_color_cb)
```

Create an LCD display with NV3007 driver

Parametri

- **hor_res** -- horizontal resolution
- **ver_res** -- vertical resolution
- **flags** -- default configuration settings (mirror, RGB ordering, etc.)
- **send_cmd** -- platform-dependent function to send a command to the LCD controller
(usually uses polling transfer)
- **send_color** -- platform-dependent function to send pixel data to the LCD controller
(usually uses DMA transfer: must implement a 'ready' callback)

Ritorna

pointer to the created display

```
void lv_nv3007_set_gap(lv_display_t *disp, uint16_t x, uint16_t y)
```

Set gap, i.e., the offset of the (0,0) pixel in the VRAM

Parametri

- **disp** -- display object
- **x** -- x offset
- **y** -- y offset

```
void lv_nv3007_set_invert(lv_display_t *disp, bool invert)
```

Set color inversion

Parametri

- **disp** -- display object
- **invert** -- false: normal, true: invert

```
void lv_nv3007_set_gamma_curve(lv_display_t *disp, uint8_t gamma)
```

Set gamma curve

Parametri

- **disp** -- display object
- **gamma** -- gamma curve

```
void lv_nv3007_send_cmd_list(lv_display_t *disp, const uint8_t *cmd_list)
```

Send list of commands.

Parametri

- **disp** -- display object
- **cmd_list** -- controller and panel-specific commands

nxp_elcdif

lv_nxp_elcdif.h

Driver for NXP's ELCD

Funzioni

```
lv_display_t *lv_nxp_display_elcdif_create_direct(LCDIF_Type *base, const elcdif_rgb_mode_config_t
                                                 *config, void *frame_buffer1, void *frame_buffer2,
                                                 size_t buf_size)
```

Attach LVGL to ELCDIF using DIRECT rendering mode. ELCDIF should be already initialized.

Parametri

- **base** -- The NXP eLCD controller base address
- **config** -- NXP eLCD config object
- **frame_buffer1** -- pointer the first frame buffers
- **frame_buffer2** -- pointer the second frame buffers
- **buf_size** -- size of a buffer in bytes (must be at least as large as the screen)

Ritorna

a display object initialized and registered on the LVGL runtime

```
lv_display_t *lv_nxp_display_elcdif_create_partial(LCDIF_Type *base, const elcdif_rgb_mode_config_t
                                                 *config, void *frame_buffer1, void *frame_buffer2,
                                                 size_t buf_size)
```

Attach LVGL to ELCDIF using PARTIAL rendering mode. ELCDIF should be already initialized.

Parametri

- **base** -- The NXP eLCD controller base address
- **config** -- NXP eLCD config object
- **frame_buffer1** -- pointer the first frame buffers
- **frame_buffer2** -- pointer the second frame buffers
- **buf_size** -- size of a buffer in bytes

Ritorna

a display object initialized and registered on the LVGL runtime

```
void lv_nxp_display_elcdif_event_handler(const lv_display_t *disp)
```

Call this function on the LCD Interrupt Service Routine It tells to LVGL what to do when a framebuffer is transmitted to the LCD panel

Nota

: the parameter disp is typically the return value after `lv_nxp_display_elcdif_create_direct` has been successfully executed

Parametri

disp -- The display instance that contains the eLCD related data

renesas_glc当地

lv_renesas_glc当地.h

Funzioni

`lv_display_t *lv_renesas_glcdfc_direct_create(void)`

Create a display using Renesas' GLCDC peripheral in DIRECT render mode

Ritorna

pointer to the created display

`lv_display_t *lv_renesas_glcdfc_partial_create(void *buf1, void *buf2, size_t buf_size)`

Create a display using Renesas' GLCDC peripheral in PARTIAL render mode

Parametri

- **buf1** -- first buffer
- **buf2** -- second buffer (can be NULL)
- **buf_size** -- buffer size in byte

Ritorna

pointer to the created display

st7735

lv_st7735.h

This driver is just a wrapper around the generic MIPI compatible LCD controller driver

Typedef

`typedef lv_lcd_send_cmd_cb_t lv_st7735_send_cmd_cb_t`

`typedef lv_lcd_send_color_cb_t lv_st7735_send_color_cb_t`

Funzioni

`lv_display_t *lv_st7735_create(uint32_t hor_res, uint32_t ver_res, lv_lcd_flag_t flags,
 lv_st7735_send_cmd_cb_t send_cmd_cb, lv_st7735_send_color_cb_t
 send_color_cb)`

Create an LCD display with ST7735 driver

Parametri

- **hor_res** -- horizontal resolution
- **ver_res** -- vertical resolution
- **flags** -- default configuration settings (mirror, RGB ordering, etc.)
- **send_cmd** -- platform-dependent function to send a command to the LCD controller (usually uses polling transfer)
- **send_color** -- platform-dependent function to send pixel data to the LCD controller (usually uses DMA transfer: must implement a 'ready' callback)

Ritorna

pointer to the created display

`void lv_st7735_set_gap(lv_display_t *disp, uint16_t x, uint16_t y)`

Set gap, i.e., the offset of the (0,0) pixel in the VRAM

Parametri

- **disp** -- display object
- **x** -- x offset
- **y** -- y offset

`void lv_st7735_set_invert(lv_display_t *disp, bool invert)`

Set color inversion

Parametri

- **disp** -- display object
- **invert** -- false: normal, true: invert

`void lv_st7735_set_gamma_curve(lv_display_t *disp, uint8_t gamma)`

Set gamma curve

Parametri

- **disp** -- display object
- **gamma** -- gamma curve

`void lv_st7735_send_cmd_list(lv_display_t *disp, const uint8_t *cmd_list)`

Send list of commands.

Parametri

- **disp** -- display object
- **cmd_list** -- controller and panel-specific commands

st7789**lv_st7789.h**

This driver is just a wrapper around the generic MIPI compatible LCD controller driver

Typedef

`typedef lv_lcd_send_cmd_cb_t lv_st7789_send_cmd_cb_t`

`typedef lv_lcd_send_color_cb_t lv_st7789_send_color_cb_t`

Funzioni

`lv_display_t *lv_st7789_create(uint32_t hor_res, uint32_t ver_res, lv_lcd_flag_t flags,
 lv_st7789_send_cmd_cb_t send_cmd_cb, lv_st7789_send_color_cb_t
 send_color_cb)`

Create an LCD display with ST7789 driver

Parametri

- **hor_res** -- horizontal resolution
- **ver_res** -- vertical resolution
- **flags** -- default configuration settings (mirror, RGB ordering, etc.)
- **send_cmd** -- platform-dependent function to send a command to the LCD controller (usually uses polling transfer)
- **send_color** -- platform-dependent function to send pixel data to the LCD controller (usually uses DMA transfer: must implement a 'ready' callback)

Ritorna

pointer to the created display

`void lv_st7789_set_gap(lv_display_t *disp, uint16_t x, uint16_t y)`

Set gap, i.e., the offset of the (0,0) pixel in the VRAM

Parametri

- **disp** -- display object
- **x** -- x offset
- **y** -- y offset

`void lv_st7789_set_invert(lv_display_t *disp, bool invert)`

Set color inversion

Parametri

- **disp** -- display object
- **invert** -- false: normal, true: invert

`void lv_st7789_set_gamma_curve(lv_display_t *disp, uint8_t gamma)`

Set gamma curve

Parametri

- **disp** -- display object
- **gamma** -- gamma curve

```
void lv_st7789_send_cmd_list(lv_display_t *disp, const uint8_t *cmd_list)
```

Send list of commands.

Parametri

- **disp** -- display object
- **cmd_list** -- controller and panel-specific commands

st7796

lv_st7796.h

This driver is just a wrapper around the generic MIPI compatible LCD controller driver

Typedef

```
typedef lv_lcd_send_cmd_cb_t lv_st7796_send_cmd_cb_t
```

```
typedef lv_lcd_send_color_cb_t lv_st7796_send_color_cb_t
```

Funzioni

```
lv_display_t *lv_st7796_create(uint32_t hor_res, uint32_t ver_res, lv_lcd_flag_t flags,
                               lv_st7796_send_cmd_cb_t send_cmd_cb, lv_st7796_send_color_cb_t
                               send_color_cb)
```

Create an LCD display with ST7796 driver

Parametri

- **hor_res** -- horizontal resolution
- **ver_res** -- vertical resolution
- **flags** -- default configuration settings (mirror, RGB ordering, etc.)
- **send_cmd** -- platform-dependent function to send a command to the LCD controller
(usually uses polling transfer)
- **send_color** -- platform-dependent function to send pixel data to the LCD controller
(usually uses DMA transfer: must implement a 'ready' callback)

Ritorna

pointer to the created display

```
void lv_st7796_set_gap(lv_display_t *disp, uint16_t x, uint16_t y)
```

Set gap, i.e., the offset of the (0,0) pixel in the VRAM

Parametri

- **disp** -- display object
- **x** -- x offset
- **y** -- y offset

```
void lv_st7796_set_invert(lv_display_t *disp, bool invert)
```

Set color inversion

Parametri

- **disp** -- display object
- **invert** -- false: normal, true: invert

```
void lv_st7796_set_gamma_curve(lv_display_t *disp, uint8_t gamma)
```

Set gamma curve

Parametri

- **disp** -- display object
- **gamma** -- gamma curve

```
void lv_st7796_send_cmd_list(lv_display_t *disp, const uint8_t *cmd_list)
```

Send list of commands.

Parametri

- **disp** -- display object

- **cmd_list** -- controller and panel-specific commands

st_ltde**lv_st_ltde.h****Funzioni**

`lv_display_t *lv_st_ltde_create_direct(void *fb_addr_1, void *fb_addr_2, uint32_t layer_idx)`

Create a direct render mode display bound to a LTDC layer.

Parametri

- **fb_addr_1** -- The LTDC layer's framebuffer memory address.
- **fb_addr_2** -- An additional framebuffer-sized buffer to use for double buffering, or NULL.
- **layer_idx** -- The LTDC layer number to bind the display to. Typically 0 or 1.

Ritorna

The display.

`lv_display_t *lv_st_ltde_create_partial(void *render_buf_1, void *render_buf_2, uint32_t buf_size, uint32_t layer_idx)`

Create a partial render mode display bound to a LTDC layer. The layer's framebuffer is flushed to internally.

Enable `LV_ST_LTDC_USE_DMA2D_FLUSH` for parallel flushing.

Parametri

- **render_buf_1** -- A render buffer.
- **render_buf_2** -- An additional render buffer for double-buffering, or NULL.
- **buf_size** -- The size of the buffer(s) in bytes.
- **layer_idx** -- The LTDC layer number to bind the display to. Typically 0 or 1.

Ritorna

The display.

tft_espi**lv_tft_espi.h****Funzioni**

`lv_display_t *lv_tft_espi_create(uint32_t hor_res, uint32_t ver_res, void *buf, uint32_t buf_size_bytes)`

14.16.3 draw**eve****lv_draw_eve_display.h****Funzioni**

`lv_display_t *lv_draw_eve_display_create(const lv_draw_eve_parameters_t *params,
 lv_draw_eve_operation_cb_t op_cb, void *user_data)`

Create a display for the EVE draw unit.

Parametri

- **params** -- Pointer to a struct of display parameters. Can be a temporary variable
- **op_cb** -- A callback that will be called to perform pin and SPI IO operations with the EVE chip
- **user_data** -- use `lv_draw_eve_display_get_user_data` to get this pointer inside the `op_cb`

Ritorna

the EVE display

`void *lv_draw_eve_display_get_user_data(lv_display_t *disp)`

Get the `user_data` parameter that was passed to `lv_draw_eve_display_create`. Useful in the operation callback.

Parametri

- **disp** -- pointer to the lv_draw_eve display

Ritorna

the user_data pointer

```
lv_indev_t *lv_draw_eve_touch_create(lv_display_t *disp)
```

Create a touchscreen indev for the EVE display.

Parametri

- **disp** -- pointer to the lv_draw_eve display

Ritorna

the EVE touchscreen indev

```
void lv_draw_eve_pre_upload_image(lv_display_t *disp, const void *src)
```

Upload an image src to RAM_G now instead of as-needed during rendering.

Parametri

- **disp** -- pointer to the lv_draw_eve display
- **src** -- image src. The value passed to `lv_image_set_src`

```
void lv_draw_eve_pre_upload_font_range(lv_display_t *disp, const lv_font_t *font, uint32_t
                                         unicode_range_start, uint32_t unicode_range_end)
```

Upload font glyphs to RAM_G now instead of as-needed during rendering. Upload all the glyphs in the range of unicode code points (inclusive of the start and end values). It can be called multiple times with different ranges.

Parametri

- **disp** -- pointer to the lv_draw_eve display
- **font** -- the font to upload glyphs from
- **unicode_range_start** -- the first unicode code point in the range of glyphs to upload
- **unicode_range_end** -- the last unicode code point (inclusive) in the range of glyphs to upload

```
void lv_draw_eve_pre_upload_font_text(lv_display_t *disp, const lv_font_t *font, const char *text)
```

Upload font glyphs to RAM_G now instead of as-needed during rendering. It will upload all the glyphs needed to render the string text. It can be called multiple times with different strings.

Parametri

- **disp** -- pointer to the lv_draw_eve display
- **font** -- the font to upload glyphs from
- **text** -- the ASCII or UTF-8 string that will be iterated for glyphs to upload

```
uint8_t lv_draw_eve_memread8(lv_display_t *disp, uint32_t address)
```

Call EVE_memRead8 for custom low-level control of the display.

Parametri

- **disp** -- the display returned by `lv_draw_eve_display_create`
- **address** -- the EVE address to read from

Ritorna

the read value

```
uint16_t lv_draw_eve_memread16(lv_display_t *disp, uint32_t address)
```

Call EVE_memRead16 for custom low-level control of the display.

Parametri

- **disp** -- the display returned by `lv_draw_eve_display_create`
- **address** -- the EVE address to read from

Ritorna

the read value

```
uint32_t lv_draw_eve_memread32(lv_display_t *disp, uint32_t address)
```

Call EVE_memRead32 for custom low-level control of the display.

Parametri

- **disp** -- the display returned by `lv_draw_eve_display_create`
- **address** -- the EVE address to read from

Ritorna

the read value

void lv_draw_eve_memwrite8(lv_display_t *disp, uint32_t address, uint8_t data)

Call EVE_memWrite8 for custom low-level control of the display.

Parametri

- **disp** -- the display returned by lv_draw_eve_display_create
- **address** -- the EVE address to write to
- **data** -- the value to write

void lv_draw_eve_memwrite16(lv_display_t *disp, uint32_t address, uint16_t data)

Call EVE_memWrite16 for custom low-level control of the display.

Parametri

- **disp** -- the display returned by lv_draw_eve_display_create
- **address** -- the EVE address to write to
- **data** -- the value to write

void lv_draw_eve_memwrite32(lv_display_t *disp, uint32_t address, uint32_t data)

Call EVE_memWrite32 for custom low-level control of the display.

Parametri

- **disp** -- the display returned by lv_draw_eve_display_create
- **address** -- the EVE address to write to
- **data** -- the value to write

lv_draw_eve_displayDefines.h**Define**

LV_EVE_EVE_RAM_G

LV_EVE_EVE_ROM_CHIPID

LV_EVE_EVE_ROM_FONT

LV_EVE_EVE_ROM_FONTROOT

LV_EVE_EVE_RAM_DL

LV_EVE_EVE_RAM_REG

LV_EVE_EVE_RAM_CMD

LV_EVE_EVE_RAM_G_SIZE

LV_EVE_EVE_CMD FIFO_SIZE

LV_EVE_EVE_RAM_DL_SIZE

LV_EVE_DL_DISPLAY

LV_EVE_DL_BITMAP_SOURCE

`LV_EVE_DL_CLEAR_COLOR_RGB`

`LV_EVE_DL_TAG`

`LV_EVE_DL_COLOR_RGB`

`LV_EVE_DL_BITMAP_HANDLE`

`LV_EVE_DL_CELL`

`LV_EVE_DL_BITMAP_LAYOUT`

`LV_EVE_DL_BITMAP_SIZE`

`LV_EVE_DL_ALPHA_FUNC`

`LV_EVE_DL_STENCIL_FUNC`

`LV_EVE_DL_BLEND_FUNC`

`LV_EVE_DL_STENCIL_OP`

`LV_EVE_DL_POINT_SIZE`

`LV_EVE_DL_LINE_WIDTH`

`LV_EVE_DL_CLEAR_COLOR_A`

`LV_EVE_DL_COLOR_A`

`LV_EVE_DL_CLEAR_STENCIL`

`LV_EVE_DL_CLEAR_TAG`

`LV_EVE_DL_STENCIL_MASK`

`LV_EVE_DL_TAG_MASK`

`LV_EVE_DL_BITMAP_TRANSFORM_A`

`LV_EVE_DL_BITMAP_TRANSFORM_B`

`LV_EVE_DL_BITMAP_TRANSFORM_C`

`LV_EVE_DL_BITMAP_TRANSFORM_D`

LV_EVE_DL_BITMAP_TRANSFORM_E

LV_EVE_DL_BITMAP_TRANSFORM_F

LV_EVE_DL_SCISSOR_XY

LV_EVE_DL_SCISSOR_SIZE

LV_EVE_DL_CALL

LV_EVE_DL_JUMP

LV_EVE_DL_BEGIN

LV_EVE_DL_COLOR_MASK

LV_EVE_DL_END

LV_EVE_DL_SAVE_CONTEXT

LV_EVE_DL_RESTORE_CONTEXT

LV_EVE_DL_RETURN

LV_EVE_DL_MACRO

LV_EVE_DL_CLEAR

LV_EVE_DL_VERTEX_FORMAT

LV_EVE_DL_BITMAP_LAYOUT_H

LV_EVE_DL_BITMAP_SIZE_H

LV_EVE_DL_PALETTE_SOURCE

LV_EVE_DL_VERTEX_TRANSLATE_X

LV_EVE_DL_VERTEX_TRANSLATE_Y

LV_EVE_DL_NOP

LV_EVE_DL_VERTEX2F

LV_EVE_DL_VERTEX2II

`LV_EVE_CLR_COL`

`LV_EVE_CLR_STN`

`LV_EVE_CLR_TAG`

`LV_EVE_EVE_ACTIVE`

`LV_EVE_EVE_STANDBY`

`LV_EVE_EVE_SLEEP`

`LV_EVE_EVE_CLKEXT`

`LV_EVE_EVE_CLKINT`

`LV_EVE_EVE_PWRDOWN`

`LV_EVE_EVE_CLKSEL`

`LV_EVE_EVE_RST_PULSE`

`LV_EVE_EVE_CORERST`

`LV_EVE_EVE_PINDRIVE`

`LV_EVE_EVE_PIN_PD_STATE`

`LV_EVE_EVE_NEVER`

`LV_EVE_EVE_LESS`

`LV_EVE_EVE_EQUAL`

`LV_EVE_EVE_GREATER`

`LV_EVE_EVE_GEQUAL`

`LV_EVE_EVE_EQUAL`

`LV_EVE_EVE_NOTEQUAL`

`LV_EVE_EVE_ALWAYS`

`LV_EVE_EVE_ARGB1555`

`LV_EVE_EVE_L1`

`LV_EVE_EVE_L4`

`LV_EVE_EVE_L8`

`LV_EVE_EVE_RGB332`

`LV_EVE_EVE_ARGB2`

`LV_EVE_EVE_ARGB4`

`LV_EVE_EVE_RGB565`

`LV_EVE_EVE_PALETTED`

`LV_EVE_EVE_TEXT8X8`

`LV_EVE_EVE_TEXTVGA`

`LV_EVE_EVE_BARGRAPH`

`LV_EVE_EVE_NEAREST`

`LV_EVE_EVE_BILINEAR`

`LV_EVE_EVE_BORDER`

`LV_EVE_EVE_REPEAT`

`LV_EVE_EVE_KEEP`

`LV_EVE_EVE_REPLACE`

`LV_EVE_EVE_INCR`

`LV_EVE_EVE_DECR`

`LV_EVE_EVE_INVERT`

`LV_EVE_EVE_DL_SWAP_DONE`

`LV_EVE_EVE_DL_SWAP_LINE`

`LV_EVE_EVE_DL_SWAP_FRAME`

`LV_EVE_EVE_INT_SWAP`

`LV_EVE_EVE_INT_TOUCH`

`LV_EVE_EVE_INT_TAG`

`LV_EVE_EVE_INT_SOUND`

`LV_EVE_EVE_INT_PLAYBACK`

`LV_EVE_EVE_INT_CMDEMPTY`

`LV_EVE_EVE_INT_CMDFLAG`

`LV_EVE_EVE_INT_CONVCOMPLETE`

`LV_EVE_EVE_TMODE_OFF`

`LV_EVE_EVE_TMODE_ONESHOT`

`LV_EVE_EVE_TMODE_FRAME`

`LV_EVE_EVE_TMODE_CONTINUOUS`

`LV_EVE_EVE_ZERO`

`LV_EVE_EVE_ONE`

`LV_EVE_EVE_SRC_ALPHA`

`LV_EVE_EVE_DST_ALPHA`

`LV_EVE_EVE_ONE_MINUS_SRC_ALPHA`

`LV_EVE_EVE_ONE_MINUS_DST_ALPHA`

`LV_EVE_EVE_BITMAPS`

`LV_EVE_EVE_POINTS`

`LV_EVE_EVE_LINES`

`LV_EVE_EVE_LINE_STRIP`

`LV_EVE_EVE_EDGE_STRIP_R`

LV_EVE_EVE_EDGE_STRIP_L

LV_EVE_EVE_EDGE_STRIP_A

LV_EVE_EVE_EDGE_STRIP_B

LV_EVE_EVE_RECTS

LV_EVE_EVE_INT_G8

LV_EVE_EVE_INT_L8C

LV_EVE_EVE_INT_VGA

LV_EVE_EVE_PALETTED565

LV_EVE_EVE_PALETTED4444

LV_EVE_EVE_PALETTED8

LV_EVE_EVE_L2

LV_EVE_EVE_OPT_MONO

LV_EVE_EVE_OPT_NODL

LV_EVE_EVE_OPT_FLAT

LV_EVE_EVE_OPT_CENTERX

LV_EVE_EVE_OPT_CENTERY

LV_EVE_EVE_OPT_CENTER

LV_EVE_EVE_OPT_NOBACK

LV_EVE_EVE_OPT_NOTICKS

LV_EVE_EVE_OPT_NOHM

LV_EVE_EVE_OPT_NOPOINTER

LV_EVE_EVE_OPT_NOSECS

LV_EVE_EVE_OPT_NOHANDS

LV_EVE_EVE_OPT_RIGHTX

LV_EVE_EVE_OPT_SIGNED

LV_EVE_EVE_OPT_MEDIAFIFO

LV_EVE_EVE_OPT_FULLSCREEN

LV_EVE_EVE_OPT_NOTEAR

LV_EVE_EVE_OPT_SOUND

LV_EVE_EVE_ADC_DIFFERENTIAL

LV_EVE_EVE_ADC_SINGLE_ENDED

LV_EVE_EVE_NUMCHAR_PERFONT

LV_EVE_EVE_FONT_TABLE_SIZE

LV_EVE_EVE_FONT_TABLE_POINTER

LV_EVE_EVE_LINEAR_SAMPLES

LV_EVE_EVE_ULAW_SAMPLES

LV_EVE_EVE_ADPCM_SAMPLES

LV_EVE_EVE_SILENCE

LV_EVE_EVE_SQUAREWAVE

LV_EVE_EVE_SINEWAVE

LV_EVE_EVE_SAWTOOTH

LV_EVE_EVE_TRIANGLE

LV_EVE_EVE_BEEPING

LV_EVE_EVE_ALARM

LV_EVE_EVE_WARBLE

LV_EVE_EVE_CAROUSEL

LV_EVE_EVE_PIPS(n)

LV_EVE_EVE_HARP

LV_EVE_EVE_XYLOPHONE

LV_EVE_EVE_TUBA

LV_EVE_EVE_GLOCKENSPIEL

LV_EVE_EVE_ORGAN

LV_EVE_EVE_TRUMPET

LV_EVE_EVE_PIANO

LV_EVE_EVE_CHIMES

LV_EVE_EVE_MUSICBOX

LV_EVE_EVE_BELL

LV_EVE_EVE_CLICK

LV_EVE_EVE_SWITCH

LV_EVE_EVE_COWBELL

LV_EVE_EVE_NOTCH

LV_EVE_EVE_HIHAT

LV_EVE_EVE_KICKDRUM

LV_EVE_EVE_POP

LV_EVE_EVE_CLACK

LV_EVE_EVE_CHACK

LV_EVE_EVE_MUTE

LV_EVE_EVE_UNMUTE

LV_EVE_EVE_MIDI_A0

LV_EVE_EVE_MIDI_A_0

LV_EVE_EVE_MIDI_B0

LV_EVE_EVE_MIDI_C1

LV_EVE_EVE_MIDI_C_1

LV_EVE_EVE_MIDI_D1

LV_EVE_EVE_MIDI_D_1

LV_EVE_EVE_MIDI_E1

LV_EVE_EVE_MIDI_F1

LV_EVE_EVE_MIDI_F_1

LV_EVE_EVE_MIDI_G1

LV_EVE_EVE_MIDI_G_1

LV_EVE_EVE_MIDI_A1

LV_EVE_EVE_MIDI_A_1

LV_EVE_EVE_MIDI_B1

LV_EVE_EVE_MIDI_C2

LV_EVE_EVE_MIDI_C_2

LV_EVE_EVE_MIDI_D2

LV_EVE_EVE_MIDI_D_2

LV_EVE_EVE_MIDI_E2

LV_EVE_EVE_MIDI_F2

LV_EVE_EVE_MIDI_F_2

LV_EVE_EVE_MIDI_G2

LV_EVE_EVE_MIDI_G_2

LV_EVE_EVE_MIDI_A2

LV_EVE_EVE_MIDI_A_2

LV_EVE_EVE_MIDI_B2

LV_EVE_EVE_MIDI_C3

LV_EVE_EVE_MIDI_C_3

LV_EVE_EVE_MIDI_D3

LV_EVE_EVE_MIDI_D_3

LV_EVE_EVE_MIDI_E3

LV_EVE_EVE_MIDI_F3

LV_EVE_EVE_MIDI_F_3

LV_EVE_EVE_MIDI_G3

LV_EVE_EVE_MIDI_G_3

LV_EVE_EVE_MIDI_A3

LV_EVE_EVE_MIDI_A_3

LV_EVE_EVE_MIDI_B3

LV_EVE_EVE_MIDI_C4

LV_EVE_EVE_MIDI_C_4

LV_EVE_EVE_MIDI_D4

LV_EVE_EVE_MIDI_D_4

LV_EVE_EVE_MIDI_E4

LV_EVE_EVE_MIDI_F4

LV_EVE_EVE_MIDI_F_4

LV_EVE_EVE_MIDI_G4

LV_EVE_EVE_MIDI_G_4

LV_EVE_EVE_MIDI_A4

LV_EVE_EVE_MIDI_A_4

LV_EVE_EVE_MIDI_B4

LV_EVE_EVE_MIDI_C5

LV_EVE_EVE_MIDI_C_5

LV_EVE_EVE_MIDI_D5

LV_EVE_EVE_MIDI_D_5

LV_EVE_EVE_MIDI_E5

LV_EVE_EVE_MIDI_F5

LV_EVE_EVE_MIDI_F_5

LV_EVE_EVE_MIDI_G5

LV_EVE_EVE_MIDI_G_5

LV_EVE_EVE_MIDI_A5

LV_EVE_EVE_MIDI_A_5

LV_EVE_EVE_MIDI_B5

LV_EVE_EVE_MIDI_C6

LV_EVE_EVE_MIDI_C_6

LV_EVE_EVE_MIDI_D6

LV_EVE_EVE_MIDI_D_6

LV_EVE_EVE_MIDI_E6

LV_EVE_EVE_MIDI_F6

LV_EVE_EVE_MIDI_F_6

`LV_EVE_EVE_MIDI_G6`

`LV_EVE_EVE_MIDI_G_6`

`LV_EVE_EVE_MIDI_A6`

`LV_EVE_EVE_MIDI_A_6`

`LV_EVE_EVE_MIDI_B6`

`LV_EVE_EVE_MIDI_C7`

`LV_EVE_EVE_MIDI_C_7`

`LV_EVE_EVE_MIDI_D7`

`LV_EVE_EVE_MIDI_D_7`

`LV_EVE_EVE_MIDI_E7`

`LV_EVE_EVE_MIDI_F7`

`LV_EVE_EVE_MIDI_F_7`

`LV_EVE_EVE_MIDI_G7`

`LV_EVE_EVE_MIDI_G_7`

`LV_EVE_EVE_MIDI_A7`

`LV_EVE_EVE_MIDI_A_7`

`LV_EVE_EVE_MIDI_B7`

`LV_EVE_EVE_MIDI_C8`

`LV_EVE_EVE_GPIO00`

`LV_EVE_EVE_GPIO01`

`LV_EVE_EVE_GPIO07`

`LV_EVE_EVE_DISPLAY_0`

`LV_EVE_EVE_DISPLAY_180`

LV_EVE_CMD_APPEND

LV_EVE_CMD_BGCOLOR

LV_EVE_CMD_BUTTON

LV_EVE_CMD_CALIBRATE

LV_EVE_CMD_CLOCK

LV_EVE_CMD_COLDSTART

LV_EVE_CMD_DIAL

LV_EVE_CMD_DLSTART

LV_EVE_CMD_FGCOLOR

LV_EVE_CMD_GAUGE

LV_EVE_CMD_GETMATRIX

LV_EVE_CMD_GETPROPS

LV_EVE_CMD_GETPTR

LV_EVE_CMD_GRADCOLOR

LV_EVE_CMD_GRADIENT

LV_EVE_CMD_INFLATE

LV_EVE_CMD_INTERRUPT

LV_EVE_CMD_KEYS

LV_EVE_CMD_LOADIDENTITY

LV_EVE_CMD_LOADIMAGE

LV_EVE_CMD_LOGO

LV_EVE_CMD_MEDIAFIFO

LV_EVE_CMD_MEMCPY

`LV_EVE_CMD_MEMCRC`

`LV_EVE_CMD_MEMSET`

`LV_EVE_CMD_MEMWRITE`

`LV_EVE_CMD_MEMZERO`

`LV_EVE_CMD_NUMBER`

`LV_EVE_CMD_PLAYVIDEO`

`LV_EVE_CMD_PROGRESS`

`LV_EVE_CMD_REGREAD`

`LV_EVE_CMD_ROMFONT`

`LV_EVE_CMD_ROTATE`

`LV_EVE_CMD_SCALE`

`LV_EVE_CMD_SCREENSAVER`

`LV_EVE_CMD_SCROLLBAR`

`LV_EVE_CMD_SETBASE`

`LV_EVE_CMD_SETBITMAP`

`LV_EVE_CMD_SETFONT`

`LV_EVE_CMD_SETFONT2`

`LV_EVE_CMD_SETMATRIX`

`LV_EVE_CMD_SETROTATE`

`LV_EVE_CMD_SETSCRATCH`

`LV_EVE_CMD_SKETCH`

`LV_EVE_CMD_SLIDER`

`LV_EVE_CMD_SNAPSHOT`

LV_EVE_CMD_SNAPSHOT2

LV_EVE_CMD_SPINNER

LV_EVE_CMD_STOP

LV_EVE_CMD_SWAP

LV_EVE_CMD_TEXT

LV_EVE_CMD_TOGGLE

LV_EVE_CMD_TRACK

LV_EVE_CMD_TRANSLATE

LV_EVE_CMD_VIDEOFRAME

LV_EVE_CMD_VIDEOSTART

LV_EVE_REG_ANA_COMP

LV_EVE_REG_BIST_EN

LV_EVE_REG_CLOCK

LV_EVE_REG_CMDB_SPACE

LV_EVE_REG_CMDB_WRITE

LV_EVE_REG_CMD_DL

LV_EVE_REG_CMD_READ

LV_EVE_REG_CMD_WRITE

LV_EVE_REG_CPURESET

LV_EVE_REG_CSPREAD

LV_EVE_REG_CTOUCH_EXTENDED

LV_EVE_REG_CTOUCH_TOUCH0_XY

LV_EVE_REG_CTOUCH_TOUCH4_X

`LV_EVE_REG_CTOUCH_TOUCH4_Y`

`LV_EVE_REG_CTOUCH_TOUCH1_XY`

`LV_EVE_REG_CTOUCH_TOUCH2_XY`

`LV_EVE_REG_CTOUCH_TOUCH3_XY`

`LV_EVE_REG_TOUCH_CONFIG`

`LV_EVE_REG_DATESTAMP`

`LV_EVE_REG_DITHER`

`LV_EVE_REG_DL_SWAP`

`LV_EVE_REG_FRAMES`

`LV_EVE_REG_FREQUENCY`

`LV_EVE_REG_GPIO`

`LV_EVE_REG_GPIOX`

`LV_EVE_REG_GPIOX_DIR`

`LV_EVE_REG_GPIO_DIR`

`LV_EVE_REG_HCYCLE`

`LV_EVE_REG_HOFFSET`

`LV_EVE_REG_HSIZE`

`LV_EVE_REG_HSYNC0`

`LV_EVE_REG_HSYNC1`

`LV_EVE_REG_ID`

`LV_EVE_REG_INT_EN`

`LV_EVE_REG_INT_FLAGS`

`LV_EVE_REG_INT_MASK`

LV_EVE_REG_MACRO_0

LV_EVE_REG_MACRO_1

LV_EVE_REG_MEDIAFIFO_READ

LV_EVE_REG_MEDIAFIFO_WRITE

LV_EVE_REG_OUTBITS

LV_EVE_REG_PCLK

LV_EVE_REG_PCLK_POL

LV_EVE_REG_PLAY

LV_EVE_REG_PLAYBACK_FORMAT

LV_EVE_REG_PLAYBACK_FREQ

LV_EVE_REG_PLAYBACK_LENGTH

LV_EVE_REG_PLAYBACK_LOOP

LV_EVE_REG_PLAYBACK_PLAY

LV_EVE_REG_PLAYBACK_READPTR

LV_EVE_REG_PLAYBACK_START

LV_EVE_REG_PWM_DUTY

LV_EVE_REG_PWM_HZ

LV_EVE_REG_RENDERMODE

LV_EVE_REG_ROTATE

LV_EVE_REG_SNAPFORMAT

LV_EVE_REG_SNAPSHOT

LV_EVE_REG_SNAPY

LV_EVE_REG_SOUND

LV_EVE_REG_SPI_WIDTH

LV_EVE_REG_SWIZZLE

LV_EVE_REG_TAG

LV_EVE_REG_TAG_X

LV_EVE_REG_TAG_Y

LV_EVE_REG_TAP_CRC

LV_EVE_REG_TAP_MASK

LV_EVE_REG_TOUCH_ADC_MODE

LV_EVE_REG_TOUCH_CHARGE

LV_EVE_REG_TOUCH_DIRECT_XY

LV_EVE_REG_TOUCH_DIRECT_Z1Z2

LV_EVE_REG_TOUCH_MODE

LV_EVE_REG_TOUCH_OVERSAMPLE

LV_EVE_REG_TOUCH_RAW_XY

LV_EVE_REG_TOUCH_RZ

LV_EVE_REG_TOUCH_RZTHRESH

LV_EVE_REG_TOUCH_SCREEN_XY

LV_EVE_REG_TOUCH_SETTLE

LV_EVE_REG_TOUCH_TAG

LV_EVE_REG_TOUCH_TAG1

LV_EVE_REG_TOUCH_TAG1_XY

LV_EVE_REG_TOUCH_TAG2

LV_EVE_REG_TOUCH_TAG2_XY

LV_EVE_REG_TOUCH_TAG3

LV_EVE_REG_TOUCH_TAG3_XY

LV_EVE_REG_TOUCH_TAG4

LV_EVE_REG_TOUCH_TAG4_XY

LV_EVE_REG_TOUCH_TAG_XY

LV_EVE_REG_TOUCH_TRANSFORM_A

LV_EVE_REG_TOUCH_TRANSFORM_B

LV_EVE_REG_TOUCH_TRANSFORM_C

LV_EVE_REG_TOUCH_TRANSFORM_D

LV_EVE_REG_TOUCH_TRANSFORM_E

LV_EVE_REG_TOUCH_TRANSFORM_F

LV_EVE_REG_TRACKER

LV_EVE_REG_TRACKER_1

LV_EVE_REG_TRACKER_2

LV_EVE_REG_TRACKER_3

LV_EVE_REG_TRACKER_4

LV_EVE_REG_TRIM

LV_EVE_REG_VCYCLE

LV_EVE_REG_VOFFSET

LV_EVE_REG_VOL_PB

LV_EVE_REG_VOL_SOUND

LV_EVE_REG_VSIZE

LV_EVE_REG_VSYNC0

`LV_EVE_REG_VSYNC1`

`LV_EVE_EVE_GLFORMAT`

`LV_EVE_DL_BITMAP_EXT_FORMAT`

`LV_EVE_DL_BITMAP_SWIZZLE`

`LV_EVE_EVE_ASTC_4X4`

`LV_EVE_EVE_ASTC_5X4`

`LV_EVE_EVE_ASTC_5X5`

`LV_EVE_EVE_ASTC_6X5`

`LV_EVE_EVE_ASTC_6X6`

`LV_EVE_EVE_ASTC_8X5`

`LV_EVE_EVE_ASTC_8X6`

`LV_EVE_EVE_ASTC_8X8`

`LV_EVE_EVE_ASTC_10X5`

`LV_EVE_EVE_ASTC_10X6`

`LV_EVE_EVE_ASTC_10X8`

`LV_EVE_EVE_ASTC_10X10`

`LV_EVE_EVE_ASTC_12X10`

`LV_EVE_EVE_ASTC_12X12`

`LV_EVE_EVE_RAM_ERR_REPORT`

`LV_EVE_EVE_RAM_FLASH`

`LV_EVE_EVE_RAM_FLASH_POSTBLOB`

`LV_EVE_EVE_OPT_FLASH`

`LV_EVE_EVE_OPT_OVERLAY`

LV_EVE_EVE_OPT_FORMAT

LV_EVE_EVE_OPT_FILL

LV_EVE_CMD_BITMAP_TRANSFORM

LV_EVE_CMD_SYNC

LV_EVE_CMD_FLASHERASE

LV_EVE_CMD_FLASHWRITE

LV_EVE_CMD_FLASHREAD

LV_EVE_CMD_FLASHUPDATE

LV_EVE_CMD_FLASHDETACH

LV_EVE_CMD_FLASHATTACH

LV_EVE_CMD_FLASHFAST

LV_EVE_CMD_FLASHSPIDESEL

LV_EVE_CMD_FLASHSPITX

LV_EVE_CMD_FLASHSPIRX

LV_EVE_CMD_FLASHSOURCE

LV_EVE_CMD_CLEARCACHE

LV_EVE_CMD_INFLATE2

LV_EVE_CMD_ROTATEAROUND

LV_EVE_CMD_RESETFONTS

LV_EVE_CMD_ANIMSTART

LV_EVE_CMD_ANIMSTOP

LV_EVE_CMD_ANIMXY

LV_EVE_CMD_ANIMDRAW

`LV_EVE_CMD_GRADIENTA`

`LV_EVE_CMD_FILLWIDTH`

`LV_EVE_CMD_APPENDF`

`LV_EVE_CMD_ANIMFRAME`

`LV_EVE_CMD_VIDEOSTARTF`

`LV_EVE_REG_ADAPTIVE_FRAMERATE`

`LV_EVE_REG_PLAYBACK_PAUSE`

`LV_EVE_REG_FLASH_STATUS`

`LV_EVE_REG_FLASH_SIZE`

`LV_EVE_REG_PLAY_CONTROL`

`LV_EVE_REG_COPRO_PATCH_PTR`

`LV_EVE_CMD_ANIMFRAMERAM`

`LV_EVE_CMD_ANIMSTARTRAM`

`LV_EVE_CMD_APILEVEL`

`LV_EVE_CMD_CALIBRATESUB`

`LV_EVE_CMD_CALLLIST`

`LV_EVE_CMD_ENDLIST`

`LV_EVE_CMD_FLASHPROGRAM`

`LV_EVE_CMD_FONTCACHE`

`LV_EVE_CMD_FONTCACHEQUERY`

`LV_EVE_CMD_GETIMAGE`

`LV_EVE_CMD_HSF`

`LV_EVE_CMD_LINETIME`

LV_EVE_CMD_NEWTLIST**LV_EVE_CMD_PCLKFREQ****LV_EVE_CMD_RETURN****LV_EVE_CMD_RUNANIM****LV_EVE_CMD_TESTCARD****LV_EVE_CMD_WAIT****LV_EVE_REG_UNDERRUN****LV_EVE_REG_AH_HCYCLE_MAX****LV_EVE_REG_PCLK_FREQ****LV_EVE_REG_PCLK_2X****LV_EVE_REG_ANIM_ACTIVE**

Funzioni

static inline uint32_t **LV_EVE_ALPHA_FUNC**(uint8_t func, uint8_t ref)

Set the alpha test function.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t **LV_EVE_BITMAP_HANDLE**(uint8_t handle)

Set the bitmap handle.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t **LV_EVE_BITMAP_LAYOUT**(uint8_t format, uint16_t linestride, uint16_t height)

Set the source bitmap memory format and layout for the current handle.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t **LV_EVE_BITMAP_SIZE**(uint8_t filter, uint8_t wrapx, uint8_t wrapy, uint16_t width, uint16_t height)

Set the source bitmap memory format and layout for the current handle.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t **LV_EVE_BITMAP_LAYOUT_H**(uint16_t linestride, uint16_t height)

Set the 2 most significant bits of the source bitmap memory format and layout for the current handle.

Nota

this is different to FTDIs implementation as this takes the original values as parameters and not only the upper bits

Parametri

- **linestride** -- 12-bit value specified to BITMAP_LAYOUT
- **height** -- 11-bit value specified to BITMAP_LAYOUT

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_BITMAP_SIZE_H(uint16_t width, uint16_t height)

Set the 2 most significant bits of bitmaps dimension for the current handle.

Nota

this is different to FTDIs implementation as this takes the original values as parameters and not only the upper bits

Parametri

- **linestride** -- 11-bit value of bitmap width, the 2 most significant bits are used
- **height** -- 11-bit value of bitmap width, the 2 most significant bits are used

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_BITMAP_SOURCE(uint32_t addr)

Set the source address of bitmap data in RAM_G or flash memory.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_BITMAP_TRANSFORM_C(uint32_t val)

Set the C coefficient of the bitmap transform matrix.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_BITMAP_TRANSFORM_F(uint32_t val)

Set the F coefficient of the bitmap transform matrix.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_BLEND_FUNC(uint8_t src, uint8_t dst)

Execute a sequence of commands at another location in the display list.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_CALL(uint16_t dest)

Execute a sequence of commands at another location in the display list.

Nota

valid range for dest is from zero to 2047

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_JUMP(uint16_t dest)

Execute commands at another location in the display list.

Nota

valid range for dest is from zero to 2047

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_CELL(uint8_t cell)

Set the bitmap cell number for the VERTEX2F command.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_CLEAR(uint8_t color, uint8_t stencil, uint8_t tag)

Clear buffers to preset values.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_CLEAR_COLOR_A(uint8_t alpha)

Set clear value for the alpha channel.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_CLEAR_COLOR_RGB(uint8_t red, uint8_t green, uint8_t blue)

Set clear values for red, green and blue channels.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_CLEAR_STENCIL(uint8_t val)

Set clear value for the stencil buffer.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_CLEAR_TAG(uint8_t val)

Set clear value for the tag buffer.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_COLOR_A(uint8_t alpha)

Set the current color alpha.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_COLOR_MASK(uint8_t red, uint8_t green, uint8_t blue, uint8_t alpha)

Enable or disable writing of color components.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_COLOR_RGB(uint8_t red, uint8_t green, uint8_t blue)

Set the current color red, green and blue.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_LINE_WIDTH(uint16_t width)

Set the width of lines to be drawn with primitive LINES in 1/16 pixel precision.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_MACRO(uint8_t macro)

Execute a single command from a macro register.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_PALETTE_SOURCE(uint32_t addr)

Set the base address of the palette.

Nota

2-byte alignment is required if pixel format is PALETTE4444 or PALETTE565.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_POINT_SIZE(uint16_t size)

Set the radius of points to be drawn with primitive POINTS in 1/16 pixel precision.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_SCISSOR_SIZE(uint16_t width, uint16_t height)

Set the size of the scissor clip rectangle.

Nota

valid range for width and height is from zero to 2048

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_SCISSOR_XY(uint16_t xc0, uint16_t yc0)

Set the top left corner of the scissor clip rectangle.

Nota

valid range for width and height is from zero to 2047

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_STENCIL_FUNC(uint8_t func, uint8_t ref, uint8_t mask)

Set function and reference value for stencil testing.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_STENCIL_MASK(uint8_t mask)

Control the writing of individual bits in the stencil planes.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_STENCIL_OP(uint8_t sfail, uint8_t spass)

Set stencil test actions.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_TAG(uint8_t tagval)

Attach the tag value for the following graphics objects drawn on the screen.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t LV_EVE_TAG_MASK(uint8_t mask)

Control the writing of the tag buffer.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t **LV_EVE_VERTEX2F**(int16_t xc0, int16_t yc0)

Set coordinates for graphics primitives.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t **LV_EVE_VERTEX2II**(uint16_t xc0, uint16_t yc0, uint8_t handle, uint8_t cell)

Set coordinates, bitmap-handle and cell-number for graphics primitives.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t **LV_EVE_VERTEX_FORMAT**(uint8_t frac)

Set the precision of VERTEX2F coordinates.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t **LV_EVE_VERTEX_TRANSLATE_X**(int32_t xco)

Set the vertex transformations X translation component.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t **LV_EVE_VERTEX_TRANSLATE_Y**(int32_t yco)

Set the vertex transformations Y translation component.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t **LV_EVE_BITMAP_EXT_FORMAT**(uint16_t format)

Set the extended format of the bitmap.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t **LV_EVE_BITMAP_SWIZZLE**(uint8_t red, uint8_t green, uint8_t blue, uint8_t alpha)

Set the source for the red, green, blue and alpha channels of a bitmap.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t **LV_EVE_BITMAP_TRANSFORM_A**(uint8_t prc, uint32_t val)

Set the A coefficient of the bitmap transform matrix.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t **LV_EVE_BITMAP_TRANSFORM_B**(uint8_t prc, uint32_t val)

Set the B coefficient of the bitmap transform matrix.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t **LV_EVE_BITMAP_TRANSFORM_D**(uint8_t prc, uint32_t val)

Set the D coefficient of the bitmap transform matrix.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

static inline uint32_t **LV_EVE_BITMAP_TRANSFORM_E**(uint8_t prc, uint32_t val)

Set the E coefficient of the bitmap transform matrix.

Ritorna

a 32 bit word for use with EVE_cmd_dl()

14.16.4 evdev

lv_evdev.h

Typedef

```
typedef void (*lv_evdev_discovery_cb_t)(lv_indev_t *indev, lv_evdev_type_t type, void *user_data)
```

Param indev

the indev created for the newly discovered evdev

Param type

the type of the evdev

Param user_data

a custom parameter

Enum

```
enum lv_evdev_type_t
```

Values:

enumerator **LV_EVDEV_TYPE_REL**

mice

enumerator **LV_EVDEV_TYPE_ABS**

touch screens, mousepads

enumerator **LV_EVDEV_TYPE_KEY**

keyboards, keypads, buttons

Funzioni

```
lv_indev_t *lv_evdev_create(lv_evdev_type_t indev_type, const char *dev_path)
```

Create evdev input device from a given path.

Parametri

- **type** -- LV_INDEV_TYPE_POINTER or LV_INDEV_TYPE_KEYPAD
- **dev_path** -- device path, e.g., /dev/input/event0

Ritorna

pointer to input device or NULL if opening failed

```
lv_indev_t *lv_evdev_create_fd(lv_evdev_type_t indev_type, int fd)
```

Create evdev input device, taking ownership of the given file descriptor.

Parametri

- **type** -- LV_INDEV_TYPE_POINTER or LV_INDEV_TYPE_KEYPAD
- **fd** -- file descriptor of the evdev device

Ritorna

pointer to input device or NULL if opening failed

```
lv_result_t lv_evdev_discovery_start(lv_evdev_discovery_cb_t cb, void *user_data)
```

Begin automatically creating evdev indevs for all new and existing evdev devices found in /dev/input/

Parametri

- **cb** -- function to call when a new evdev indev is discovered, or NULL
- **user_data** -- parameter to pass to the callback

Ritorna

the success or failure status. It will fail if it's already running or resources could not be initialized.

```
lv_result_t lv_evdev_discovery_stop(void)
```

Stop automatically creating evdev indevs. Safe to call from the discovery callback.

Ritorna

the success or failure status. It will fail if it's already running.

`void lv_evdev_set_swap_axes(lv_indev_t *indev, bool swap_axes)`

Set whether coordinates of pointer device should be swapped. Defaults to false.

Parametri

- **indev** -- evdev input device
- **swap_axes** -- whether to swap x and y axes

`void lv_evdev_set_calibration(lv_indev_t *indev, int min_x, int min_y, int max_x, int max_y)`

Configure a coordinate transformation for pointer devices. Applied after axis swap, if any. Defaults to apply no transformation.

Parametri

- **indev** -- evdev input device
- **min_x** -- pointer coordinate mapped to min x of display
- **min_y** -- pointer coordinate mapped to min y of display
- **max_x** -- pointer coordinate mapped to max x of display
- **max_y** -- pointer coordinate mapped to max y of display

`void lv_evdev_delete(lv_indev_t *indev)`

Remove evdev input device.

Parametri

indev -- evdev input device to close and free

lv_evdev_private.h

Funzioni

`void lv_evdev_deinit(void)`

14.16.5 libinput

lv_libinput.h

Define

`LV_LIBINPUT_MAX_EVENTS`

Enum

`enum lv_libinput_capability`

Values:

enumerator `LV_LIBINPUT_CAPABILITY_NONE`

enumerator `LV_LIBINPUT_CAPABILITY_KEYBOARD`

enumerator `LV_LIBINPUT_CAPABILITY_POINTER`

enumerator `LV_LIBINPUT_CAPABILITY_TOUCH`

Funzioni

`lv_libinput_capability lv_libinput_query_capability(struct libinput_device *device)`

Determine the capabilities of a specific libinput device.

Parametri

device -- the libinput device to query

Ritorna

the supported input capabilities

```
char *lv_libinput_find_dev(lv_libinput_capability capabilities, bool force_rescan)
```

Find connected input device with specific capabilities

Parametri

- **capabilities** -- required device capabilities
- **force_rescan** -- erase the device cache (if any) and rescan the file system for available devices

Ritorna

device node path (e.g. /dev/input/event0) for the first matching device or NULL if no device was found. The pointer is safe to use until the next forceful device search.

```
size_t lv_libinput_find_devs(lv_libinput_capability capabilities, char **found, size_t count, bool force_rescan)
```

Find connected input devices with specific capabilities

Parametri

- **capabilities** -- required device capabilities
- **devices** -- pre-allocated array to store the found device node paths (e.g. /dev/input/event0). The pointers are safe to use until the next forceful device search.
- **count** -- maximum number of devices to find (the devices array should be at least this long)
- **force_rescan** -- erase the device cache (if any) and rescan the file system for available devices

Ritorna

number of devices that were found

```
lv_indev_t *lv_libinput_create(lv_indev_type_t indev_type, const char *dev_path)
```

Create a new libinput input device

Parametri

- **type** -- LV_INDEV_TYPE_POINTER or LV_INDEV_TYPE_KEYPAD
- **dev_path** -- device path, e.g. /dev/input/event0

Ritorna

pointer to input device or NULL if opening failed

```
void lv_libinput_delete(lv_indev_t *indev)
```

Delete a libinput input device

Parametri

indev -- pointer to input device

lv_libinput_private.h

```
struct _lv_libinput_event_t
```

Membri Pubblici

lv_indev_state_t **pressed**

int **key_val**

lv_point_t **point**

```
struct _lv_libinput_t
```

Membri Pubblici

int **fd**

```

struct pollfd fds[1]

lv_libinput_event_t points[32]

lv_libinput_event_t slots[2]

lv_point_t pointer_position

bool pointer_button_down

int start

int end

lv_libinput_event_t last_event

bool deinit

pthread_mutex_t event_lock

pthread_t worker_thread

struct libinput *libinput_context

struct libinput_device *libinput_device

```

lv_xkb.h

lv_xkb_private.h

14.16.6 nuttx

lv_nuttx_cache.h

Funzioni

void **lv_nuttx_cache_init**(void)

void **lv_nuttx_cache_deinit**(void)

lv_nuttx_entry.h

Typedef

typedef struct *lv_nuttx_ctx_t* **lv_nuttx_ctx_t**

Funzioni

void **lv_nuttx_dsc_init**(*lv_nuttx_dsc_t* *dsc)

Initialize the *lv_nuttx_dsc_t* structure with default values for the NuttX port of LVGL.

Parametri

dsc -- Pointer to the *lv_nuttx_dsc_t* structure to be initialized.

```
void lv_nuttx_init(const lv_nuttx_dsc_t *dsc, lv_nuttx_result_t *result)
Initialize the LVGL display driver for NuttX using the provided configuration information.

Parametri

- dsc -- Pointer to the lv_nuttx_dsc_t structure containing the configuration information for the display driver.
- result -- Pointer to the lv_nuttx_result_t structure containing display and input device handler.



void lv_nuttx_deinit(lv_nuttx_result_t *result)
Deinitialize the LVGL display driver for NuttX.

Parametri

- result -- Pointer to the lv_nuttx_result_t structure containing display and input device handler.



void lv_nuttx_init_custom(const lv_nuttx_dsc_t *dsc, lv_nuttx_result_t *result)
Initialize the LVGL display driver for NuttX using the provided custom configuration information.

Parametri

- dsc -- Pointer to the lv_nuttx_dsc_t structure containing the custom configuration for the display driver.
- result -- Pointer to the lv_nuttx_result_t structure containing display and input device handler.



void lv_nuttx_deinit_custom(lv_nuttx_result_t *result)
Deinitialize the LVGL display driver for NuttX using the provided custom configuration information.

Parametri

- result -- Pointer to the lv_nuttx_result_t structure containing display and input device handler.



void lv_nuttx_run(lv_nuttx_result_t *result)
Call lv_timer_handler() (LVGL's super loop) in an endless loop. If LV_USE_NUTTX_LIBUV is enabled an UV timer will be created, else lv_timer_handler() will be called in a loop with some sleep.

Parametri

- result -- pointer to a variable initialized by lv_nuttx_init() or lv_nuttx_init_custom()



uint32_t lv_nuttx_get_idle(void)
Get the idle percentage of the system.

Ritorna
The idle percentage of the system.

struct lv_nuttx_dsc_t

Membri Pubblici

const char *fb_path

const char *input_path

const char *utouch_path

const char *mouse_path

const char *trace_path

struct lv_nuttx_result_t
```

Membri Pubblici

```
lv_display_t *disp  
  
lv_indev_t *indev  
  
lv_indev_t *utouch_indev  
  
lv_indev_t *mouse_indev  
  
struct lv_nuttx_ctx_t
```

Membri Pubblici

```
void *image_cache  
  
int trace_fd
```

lv_nuttx_fbdev.h**Funzioni**

```
lv_display_t *lv_nuttx_fbdev_create(void)  
Create a new display with NuttX backend.  
  
int lv_nuttx_fbdev_set_file(lv_display_t *disp, const char *file)
```

Initialize display with specified framebuffer device

Parametri

- **disp** -- pointer to display with NuttX backend
- **file** -- the name of framebuffer device

lv_nuttx_image_cache.h**Funzioni**

```
void lv_nuttx_image_cache_init(bool use_independent_image_heap)  
void lv_nuttx_image_cache_deinit(void)
```

lv_nuttx_lcd.h**Funzioni**

```
lv_display_t *lv_nuttx_lcd_create(const char *dev_path)
```

lv_nuttx_libuv.h**Funzioni**

```
void *lv_nuttx_uv_init(lv_nuttx_uv_t *uv_info)  
Initialize the uv_loop using the provided configuration information.  
Parametri  
uv_info -- Pointer to the lv_nuttx_uv_t structure to be initialized.
```

```
void lv_nuttx_uv_deinit(void **data)
    Deinitialize the uv_loop configuration for NuttX porting layer.
Parametri
    data -- Pointer to user data.
```

struct **lv_nuttx_uv_t**

Membri Pubblici

```
void *loop
lv_display_t *disp
lv_indev_t *indev
```

lv_nuttx_mouse.h

Funzioni

```
lv_indev_t *lv_nuttx_mouse_create(const char *dev_path)
    Initialize indev with specified input device.
Parametri
    dev_path -- path of input device
```

lv_nuttx_profiler.h

Funzioni

```
void lv_nuttx_profiler_init(void)
void lv_nuttx_profiler_set_file(const char *file)
void lv_nuttx_profiler_deinit(void)
```

lv_nuttx_touchscreen.h

Funzioni

```
lv_indev_t *lv_nuttx_touchscreen_create(const char *dev_path)
    Initialize indev with specified input device.
Parametri
    dev_path -- path of input device
```

14.16.7 opengles

lv_opengles_debug.h

Define

GL_CALL(x)

Funzioni

```
void GLClearError(void)
void GLLogCall(const char *function, const char *file, int line)
```

lv_opengles_driver.h**Funzioni**

`void lv_opengles_init(void)`

Initialize OpenGL

 **Nota**

it is not necessary to call this if you use `lv_opengles_glfw_window_create`

`void lv_opengles_deinit(void)`

Deinitialize OpenGL

 **Nota**

it is not necessary to call this if you use `lv_opengles_glfw_window_create`

`void lv_opengles_render_texture(unsigned int texture, const lv_area_t *texture_area, lv_opa_t opa, int32_t disp_w, int32_t disp_h, const lv_area_t *texture_clip_area, bool h_flip, bool v_flip)`

Render a texture using alternate blending mode for smoother translucent materials and correct anti-aliasing of glTF elements when using transparent background

Parametri

- **texture** -- OpenGL texture ID
- **texture_area** -- the area in the window to render the texture in
- **opa** -- opacity to blend the texture with existing contents
- **disp_w** -- width of the window/framebuffer being rendered to
- **disp_h** -- height of the window/framebuffer being rendered to
- **h_flip** -- horizontal flip
- **v_flip** -- vertical flip

`void lv_opengles_render_display_texture(lv_display_t *display, bool h_flip, bool v_flip)`

Render a display texture - Supports rotation

Parametri

- **display** -- LVGL Texture display. Created with the `lv_opengles_texture` module
- **h_flip** -- horizontal flip
- **v_flip** -- vertical flip

`void lv_opengles_render_fill(lv_color_t color, const lv_area_t *area, lv_opa_t opa, int32_t disp_w, int32_t disp_h)`

Render a fill

Parametri

- **color** -- the color of the fill
- **area** -- the area in the window to render the fill
- **opa** -- opacity to blend the fill with existing contents
- **disp_w** -- width of the window/framebuffer being rendered to
- **disp_h** -- height of the window/framebuffer being rendered to

`void lv_opengles_render_clear(void)`

Clear the window/display

`void lv_opengles_viewport(int32_t x, int32_t y, int32_t w, int32_t h)`

Set the OpenGL viewport

Parametri

- **x** -- x position of the viewport
- **y** -- y position of the viewport

- **w** -- width of the viewport
- **h** -- height of the viewport

lv_opengles_egl.h**lv_opengles_egl_private.h****lv_opengles_glfw.h****Funzioni**

```
lv_opengles_window_t *lv_opengles_glfw_window_create(int32_t hor_res, int32_t ver_res, bool
                                                    use_mouse_indev)
```

Create a GLFW OpenGL window with no textures and initialize OpenGL

Parametri

- **hor_res** -- width in pixels of the window
- **ver_res** -- height in pixels of the window
- **use_mouse_indev** -- send pointer indev input to LVGL display textures

Ritorna

the new GLFW OpenGL window handle

```
lv_opengles_window_t *lv_opengles_glfw_window_create_ex(int32_t hor_res, int32_t ver_res, bool
                                                       use_mouse_indev, bool h_flip, bool v_flip, const
                                                       char *title)
```

Create a GLFW window with no textures and initialize OpenGL

Parametri

- **hor_res** -- width in pixels of the window
- **ver_res** -- height in pixels of the window
- **use_mouse_indev** -- send pointer indev input to LVGL display textures
- **h_flip** -- Should the window contents be horizontally mirrored?
- **v_flip** -- Should the window contents be vertically mirrored?
- **title** -- The window title

Ritorna

the new GLFW window handle

```
void lv_opengles_glfw_window_set_title(lv_opengles_window_t *window, const char *new_title)
```

Set the window's title text

Parametri

- **window** -- GLFW window to configure
- **new_title** -- The new title text

```
void lv_opengles_glfw_window_set_flip(lv_opengles_window_t *window, bool h_flip, bool v_flip)
```

Set the horizontal / vertical flipping of a GLFW window

Parametri

- **window** -- GLFW window to configure
- **h_flip** -- Should the window contents be horizontally mirrored?
- **v_flip** -- Should the window contents be vertically mirrored?

```
void *lv_opengles_glfw_window_get_glfw_window(lv_opengles_window_t *window)
```

Get the GLFW window handle for a GLFW lv_opengles_window_t

Parametri

window -- GLFW window to return the handle of

Ritorna

the GLFW window handle

lv_opengles_private.h**Define****GL_BGRA**

GL_TEXTURE_MAX_LEVEL**GL_UNPACK_ROW_LENGTH****glGenVertexArrays****glBindVertexArray****glDeleteVertexArrays****glTexStorage2D****GL_RGBA32F****GL_NUM_EXTENSIONS****GL_RGB8****GL_RGBA8****LV_GL_PREFERRED_DEPTH**

Funzioni

```
void lv_opengles_render(unsigned int texture, const lv_area_t *texture_area, lv_opa_t opa, int32_t disp_w,
                        int32_t disp_h, const lv_area_t *texture_clip_area, bool h_flip, bool v_flip, lv_color_t
                        fill_color, bool blend_opt)
```

lv_opengles_texture.h

Funzioni

`lv_display_t *lv_opengles_texture_create(int32_t w, int32_t h)`

Create a display that flushes to an OpenGL texture If you already have a texture and want to bind it to the display, see `lv_opengles_texture_create_from_texture_id`

Parametri

- **w** -- width in pixels of the texture
- **h** -- height in pixels of the texture

Ritorna

the new display or NULL on failure

`lv_display_t *lv_opengles_texture_create_from_texture_id(int32_t w, int32_t h, unsigned int texture_id)`

Create a display that flushes to the provided OpenGL texture If you don't have a texture to bind it to the display, see `lv_opengles_texture_create`

Parametri

- **w** -- width in pixels of the texture
- **h** -- height in pixels of the texture
- **texture_id** -- the texture LVGL will render to

Ritorna

the new display or NULL on failure

unsigned int **lv_opengles_texture_get_texture_id**(lv_display_t *disp)

Get the OpenGL texture ID of the display

Parametri

disp -- display

Ritorna

texture ID

lv_display_t ***lv_opengles_texture_get_from_texture_id**(unsigned int texture_id)

Get the display of an OpenGL texture if it is associated with one

Parametri

texture_id -- OpenGL texture ID

Ritorna

display or NULL if there no display with that texture ID

lv_opengles_texture_private.h

Funzioni

lv_result_t **lv_opengles_texture_create_draw_buffers**(lv_opengles_texture_t *texture, lv_display_t *display)

void **lv_opengles_texture_reshape**(lv_display_t *disp, int32_t width, int32_t height)

void **lv_opengles_texture_deinit**(lv_opengles_texture_t *texture)

struct **lv_opengles_texture_t**

Membri Pubblici

unsigned int **texture_id**

bool **is_texture_owner**

LV_USE_DRAW_OPENGLS

lv_opengles_window.h

Funzioni

void **lv_opengles_window_delete**(lv_opengles_window_t *window)

Delete an OpenGL window. If it is the last one, the process will exit

Parametri

window -- OpenGL window to delete

lv_opengles_window_texture_t ***lv_opengles_window_add_texture**(lv_opengles_window_t *window, unsigned int texture_id, int32_t w, int32_t h)

Add a texture to the OpenGL window. It can be an LVGL display texture, or any OpenGL texture

Parametri

- **window** -- OpenGL window
- **texture_id** -- OpenGL texture ID
- **w** -- width in pixels of the texture
- **h** -- height in pixels of the texture

Ritorna

the new texture handle

lv_display_t ***lv_opengles_window_display_create**(lv_opengles_window_t *window, int32_t w, int32_t h)

lv_opengles_window_texture_t ***lv_opengles_window_display_get_window_texture**(lv_display_t *window_display)

```
void lv.opengles_window_texture_remove(lv_opengles_window_texture_t *texture)
    Remove a texture from its OpenGL window and delete it
Parametri
    texture -- handle of an OpenGL window texture

void lv.opengles_window_texture_set_x(lv_opengles_window_texture_t *texture, int32_t x)
    Set the x position of a texture within its OpenGL window
Parametri
    • texture -- handle of an OpenGL window texture
    • x -- new x position of the texture

void lv.opengles_window_texture_set_y(lv_opengles_window_texture_t *texture, int32_t y)
    Set the y position of a texture within its OpenGL window
Parametri
    • texture -- handle of an OpenGL window texture
    • y -- new y position of the texture

void lv.opengles_window_texture_set_opa(lv_opengles_window_texture_t *texture, lv_opa_t opa)
    Set the opacity of a texture in an OpenGL window
Parametri
    • texture -- handle of an OpenGL window texture
    • opa -- new opacity of the texture

lv_indev_t *lv.opengles_window_texture_get_mouse_indev(lv_opengles_window_texture_t *texture)
    Get the mouse indev associated with a texture in an OpenGL window, if it exists
```

Nota

there will only be an indev if the texture is based on an LVGL display texture and the window was created with `use_mouse_indev` as `true`

Parametri

texture -- handle of an OpenGL window texture

Ritorna

the indev or NULL

assets

lv.opengles_shader.h

Funzioni

```
char *lv.opengles_shader_get_vertex(lv.opengl_gsl_version version)
char *lv.opengles_shader_get_fragment(lv.opengl_gsl_version version)
void lv.opengles_shader_get_source(lv.opengl_shader_portions_t *portions, lv.opengl_gsl_version version)
```

opengl_shader

lv.opengl_shader_internal.h

TypeDef

```
typedef struct lv.opengl_shader_cache_struct lv.opengl_shader_manager_t
```

```
typedef void (*lv.opengl_shader_program_update_uniformi_t)(struct lv.shader_program*, const char*, int)
```

```
typedef void (*lv.opengl.shader.program.update_uniformf_t)(struct _lv_shader_program*, const char*, float)
```

```
typedef struct _lv_shader_program lv.opengl.shader.program_t
```

Enum

```
enum lv.opengl.gsl.version
```

Values:

```
enumerator LV_OPENGL_GLSL_VERSION_300ES
```

```
enumerator LV_OPENGL_GLSL_VERSION_100
```

```
enumerator LV_OPENGL_GLSL_VERSION_LAST
```

Funzioni

```
char *lv.opengl.shader.manager_process_includes(const char *c_src, const lv.opengl.shader_t
                                                *src_includes, size_t num_items)
```

```
lv.opengl.shader.program_t *lv.opengl.shader.program_create(uint32_t id)
```

```
void lv.opengl.shader.program_destroy(lv.opengl.shader.program_t *program)
```

```
GLuint lv.opengl.shader.program_get_id(lv.opengl.shader.program_t *program)
```

```
void lv.opengl.shader.manager_init(lv.opengl.shader.manager_t *manager, const lv.opengl.shader_t
                                    *sources, size_t len, const char *vert_src, const char *frag_src)
```

```
void lv.opengl.shader.manager_deinit(lv.opengl.shader.manager_t *manager)
```

```
uint32_t lv.opengl.shader.hash(const char *value)
```

```
GLuint lv.opengl.shader.manager_get_texture(lv.opengl.shader.manager_t *manager, uint32_t hash)
```

```
void lv.opengl.shader.manager_store_texture(lv.opengl.shader.manager_t *manager, uint32_t hash,
                                            GLuint id)
```

```
lv_result_t lv.opengl.shader.manager_select_shader(lv.opengl.shader.manager_t *shader, const char
                                                   *shader_identifier, const lv.opengl.shader.define_t
                                                   *permutations, size_t permutations_len,
                                                   lv.opengl.gsl.version gsl_version, uint32_t *out_hash)
```

```
lv.opengl.shader.program_t *lv.opengl.shader.manager_get_program(lv.opengl.shader.manager_t
                                                               *manager, uint32_t
                                                               fragment_shader_hash, uint32_t
                                                               vertex_shader_hash)
```

```
const char *lv.opengles_gsl_version_to_string(lv.opengl.gsl.version version)
```

```
struct lv.opengl.shader_t
```

Membri Pubblici`const char *name``const char *source``struct lv_opengl_shader_portions_t`**Membri Pubblici**`const lv_opengl_shader_t *all``uint32_t count``struct lv_opengl_shader_define_t`**Membri Pubblici**`const char *name``const char *value``bool value_allocated``struct lv_opengl_shader_source_t`**Membri Pubblici**`lv_opengl_shader_t data``bool src_allocated``struct lv_opengl_compiled_shader_t`**Membri Pubblici**`uint32_t hash``GLuint id``struct lv_opengl_shader_cache_struct`**Membri Pubblici**`lv_rb_t sources_map``lv_rb_t textures_map`

```
lv_rb_t compiled_shaders_map

lv_rb_t programs_map

GLuint bg_index_buf

GLuint bg_vertex_buf

GLuint bg_program

GLuint bg_vao

struct _lv_shader_program
```

Membri Pubblici

lv_opengl_shader_program_update_uniformi_t **update_uniform_li**

lv_opengl_shader_program_update_uniformf_t **update_uniform lf**

uint32_t **id**

14.16.8 qnx

lv_qnx.h

LVGL driver for the QNX Screen compositing window manager.

Funzioni

lv_display_t *lv_qnx_window_create(int32_t hor_res, int32_t ver_res)

Create a window to use as a display for LVGL.

Parametri

- **hor_res** -- The horizontal resolution (size) of the window
- **ver_res** -- The vertical resolution (size) of the window

Ritorna

A pointer to a new display object if successful, NULL otherwise

void lv_qnx_window_set_title(lv_display_t *disp, const char *title)

Set the title of the window identified by the given display.

Parametri

- **disp** -- The display object for the window
- **title** -- The new title to set

bool lv_qnx_add_pointer_device(lv_display_t *disp)

Create a pointer input device for the display. Only one pointer object is currently supported.

Parametri

disp -- The display object associated with the device

Ritorna

true if successful, false otherwise

```
bool lv_qnx_add_keyboard_device(lv_display_t *disp)
```

Create a keyboard input device for the display. Only one keyboard object is currently supported.

Parametri

disp -- The display object associated with the device

Ritorna

true if successful, false otherwise

```
int lv_qnx_event_loop(lv_display_t *disp)
```

Runs the event loop for the display. The function only returns in response to a close event.

Parametri

disp -- The display for the event loop

Ritorna

Exit code

14.16.9 sdl

lv_sdl_keyboard.h

Define

KEYBOARD_BUFFER_SIZE

Funzioni

```
lv_indev_t *lv_sdl_keyboard_create(void)
```

lv_sdl_mouse.h

Funzioni

```
lv_indev_t *lv_sdl_mouse_create(void)
```

lv_sdl_mousewheel.h

Funzioni

```
lv_indev_t *lv_sdl_mousewheel_create(void)
```

lv_sdl_private.h

Funzioni

```
void lv_sdl_keyboard_handler(SDL_Event *event)
```

```
void lv_sdl_mouse_handler(SDL_Event *event)
```

```
void lv_sdl_mousewheel_handler(SDL_Event *event)
```

```
lv_display_t *lv_sdl_get_disp_from_win_id(uint32_t win_id)
```

lv_sdl_window.h

Define

LV SDL MOUSEWHEEL MODE ENCODER

Modified by NXP in 2025

LV SDL MOUSEWHEEL MODE CROWN

Funzioni

```
lv_display_t *lv_sdl_window_create(int32_t hor_res, int32_t ver_res)
void lv_sdl_window_set_resizeable(lv_display_t *disp, bool value)
void lv_sdl_window_set_size(lv_display_t *disp, int32_t hor_res, int32_t ver_res)
void lv_sdl_window_set_zoom(lv_display_t *disp, float zoom)
float lv_sdl_window_get_zoom(lv_display_t *disp)
void lv_sdl_window_set_title(lv_display_t *disp, const char *title)
void lv_sdl_window_set_icon(lv_display_t *disp, void *icon, int32_t width, int32_t height)
void *lv_sdl_window_get_renderer(lv_display_t *disp)
void lv_sdl_quit(void)
struct SDL_Window *lv_sdl_window_get_window(lv_display_t *disp)
```

14.16.10 uefi

[lv_uefi.h](#)

[lv_uefi_context.h](#)

Funzioni

```
void lv_uefi_init(EFI_HANDLE image_handle, EFI_SYSTEM_TABLE *system_table)
```

Initialize the UEFI cache variables.

Remark

This has to be called before *lv_init()*.

Parametri

- **image_handle** -- The handle of the current image
- **system_table** -- Pointer to the system table

```
void lv_uefi_platform_init(void)
```

Initialize the LVGL UEFI backend.

Remark

This is a private API which is used for LVGL UEFI backend implementation. LVGL users shouldn't use that because the LVGL has already used it in *lv_init*.

```
void lv_uefi_platform_deinit(void)
```

Cleanup the LVGL UEFI backend.

Remark

This is a private API which is used for LVGL UEFI backend implementation. LVGL users shouldn't use that because the LVGL has already used it in *lv_deinit*.

lv_uefi_display.h**Funzioni**

`lv_display_t *lv_uefi_display_create(void *handle)`

Create a LVGL display object.

Parametri

`handle` -- The handle on which an instance of the EFI_GRAPHICS_OUTPUT_PROTOCOL_COL protocol is installed.

Ritorna

The created LVGL display object.

`void *lv_uefi_display_get_active(void)`

Try to find the active display handle.

 **Remark**

The active display need interfaces for EFI_GRAPHICS_OUTPUT_PROTOCOL and EFI_EDID_ACTIVE_PROTOCOL

Ritorna

The handle or NULL if not found.

`void *lv_uefi_display_get_any(void)`

Try to find any display handle.

Ritorna

The handle or NULL if not found.

lv_uefi_edk2.h**lv_uefi_gnu_efi.h****lv_uefi_indev.h****Funzioni**

`lv_indev_t *lv_uefi_simple_pointer_indev_create(lv_point_t *display_res)`

Create an indev object.

Parametri

`display_res` -- The resolution of the display in pixels, needed to scale the input. If NULL the resolution of the current default display will be used.

Ritorna

The created LVGL indev object.

`bool lv_uefi_simple_pointer_indev_add_handle(lv_indev_t *indev, EFI_HANDLE handle)`

Add an EFI_SIMPLE_POINTER_PROTOCOL interface to the indev.

Parametri

- `indev` -- Indev that was created with `lv_uefi_simple_pointer_indev_create`.
- `handle` -- The handle on which an instance of the EFI_SIMPLE_POINTER_PROTOCOL protocol is installed.

Ritorna

True if the interface was added.

`void lv_uefi_simple_pointer_indev_add_all(lv_indev_t *indev)`

Add all available EFI_SIMPLE_POINTER_PROTOCOL interfaces to the indev.

Parametri

`indev` -- Indev that was created with `lv_uefi_simple_pointer_indev_create`.

`lv_indev_t *lv_uefi_absolute_pointer_indev_create(lv_point_t *display_res)`

Create a LVGL indev object.

Parametri

- display_res** -- The resolution of the display in pixels, needed to scale the input.

Ritorna

The created LVGL indev object.

```
bool lv_uefi_absolute_pointer_indev_add_handle(lv_indev_t *indev, EFI_HANDLE handle)
```

Add an EFI_ABSOLUTE_POINTER_PROTOCOL interface to the indev.

Parametri

- indev** -- Indev that was created with lv_uefi_absolute_pointer_indev_create.
- handle** -- The handle on which an instance of the EFI_ABSOLUTE_POINTER_PROTOCOL protocol is installed.

Ritorna

True if the interface was added.

```
void lv_uefi_absolute_pointer_indev_add_all(lv_indev_t *indev)
```

Add all available EFI_ABSOLUTE_POINTER_PROTOCOL interfaces to the indev.

Parametri

- indev** -- Indev that was created with lv_uefi_absolute_pointer_indev_create.

```
lv_indev_t *lv_uefi_simple_text_input_indev_create(void)
```

Create an indev object.

Ritorna

The created LVGL indev object.

```
bool lv_uefi_simple_text_input_indev_add_handle(lv_indev_t *indev, EFI_HANDLE handle)
```

Add an EFI_SIMPLE_TEXT_INPUT_EX_PROTOCOL interface to the indev.

Parametri

- indev** -- Indev that was created with lv_uefi_simple_text_input_indev_create.
- handle** -- The handle on which an instance of the EFI_SIMPLE_TEXT_INPUT_EX_PROTOCOL protocol is installed.

Ritorna

True if the interface was added.

```
void lv_uefi_simple_text_input_indev_add_all(lv_indev_t *indev)
```

Add all available EFI_SIMPLE_TEXT_INPUT_EX_PROTOCOL interfaces to the indev.

Parametri

- indev** -- Indev that was created with lv_uefi_simple_text_input_indev_create.

lv_uefi_private.h**Funzioni**

```
bool lv_uefi_protocol_test(EFI_HANDLE handle, EFI_GUID *protocol)
```

Test if a protocol is installed at a handle.

Parametri

- handle** -- The handle on which the protocol might be installed.
- protocol** -- The guid of the protocol.

Ritorna

TRUE if the protocol is installed, FALSE if not.

```
void *lv_uefi_protocol_open(EFI_HANDLE handle, EFI_GUID *protocol)
```

Open a protocol.

Parametri

- handle** -- The handle on which the protocol is installed.
- protocol** -- The guid of the protocol.

Ritorna

A pointer to the interface, NULL if the protocol couldn't be opened.

```
void lv_uefi_protocol_close(EFI_HANDLE handle, EFI_GUID *protocol)
```

Close a protocol.

Parametri

- **handle** -- The handle on which the protocol is installed.
- **protocol** -- The guid of the protocol.

`size_t lv_uefi_ucs2_to_ascii(const CHAR16 *ucs2, char *ascii, size_t ascii_len)`

Convert an UCS-2 string to an ASCII string. The string must contain only characters >= 0x20 and <= 0X7E.

Parametri

- **ucs2** -- The UCS-2 string.
- **ascii** -- The buffer to store the ASCII string.
- **ascii_len** -- The size of the buffer in ASCII characters.

Ritorna

The number of characters written to the buffer or 0 if there was an error.

`size_t lv_uefi_ascii_to_ucs2(const char *ascii, CHAR16 *ucs2, size_t ucs2_len)`

Convert an ASCII string to an UCS-2 string. The string must contain only characters >= 0x20 and <= 0X7E.

Parametri

- **ascii** -- The ASCII string.
- **ucs2** -- The buffer to store the UCS-2 string.
- **ucs2_len** -- The size of the buffer in UCS-2 characters.

Ritorna

The number of bytes written to the buffer or 0 if there was an error.

Variabili

EFI_HANDLE gLvEfiImageHandle

Internal cache for the image handle (source: application entry point)

EFI_SYSTEM_TABLE *gLvEfiST

Internal cache for the system table (source: application entry point)

EFI_BOOT_SERVICES *gLvEfiBS

Internal cache for the boot services table (source: gLvEfiST)

EFI_RUNTIME_SERVICES *gLvEfiRT

Internal cache for the boot runtime service table (source: gLvEfiST)

lv_uefi_std_wrapper.h

14.16.11 wayland

lv_wayland.h

Funzioni

`uint32_t lv_wayland_timer_handler(void)`

Wrapper around lv_timer_handler

 **Nota**

Must be called in the application run loop instead of the regular lv_timer_handler provided by LVGL

Ritorna

time till it needs to be run next (in ms)

`int lv_wayland_get_fd(void)`

Retrieves the file descriptor of the wayland socket

lv_wayland_private.h**Define****LV_WAYLAND_DEFAULT_CURSOR_NAME****LV_WAYLAND_MAX_OUTPUTS****LVGL_DRAW_BUFFER_DIV****DMG_CACHE_CAPACITY****TAG_LOCAL****TAG_BUFFER_DAMAGE****TITLE_BAR_HEIGHT****BORDER_SIZE****BUTTON_MARGIN****BUTTON_PADDING****BUTTON_SIZE****LV_WAYLAND_CYCLE_PERIOD****FIRST_DECORATION****LAST_DECORATION****NUM_DECORATIONS****Enum**enum **object_type***Values:*enumerator **OBJECT_TITLEBAR**enumerator **OBJECT_BUTTON_CLOSE**enumerator **OBJECT_BUTTON_MAXIMIZE**enumerator **OBJECT_BUTTON_MINIMIZE**

enumerator **OBJECT_BORDER_TOP**

enumerator **OBJECT_BORDER_BOTTOM**

enumerator **OBJECT_BORDER_LEFT**

enumerator **OBJECT_BORDER_RIGHT**

enumerator **OBJECT_WINDOW**

Funzioni

`void lv_wayland_init(void)`

`void lv_wayland_deinit(void)`

`void lv_wayland_wait_flush_cb(lv_display_t *disp)`

`void lv_wayland_event_cb(lv_event_t *e)`

`const struct wl_callback_listener *lv_wayland_window_get_wl_surface_frame_listener(void)`

`void lv_wayland_window_draw(struct window *window, uint32_t width, uint32_t height)`

`lv_result_t lv_wayland_window_resize(struct window *window, int width, int height)`

`void lv_wayland_window_destroy(struct window *window)`

`const struct xdg_surface_listener *lv_wayland_xdg_shell_get_surface_listener(void)`

`const struct xdg_toplevel_listener *lv_wayland_xdg_shell_get_toplevel_listener(void)`

`const struct xdg_wm_base_listener *lv_wayland_xdg_shell_get_wm_base_listener(void)`

`lv_result_t lv_wayland_xdg_shell_set_maximized(struct window *window, bool maximized)`

`lv_result_t lv_wayland_xdg_shell_set_minimized(struct window *window)`

`lv_result_t lv_wayland_xdg_shell_set_fullscreen(struct window *window, bool fullscreen, struct wl_output *output)`

`lv_result_t lv_wayland_xdg_shell_create_window(struct lv_wayland_context *app, struct window *window, const char *title)`

`void lv_wayland_xdg_shell_configure_surface(struct window *window)`

`lv_result_t lv_wayland_xdg_shell_destroy_window_toplevel(struct window *window)`

`lv_result_t lv_wayland_xdg_shell_destroy_window_surface(struct window *window)`

`void lv_wayland_xdg_shell_handle_pointer_event(struct lv_wayland_context *app, uint32_t serial, uint32_t button, uint32_t state)`

`const char *lv_wayland_xdg_shell_get_cursor_name(const struct lv_wayland_context *app)`

`void lv_wayland_xdg_shell_deinit(void)`

`void lv_wayland_shm_set_interface(shm_ctx_t *context, struct wl_registry *registry, uint32_t name, const char *interface, uint32_t version)`

```

struct graphic_object *lv_wayland_shm_on_graphical_object_creation(shm_ctx_t *context, struct graphic_object *obj)

void lv_wayland_shm_on_graphical_object_destruction(shm_ctx_t *context, struct graphic_object *obj)

lv_result_t lv_wayland_shm_set_draw_buffers(shm_ctx_t *context, lv_display_t *display, struct window *window)

lv_result_t lv_wayland_shm_create_draw_buffers(shm_ctx_t *context, struct window *window)

lv_result_t lv_wayland_shm_resize_window(shm_ctx_t *context, struct window *window, int32_t width, int32_t height)

lv_result_t lv_wayland_shm_is_ready(shm_ctx_t *context)

void lv_wayland_shm_delete_draw_buffers(shm_ctx_t *context, struct window *window)

void lv_wayland_shm_initialize_context(shm_ctx_t *context)

void lv_wayland_shm_deinit(shm_ctx_t *context)

void lv_wayland_shm_flush_partial_mode(lv_display_t *disp, const lv_area_t *area, unsigned char *color_p)

struct wl_cursor_theme *lv_wayland_shm_load_cursor_theme(shm_ctx_t *context)

void lv_wayland_dmabuf_set_interface(dmabuf_ctx_t *context, struct wl_registry *registry, uint32_t name, const char *interface, uint32_t version)

struct graphic_object *lv_wayland_dmabuf_on_graphical_object_creation(dmabuf_ctx_t *context, struct graphic_object *obj)

void lv_wayland_dmabuf_on_graphical_object_destruction(dmabuf_ctx_t *context, struct graphic_object *obj)

lv_result_t lv_wayland_dmabuf_set_draw_buffers(dmabuf_ctx_t *context, lv_display_t *display)

lv_result_t lv_wayland_dmabuf_create_draw_buffers(dmabuf_ctx_t *context, struct window *window)

lv_result_t lv_wayland_dmabuf_resize_window(dmabuf_ctx_t *context, struct window *window, int width, int height)

lv_result_t lv_wayland_dmabuf_is_ready(dmabuf_ctx_t *context)

void destroy_decorators_buf(struct window *window, struct graphic_object *decoration)

void lv_wayland_dmabuf_destroy_draw_buffers(dmabuf_ctx_t *context, struct window *window)

void lv_wayland_dmabuf_initialize_context(dmabuf_ctx_t *context)

void lv_wayland_dmabuf_deinit(dmabuf_ctx_t *context)

void lv_wayland_dmabuf_flush_full_mode(lv_display_t *disp, const lv_area_t *area, unsigned char *color_p)

const struct smm_events *lv_wayland_sme_get_events(void)

const struct wl_seat_listener *lv_wayland_seat_get_listener(void)

const struct wl_keyboard_listener *lv_wayland_keyboard_get_listener(void)

const struct wl_pointer_listener *lv_wayland_pointer_get_listener(void)

const struct wl_touch_listener *lv_wayland_touch_get_listener(void)

void lv_wayland_cache_add_area(struct window *window, smm_buffer_t *buf, const lv_area_t *area)

```

```
void lv_wayland_cache_clear(struct window *window)
void lv_wayland_cache_apply_areas(struct window *window, void *dest, void *src, smm_buffer_t *src_buf)
void lv_wayland_cache_purge(struct window *window, smm_buffer_t *buf)
```

Variabili

struct *lv_wayland_context* **lv_wl_ctx**

struct **input**

Membri Pubblici

uint32_t **x**

uint32_t **y**

lv_indev_state_t **left_button**

lv_indev_state_t **right_button**

lv_indev_state_t **wheel_button**

int16_t **wheel_diff**

struct *input* **pointer**

lv_key_t **key**

lv_indev_state_t **state**

struct *input* **keyboard**

lv_indev_touch_data_t **touches[10]**

uint8_t **touch_event_cnt**

uint8_t **primary_id**

struct **seat**

Membri Pubblici

struct *wl_touch* ***wl_touch**

struct *wl_pointer* ***wl_pointer**

```
struct wl_keyboard *wl_keyboard  
  
struct xkb_keymap *keymap  
  
struct xkb_state *state  
  
struct seat xkb  
  
struct graphic_object
```

Membri Pubblici

```
struct window *window  
  
struct wl_surface *surface  
  
struct wl_subsurface *subsurface  
  
smm_buffer_t *pending_buffer  
  
smm_group_t *buffer_group  
  
struct input input  
  
enum object_type type  
  
int width  
  
int height  
  
struct dmabuf_ctx_t
```

Membri Pubblici

```
struct buffer *buffers  
  
struct zwp_linux_dmabuf_v1 *handler  
  
uint32_t format  
  
uint8_t last_used  
  
struct shm_ctx_t
```

Membri Pubblici

```
struct wl_shm *handler  
  
uint32_t format  
  
struct output_info
```

Membri Pubblici

```
struct wl_output *wl_output
```

```
char name[64]
```

```
int width
```

```
int height
```

```
int refresh
```

```
int scale
```

```
int flags
```

```
struct lv_wayland_context
```

Membri Pubblici

```
struct wl_display *display
```

```
struct wl_registry *registry
```

```
struct wl_compositor *compositor
```

```
struct wl_subcompositor *subcompositor
```

```
struct wl_seat *wl_seat
```

```
struct wl_cursor_theme *cursor_theme
```

```
struct wl_surface *cursor_surface
```

```
shm_ctx_t shm_ctx
```

```
struct output_info outputs[8]
```

```
uint8_t wl_output_count

struct xdg_wm_base *xdg_wm

bool opt_disable_decorations

struct xkb_context *xkb_context

struct seat seat

struct graphic_object *touch_obj

struct graphic_object *pointer_obj

struct graphic_object *keyboard_obj

lv_ll_t window_ll

lv_timer_t *cycle_timer

bool cursor_flush_pending

struct pollfd wayland_pfd

struct window
```

Membri Pubblici

```
lv_display_t *lv_disp

lv_indev_t *lv_indev_pointer

lv_indev_t *lv_indev_pointeraxis

lv_indev_t *lv_indev_touch

lv_indev_t *lv_indev_keyboard

lv_draw_buf_t *lv_draw_buf

lv_wayland_display_close_f_t close_cb

struct lv_wayland_context *wl_ctx

struct wl_output *assigned_output
```

```
struct xdg_surface *xdg_surface

struct xdg_toplevel *xdg_toplevel

uint32_t wm_capabilities

struct graphic_object *body

lv_area_t cache[(32)]

unsigned char start

unsigned char end

unsigned size

struct window dmg_cache

int width

int height

bool resize_pending

int resize_width

int resize_height

bool flush_pending

bool shall_close

bool closed

bool maximized

bool fullscreen

uint32_t frame_counter

bool is_window_configured
```

lv_wayland_smm.h**Define****SMM_FD_NAME****SMM_POOL_TAGS****SMM_BUFFER_TAGS****SMM_GROUP_TAGS****SMM_POOL_PROPERTIES(p)****SMM_BUFFER_PROPERTIES(b)****SMM_GROUP_PROPERTIES(g)****SMM_TAG(o, n, v)****Typedef**typedef void **smm_pool_t**typedef void **smm_buffer_t**typedef void **smm_group_t****Funzioni**void **smm_init**(const struct *smm_events* *evs)void **smm_setctx**(void *ctx)void **smm_deinit**(void)*smm_group_t* ***smm_create**(void)void **smm_resize**(*smm_group_t* *grp, size_t sz)void **smm_destroy**(*smm_group_t* *grp)*smm_buffer_t* ***smm_acquire**(*smm_group_t* *grp)void ***smm_map**(*smm_buffer_t* *buf)void **smm_release**(*smm_buffer_t* *buf)*smm_buffer_t* ***smm_latest**(*smm_group_t* *grp)*smm_buffer_t* ***smm_next**(*smm_buffer_t* *buf)struct **smm_events**

Membri Pubblici

```
void *ctx  
  
bool (*new_pool)(void *ctx, smm_pool_t *pool)  
  
void (*expand_pool)(void *ctx, smm_pool_t *pool)  
  
void (*free_pool)(void *ctx, smm_pool_t *pool)  
  
bool (*new_buffer)(void *ctx, smm_buffer_t *buf)  
  
bool (*init_buffer)(void *ctx, smm_buffer_t *buf)  
  
void (*free_buffer)(void *ctx, smm_buffer_t *buf)
```

```
struct smm_pool_properties
```

Membri Pubblici

```
void *tag[(1)]  
  
size_t size  
  
int fd
```

```
struct smm_buffer_properties
```

Membri Pubblici

```
void *tag[(2)]  
  
smm_group_t *const group  
  
smm_pool_t *const pool  
  
size_t offset
```

```
struct smm_group_properties
```

Membri Pubblici

```
void *tag[(1)]
```

lv_wl_keyboard.h**Funzioni**

```
lv_indev_t *lv_wayland_keyboard_create(void)
lv_indev_t *lv_wayland_get_keyboard(lv_display_t *display)
```

Get keyboard input device for given LVGL display

Parametri

display -- LVGL display

Ritorna

input device connected to keyboard, or NULL on error

lv_wl_pointer.h**Funzioni**

```
lv_indev_t *lv_wayland_pointer_create(void)
lv_indev_t *lv_wayland_get_pointer(lv_display_t *disp)
```

Obtains the input device of the mouse pointer

Nota

It is used to create an input group on application start

Parametri

disp -- Reference to the LVGL display associated to the window

Ritorna

The input device

lv_wl_pointer_axis.h**Funzioni**

```
lv_indev_t *lv_wayland_pointer_axis_create(void)
lv_indev_t *lv_wayland_get_pointeraxis(lv_display_t *display)
```

Obtains the input device of the encoder

Nota

It is used to create an input group on application start

Parametri

display -- Reference to the LVGL display associated to the window

Ritorna

The input device

lv_wl_touch.h**Funzioni**

```
lv_indev_t *lv_wayland_touch_create(void)
lv_indev_t *lv_wayland_get_touchscreen(lv_display_t *display)
```

Get touchscreen input device for given LVGL display

Parametri

display -- LVGL display

Ritorna

input device connected to touchscreen, or NULL on error

lv_wl_window.h**Typedef**

```
typedef bool (*lv_wayland_display_close_f_t)(lv_display_t *disp)
```

Funzioni

```
lv_display_t *lv_wayland_window_create(uint32_t hor_res, uint32_t ver_res, char *title,  
                                         lv_wayland_display_close_f_t close_cb)
```

Creates a window

Parametri

- **hor_res** -- The width of the window in pixels
- **ver_res** -- The height of the window in pixels
- **title** -- The title of the window
- **close_cb** -- The callback that will be execute when the user closes the window

Ritorna

The LVGL display associated to the window

```
void lv_wayland_window_close(lv_display_t *disp)
```

Closes the window programmatically

Parametri

disp -- Reference to the LVGL display associated to the window

```
bool lv_wayland_window_is_open(lv_display_t *disp)
```

Check if the window is open

Parametri

disp -- Reference to the LVGL display associated to the window

Ritorna

true: The window is open

```
void lv_wayland_assign_physical_display(lv_display_t *disp, uint8_t display)
```

Assigns the window to a specific physical display

Parametri

- **disp** -- Reference to the LVGL display associated to the window
- **display** -- Physical display number

```
void lv_wayland_unassign_physical_display(lv_display_t *disp)
```

Unassigns the current physical display attached to the window

Parametri

disp -- Reference to the LVGL display associated to the window

```
void lv_wayland_window_setFullscreen(lv_display_t *disp, bool fullscreen)
```

Sets the fullscreen state of the window

Parametri

- **disp** -- Reference to the LVGL display associated to the window
- **fullscreen** -- If true the window enters fullscreen

```
void lv_wayland_window_setMaximized(lv_display_t *disp, bool maximize)
```

Sets the maximized state of the window

Parametri

- **disp** -- Reference to the LVGL display associated to the window
- **fullscreen** -- If true the window is maximized

14.16.12 windows

`lv_windows_context.h`

Define

`CREATE_WAITABLE_TIMER_MANUAL_RESET`

`CREATE_WAITABLE_TIMER_HIGH_RESOLUTION`

Typedef

`typedef struct _lv_windows_pointer_context_t lv_windows_pointer_context_t`

`typedef struct _lv_windows_keypad_queue_item_t lv_windows_keypad_queue_item_t`

`typedef struct _lv_windows_keypad_context_t lv_windows_keypad_context_t`

`typedef struct _lv_windows_encoder_context_t lv_windows_encoder_context_t`

`typedef struct _lv_windows_window_context_t lv_windows_window_context_t`

`typedef struct _lv_windows_create_display_data_t lv_windows_create_display_data_t`

Funzioni

`void lv_windows_platform_init(void)`

Initialize the LVGL Windows backend.

ⓘ Remark

This is a private API which is used for LVGL Windows backend implementation. LVGL users shouldn't use that because the LVGL has already used it in `lv_init`.

`lv_windows_window_context_t *lv_windows_get_window_context(HWND window_handle)`

Get the window context from specific LVGL display window.

ⓘ Remark

This is a private API which is used for LVGL Windows backend implementation. LVGL users shouldn't use that because the maintainer doesn't promise the application binary interface compatibility for this API.

Parametri

`window_handle` -- The window handle of specific LVGL display window.

Ritorna

The window context from specific LVGL display window.

`struct _lv_windows_pointer_context_t`

Membri Pubblici

lv_indev_state_t **state**

lv_point_t **point**

lv_indev_t ***indev**

struct **_lv_windows_keypad_queue_item_t**

Membri Pubblici

uint32_t **key**

lv_indev_state_t **state**

struct **_lv_windows_keypad_context_t**

Membri Pubblici

lv_ll_t **queue**

uint16_t **utf16_high_surrogate**

uint16_t **utf16_low_surrogate**

lv_indev_t ***indev**

struct **_lv_windows_encoder_context_t**

Membri Pubblici

lv_indev_state_t **state**

int16_t **enc_diff**

lv_indev_t ***indev**

struct **_lv_windows_window_context_t**

Membri Pubblici

lv_display_t ***display_device_object**

lv_timer_t ***display_timer_object**

int32_t **window_dpi**

```
int32_t zoom_level  
  
bool allow_dpi_override  
  
bool simulator_mode  
  
bool display_resolution_changed  
  
lv_point_t requested_display_resolution  
  
HDC display_framebuffer_context_handle  
  
uint32_t *display_framebuffer_base  
  
size_t display_framebuffer_size  
  
lv_windows_pointer_context_t pointer  
  
lv_windows_keypad_context_t keypad  
  
lv_windows_encoder_context_t encoder  
  
struct _lv_windows_create_display_data_t
```

Membri Pubblici

```
const wchar_t *title  
  
int32_t hor_res  
  
int32_t ver_res  
  
int32_t zoom_level  
  
bool allow_dpi_override  
  
bool simulator_mode  
  
HANDLE mutex  
  
lv_display_t *display
```

lv_windows_display.h**Define****LV_WINDOWS_ZOOM_BASE_LEVEL****USER_DEFAULT_SCREEN_DPI****Funzioni**

```
lv_display_t *lv_windows_create_display(const wchar_t *title, int32_t hor_res, int32_t ver_res, int32_t zoom_level, bool allow_dpi_override, bool simulator_mode)
```

Create a LVGL display object.

Parametri

- **title** -- The window title of LVGL display.
- **hor_res** -- The horizontal resolution value of LVGL display.
- **ver_res** -- The vertical resolution value of LVGL display.
- **zoom_level** -- The zoom level value. Base value is 100 a.k.a 100%.
- **allow_dpi_override** -- Allow DPI override if true, or follow the Windows DPI scaling setting dynamically.
- **simulator_mode** -- Create simulator mode display if true (not resizable), or create application mode display (resizable).

Ritorna

The created LVGL display object.

```
HWND lv_windows_get_display_window_handle(lv_display_t *display)
```

Get the window handle from specific LVGL display object.

Parametri

- **display** -- The specific LVGL display object.

Ritorna

The window handle from specific LVGL display object.

```
int32_t lv_windows_zoom_to_logical(int32_t physical, int32_t zoom_level)
```

Get logical pixel value from physical pixel value taken account with zoom level.

Remark

It uses the same calculation style as Windows OS implementation. It will be useful for integrate LVGL Windows backend to other Windows applications.

Parametri

- **physical** -- The physical pixel value taken account with zoom level.
- **zoom_level** -- The zoom level value. Base value is 100 a.k.a 100%.

Ritorna

The logical pixel value.

```
int32_t lv_windows_zoom_to_physical(int32_t logical, int32_t zoom_level)
```

Get physical pixel value taken account with zoom level from logical pixel value.

Remark

It uses the same calculation style as Windows OS implementation. It will be useful for integrate LVGL Windows backend to other Windows applications.

Parametri

- **logical** -- The logical pixel value.
- **zoom_level** -- The zoom level value. Base value is 100 a.k.a 100%.

Ritorna

The physical pixel value taken account with zoom level.

`int32_t lv_windows_dpi_to_logical(int32_t physical, int32_t dpi)`

Get logical pixel value from physical pixel value taken account with DPI scaling.

 **Remark**

It uses the same calculation style as Windows OS implementation. It will be useful for integrate LVGL Windows backend to other Windows applications.

Parametri

- **physical** -- The physical pixel value taken account with DPI scaling.
- **dpi** -- The DPI scaling value. Base value is USER_DEFAULT_SCREEN_DPI.

Ritorna

The logical pixel value.

`int32_t lv_windows_dpi_to_physical(int32_t logical, int32_t dpi)`

Get physical pixel value taken account with DPI scaling from logical pixel value.

 **Remark**

It uses the same calculation style as Windows OS implementation. It will be useful for integrate LVGL Windows backend to other Windows applications.

Parametri

- **logical** -- The logical pixel value.
- **dpi** -- The DPI scaling value. Base value is USER_DEFAULT_SCREEN_DPI.

Ritorna

The physical pixel value taken account with DPI scaling.

lv_windows_input.h**Funzioni**

`HWND lv_windows_get_indev_window_handle(lv_indev_t *indev)`

Get the window handle from specific LVGL input device object.

Parametri

- **indev** -- The specific LVGL input device object.

Ritorna

The window handle from specific LVGL input device object.

`lv_indev_t *lv_windows_acquire_pointer_indev(lv_display_t *display)`

Open a LVGL pointer input device object for the specific LVGL display object, or create it if the LVGL pointer input device object is not created or removed before.

Parametri

- **display** -- The specific LVGL display object.

Ritorna

The LVGL pointer input device object for the specific LVGL display object.

`lv_indev_t *lv_windows_acquire_keypad_indev(lv_display_t *display)`

Open a LVGL keypad input device object for the specific LVGL display object, or create it if the LVGL keypad input device object is not created or removed before.

Parametri

- **display** -- The specific LVGL display object.

Ritorna

The LVGL keypad input device object for the specific LVGL display object.

`lv_indev_t *lv_windows_acquire_encoder_indev(lv_display_t *display)`

Open a LVGL encoder input device object for the specific LVGL display object, or create it if the LVGL encoder input device object is not created or removed before.

Parametri

`display` -- The specific LVGL display object.

Ritorna

The LVGL encoder input device object for the specific LVGL display object.

lv_windows_input_private.h

14.16.13 x11

lv_x11.h

Typedef

`typedef void (*lv_x11_close_cb)(void *user_data)`

optional window close callback function type

Vedi anche

[lv_x11_window_set_close_cb](#)

Funzioni

`void lv_x11_inputs_create(lv_display_t *disp, lv_image_dsc_t const *mouse_img)`

create and add keyboard, mouse and scrollwheel objects and connect them to x11 display.

This is a convenience method handling the typical input initialisation of an X11 window:

- create keyboard (`lv_x11_keyboard_create`)
- create mouse (with scrollwheel, `lv_x11_mouse_create` `lv_x11_mousewheel_create`)

Parametri

- `disp` -- [in] the created X11 display object from `lv_x11_window_create`
- `mouse_img` -- [in] optional image description for the mouse cursor (NULL for no/invisible mouse cursor)

`lv_display_t *lv_x11_window_create(char const *title, int32_t hor_res, int32_t ver_res)`

create the X11 display

The minimal initialisation for initializing the X11 display driver with keyboard/mouse support:

```
lv_display_t* disp = lv_x11_window_create("My Window Title", window_width, window_width);
lv_x11_inputs_create(disp, NULL);
```

or with mouse cursor icon:

```
lv_image_dsc_t mouse_symbol = {....};
lv_display_t* disp = lv_x11_window_create("My Window Title", window_width, window_width);
lv_x11_inputs_create(disp, &mouse_symbol);
```

Parametri

- `title` -- [in] title of the created X11 window
- `hor_res` -- [in] horizontal resolution (=width) of the X11 window
- `ver_res` -- [in] vertical resolution (=height) of the X11 window

Ritorna

pointer to the display object

`struct _x11_user_hdr_t`

`#include <lv_x11.h>` Header of private display driver user data - for internal use only

Membri Pubblici

```
struct _XDisplay *display  
    X11 display object  
  
struct _x11_inp_data *inp_data  
    input user data object
```

14.17 font

14.17.1 lv_font.h

Define

```
LV_FONT_DECLARE(font_name)
```

Enum

```
enum lv_font_glyph_format_t
```

The font format.

Values:

```
enumerator LV_FONT_GLYPH_FORMAT_NONE
```

Maybe not visible Legacy simple formats

```
enumerator LV_FONT_GLYPH_FORMAT_A1
```

1 bit per pixel

```
enumerator LV_FONT_GLYPH_FORMAT_A2
```

2 bit per pixel

```
enumerator LV_FONT_GLYPH_FORMAT_A3
```

3 bit per pixel

```
enumerator LV_FONT_GLYPH_FORMAT_A4
```

4 bit per pixel

```
enumerator LV_FONT_GLYPH_FORMAT_A8
```

8 bit per pixel

```
enumerator LV_FONT_GLYPH_FORMAT_IMAGE
```

Image format Advanced formats

```
enumerator LV_FONT_GLYPH_FORMAT_VECTOR
```

Vectorial format

```
enumerator LV_FONT_GLYPH_FORMAT_SVG
```

SVG format

enumerator **LV_FONT_GLYPH_FORMAT_CUSTOM**

Custom format

enum **lv_font_subpx_t**

The bitmaps might be upscaled by 3 to achieve subpixel rendering.

Values:

enumerator **LV_FONT_SUBPX_NONE**

enumerator **LV_FONT_SUBPX_HOR**

enumerator **LV_FONT_SUBPX_VER**

enumerator **LV_FONT_SUBPX_BOTH**

enum **lv_font_kerning_t**

Adjust letter spacing for specific character pairs.

Values:

enumerator **LV_FONT_KERNING_NORMAL**

enumerator **LV_FONT_KERNING_NONE**

Funzioni

const void ***lv_font_get_glyph_bitmap**(*lv_font_glyph_dsc_t* *g_dsc, lv_draw_buf_t *draw_buf)

Return with the bitmap of a font. It always converts the normal fonts to A8 format in a draw_buf with LV_DRAW_BUF_ALIGN and LV_DRAW_BUF_STRIDE_ALIGN

Nota

You must call *lv_font_get_glyph_dsc()* to get g_dsc (*lv_font_glyph_dsc_t*) before you can call this function.

Parametri

- **g_dsc** -- the glyph descriptor including which font to use, which supply the glyph_index and the format.
- **draw_buf** -- a draw buffer that can be used to store the bitmap of the glyph.

Ritorna

pointer to the glyph's data. It can be a draw buffer for bitmap fonts or an image source for imgfonts.

const void ***lv_font_get_glyph_static_bitmap**(*lv_font_glyph_dsc_t* *g_dsc)

Return the bitmap as it is. It works only if the font stores the bitmap in a non-volatile memory.

Parametri

g_dsc -- the glyph descriptor including which font to use, which supply the glyph_index and the format.

Ritorna

the bitmap as it is

bool **lv_font_get_glyph_dsc**(const lv_font_t *font, *lv_font_glyph_dsc_t* *dsc_out, uint32_t letter, uint32_t letter_next)

Get the descriptor of a glyph

Parametri

- **font** -- pointer to font
- **dsc_out** -- store the result descriptor here
- **letter** -- a UNICODE letter code
- **letter_next** -- the next letter after **letter**. Used for kerning

Ritorna

true: descriptor is successfully loaded into **dsc_out**. false: the letter was not found, no data is loaded to **dsc_out**

```
void lv_font_glyph_release_draw_data(lv_font_glyph_dsc_t *g_dsc)
```

Release the bitmap of a font.

 **Nota**

You must call *lv_font_get_glyph_dsc()* to get **g_dsc** (*lv_font_glyph_dsc_t*) before you can call this function.

Parametri

- **g_dsc** -- the glyph descriptor including which font to use, which supply the **glyph_index** and the format.

```
uint16_t lv_font_get_glyph_width(const lv_font_t *font, uint32_t letter, uint32_t letter_next)
```

Get the width of a glyph with kerning

Parametri

- **font** -- pointer to a font
- **letter** -- a UNICODE letter
- **letter_next** -- the next letter after **letter**. Used for kerning

Ritorna

the width of the glyph

```
int32_t lv_font_get_line_height(const lv_font_t *font)
```

Get the line height of a font. All characters fit into this height

Parametri

- **font** -- pointer to a font

Ritorna

the height of a font

```
void lv_font_set_kerning(lv_font_t *font, lv_font_kerning_t kerning)
```

Configure the use of kerning information stored in a font

Parametri

- **font** -- pointer to a font
- **kerning** -- LV_FONT_KERNING_NORMAL (default) or LV_FONT_KERNING_NONE

```
const lv_font_t *lv_font_get_default(void)
```

Get the default font, defined by LV_FONT_DEFAULT

Ritorna

return pointer to the default font

```
bool lv_font_info_is_equal(const lv_font_info_t *ft_info_1, const lv_font_info_t *ft_info_2)
```

Compare font information.

Parametri

- **ft_info_1** -- font information 1.
- **ft_info_2** -- font information 2.

Ritorna

return true if the fonts are equal.

```
bool lv_font_has_static_bitmap(const lv_font_t *font)
```

Checks if a font has a static rendering bitmap.

Parametri

- **font** -- pointer to a font

Ritorna

return true if the font has a bitmap generated for static rendering.

Variabili

```
const lv_font_t lv_font_montserrat_8
```

```
const lv_font_t lv_font_montserrat_10
```

```
const lv_font_t lv_font_montserrat_12
```

```
const lv_font_t lv_font_montserrat_14
```

```
const lv_font_t lv_font_montserrat_16
```

```
const lv_font_t lv_font_montserrat_18
```

```
const lv_font_t lv_font_montserrat_20
```

```
const lv_font_t lv_font_montserrat_22
```

```
const lv_font_t lv_font_montserrat_24
```

```
const lv_font_t lv_font_montserrat_26
```

```
const lv_font_t lv_font_montserrat_28
```

```
const lv_font_t lv_font_montserrat_30
```

```
const lv_font_t lv_font_montserrat_32
```

```
const lv_font_t lv_font_montserrat_34
```

```
const lv_font_t lv_font_montserrat_36
```

```
const lv_font_t lv_font_montserrat_38
```

```
const lv_font_t lv_font_montserrat_40
```

```
const lv_font_t lv_font_montserrat_42
```

```
const lv_font_t lv_font_montserrat_44
```

```
const lv_font_t lv_font_montserrat_46
```

```
const lv_font_t lv_font_montserrat_48
```

```
const lv_font_t lv_font_montserrat_28_compressed
```

```
const lv_font_t lv_font_dejavu_16_persian_hebrew
```

```
const lv_font_t lv_font_source_han_sans_sc_14_cjk
```

```
const lv_font_t lv_font_source_han_sans_sc_16_cjk
```

```
const lv_font_t lv_font_unscii_8
```

```
const lv_font_t lv_font_unscii_16
```

```
struct lv_font_glyph_dsc_t
```

#include <lv_font.h> Describes the properties of a glyph.

Membri Pubblici

```
const lv_font_t *resolved_font
```

Pointer to a font where the glyph was actually found after handling fallbacks

```
uint16_t adv_w
```

The glyph needs this space. Draw the next glyph after this width.

```
uint16_t box_w
```

Width of the glyph's bounding box

```
uint16_t box_h
```

Height of the glyph's bounding box

```
int16_t ofs_x
```

x offset of the bounding box

```
int16_t ofs_y
```

y offset of the bounding box

```
uint16_t stride
```

Bytes in each line. If 0 than there is no padding at the end of the line.

lv_font_glyph_format_t format

Font format of the glyph see *lv_font_glyph_format_t*

```
uint8_t is_placeholder
```

Glyph is missing. But placeholder will still be displayed

```
uint8_t req_raw_bitmap
```

0: Get bitmap should return an A8 or ARGB8888 image. 1: return the bitmap as it is (Maybe A1/2/4 or any proprietary formats).

int32_t outline_stroke_width

used with freetype vector fonts - width of the letter border

uint32_t index

Glyph descriptor index

const void *src

Pointer to the source data used by image fonts

union *lv_font_glyph_dsc_t* gid

The index of the glyph in the font file. Used by the font cache

lv_cache_entry_t *entry

The cache entry of the glyph draw data. Used by the font cache

struct _lv_font_t

#include <lv_font.h> Describe the properties of a font

Membri Pubblici**bool (*get_glyph_dsc)(const lv_font_t*, *lv_font_glyph_dsc_t**, uint32_t letter, uint32_t letter_next)**

Get a glyph's descriptor from a font

const void *(*get_glyph_bitmap)(*lv_font_glyph_dsc_t, lv_draw_buf_t*)**

Get a glyph's bitmap from a font

void (*release_glyph)(const lv_font_t*, *lv_font_glyph_dsc_t)**

Release a glyph

int32_t line_height

The real line height where any text fits

int32_t base_line

Base line measured from the bottom of the line_height

uint8_t subpx

An element of *lv_font_subpx_t*

uint8_t kerning

An element of *lv_font_kerning_t*

uint8_t static_bitmap

The font will be used as static bitmap

int8_t underline_position

Distance between the top of the underline and base line (< 0 means below the base line)

int8_t underline_thickness

Thickness of the underline

```
const void *dsc
    Store implementation specific or run_time data or caching here

const lv_font_t *fallback
    Fallback font for missing glyph. Resolved recursively

void *user_data
    Custom user data for font.

struct _lv_font_class_t
```

Membri Pubblici

```
lv_font_t *(*create_cb)(const lv_font_info_t *info, const void *src)
    Font creation callback function
```

```
void (*delete_cb)(lv_font_t *font)
    Font deletion callback function
```

```
void *(*dup_src_cb)(const void *src)
    Font source duplication callback function
```

```
void (*free_src_cb)(void *src)
    Font source free callback function
```

```
struct _lv_font_info_t
```

Membri Pubblici

```
const char *name
    Font name, used to distinguish different font resources
```

```
const lv_font_class_t *class_p
    Font backend implementation
```

```
uint32_t size
    Font size in pixel
```

```
uint32_t render_mode
    Font rendering mode, see lv\_freetype\_font\_render\_mode\_t
```

```
uint32_t style
    Font style, see lv\_freetype\_font\_style\_t
```

```
lv_font_kerning_t kerning
    Font kerning, see lv\_font\_kerning\_t
```

14.17.2 lv_symbol_def.h

Define

`LV_SYMBOL_BULLET`

`LV_SYMBOL_AUDIO`

`LV_SYMBOL_VIDEO`

`LV_SYMBOL_LIST`

`LV_SYMBOL_OK`

`LV_SYMBOL_CLOSE`

`LV_SYMBOL_POWER`

`LV_SYMBOL_SETTINGS`

`LV_SYMBOL_HOME`

`LV_SYMBOL_DOWNLOAD`

`LV_SYMBOL_DRIVE`

`LV_SYMBOL_REFRESH`

`LV_SYMBOL_MUTE`

`LV_SYMBOL_VOLUME_MID`

`LV_SYMBOL_VOLUME_MAX`

`LV_SYMBOL_IMAGE`

`LV_SYMBOL_TINT`

`LV_SYMBOL_PREV`

`LV_SYMBOL_PLAY`

`LV_SYMBOL_PAUSE`

`LV_SYMBOL_STOP`

LV_SYMBOL_NEXT

LV_SYMBOL_EJECT

LV_SYMBOL_LEFT

LV_SYMBOL_RIGHT

LV_SYMBOL_PLUS

LV_SYMBOL_MINUS

LV_SYMBOL_EYE_OPEN

LV_SYMBOL_EYE_CLOSE

LV_SYMBOL_WARNING

LV_SYMBOL_SHUFFLE

LV_SYMBOL_UP

LV_SYMBOL_DOWN

LV_SYMBOL_LOOP

LV_SYMBOL_DIRECTORY

LV_SYMBOL_UPLOAD

LV_SYMBOL_CALL

LV_SYMBOL_CUT

LV_SYMBOL_COPY

LV_SYMBOL_SAVE

LV_SYMBOL_BARS

LV_SYMBOL_ENVELOPE

LV_SYMBOL_CHARGE

LV_SYMBOL_PASTE

LV_SYMBOL_BELL

LV_SYMBOL_KEYBOARD

LV_SYMBOL_GPS

LV_SYMBOL_FILE

LV_SYMBOL_WIFI

LV_SYMBOL_BATTERY_FULL

LV_SYMBOL_BATTERY_3

LV_SYMBOL_BATTERY_2

LV_SYMBOL_BATTERY_1

LV_SYMBOL_BATTERY_EMPTY

LV_SYMBOL_USB

LV_SYMBOL_BLUETOOTH

LV_SYMBOL_TRASH

LV_SYMBOL_EDIT

LV_SYMBOL_BACKSPACE

LV_SYMBOL_SD_CARD

LV_SYMBOL_NEW_LINE

LV_SYMBOL_DUMMY

Invalid symbol at (U+F8FF). If written before a string then `lv_img` will show it as a label

Enum

enum `_lv_str_symbol_id_t`

Values:

enumerator **LV_STR_SYMBOL_BULLET**

enumerator **LV_STR_SYMBOL_AUDIO**

enumerator **LV_STR_SYMBOL_VIDEO**

enumerator **LV_STR_SYMBOL_LIST**

enumerator **LV_STR_SYMBOL_OK**

enumerator **LV_STR_SYMBOL_CLOSE**

enumerator **LV_STR_SYMBOL_POWER**

enumerator **LV_STR_SYMBOL_SETTINGS**

enumerator **LV_STR_SYMBOL_HOME**

enumerator **LV_STR_SYMBOL_DOWNLOAD**

enumerator **LV_STR_SYMBOL_DRIVE**

enumerator **LV_STR_SYMBOL_REFRESH**

enumerator **LV_STR_SYMBOL_MUTE**

enumerator **LV_STR_SYMBOL_VOLUME_MID**

enumerator **LV_STR_SYMBOL_VOLUME_MAX**

enumerator **LV_STR_SYMBOL_IMAGE**

enumerator **LV_STR_SYMBOL_TINT**

enumerator **LV_STR_SYMBOL_PREV**

enumerator **LV_STR_SYMBOL_PLAY**

enumerator **LV_STR_SYMBOL_PAUSE**

enumerator **LV_STR_SYMBOL_STOP**

enumerator **LV_STR_SYMBOL_NEXT**

enumerator **LV_STR_SYMBOL_EJECT**

enumerator **LV_STR_SYMBOL_LEFT**

enumerator **LV_STR_SYMBOL_RIGHT**

enumerator **LV_STR_SYMBOL_PLUS**

enumerator **LV_STR_SYMBOL_MINUS**

enumerator **LV_STR_SYMBOL_EYE_OPEN**

enumerator **LV_STR_SYMBOL_EYE_CLOSE**

enumerator **LV_STR_SYMBOL_WARNING**

enumerator **LV_STR_SYMBOL_SHUFFLE**

enumerator **LV_STR_SYMBOL_UP**

enumerator **LV_STR_SYMBOL_DOWN**

enumerator **LV_STR_SYMBOL_LOOP**

enumerator **LV_STR_SYMBOL_DIRECTORY**

enumerator **LV_STR_SYMBOL_UPLOAD**

enumerator **LV_STR_SYMBOL_CALL**

enumerator **LV_STR_SYMBOL_CUT**

enumerator **LV_STR_SYMBOL_COPY**

enumerator **LV_STR_SYMBOL_SAVE**

enumerator **LV_STR_SYMBOL_BARS**

enumerator **LV_STR_SYMBOL_ENVELOPE**

enumerator **LV_STR_SYMBOL_CHARGE**

enumerator **LV_STR_SYMBOL_PASTE**

enumerator **LV_STR_SYMBOL_BELL**

enumerator **LV_STR_SYMBOL_KEYBOARD**

enumerator **LV_STR_SYMBOL_GPS**

enumerator **LV_STR_SYMBOL_FILE**

enumerator **LV_STR_SYMBOL_WIFI**

enumerator **LV_STR_SYMBOL_BATTERY_FULL**

enumerator **LV_STR_SYMBOL_BATTERY_3**

enumerator **LV_STR_SYMBOL_BATTERY_2**

enumerator **LV_STR_SYMBOL_BATTERY_1**

enumerator **LV_STR_SYMBOL_BATTERY_EMPTY**

enumerator **LV_STR_SYMBOL_USB**

enumerator **LV_STR_SYMBOL_BLUETOOTH**

enumerator **LV_STR_SYMBOL_TRASH**

enumerator **LV_STR_SYMBOL_EDIT**

enumerator **LV_STR_SYMBOL_BACKSPACE**

enumerator **LV_STR_SYMBOL_SD_CARD**

enumerator **LV_STR_SYMBOL_NEW_LINE**

enumerator **LV_STR_SYMBOL_DUMMY**

14.17.3 binfont_loader

lv_binfont_loader.h

Funzioni

`lv_font_t *lv_binfont_create(const char *path)`

Loads a `lv_font_t` object from a binary font file

Parametri

`path` -- path to font file

Ritorna

pointer to font where to load

`void lv_binfont_destroy(lv_font_t *font)`

Frees the memory allocated by the `lv_binfont_create()` function

Parametri

`font` -- `lv_font_t` object created by the `lv_binfont_create` function

Variabili

const `lv_font_class_t lv_binfont_font_class`

struct `lv_binfont_font_src_t`

Membri Pubblici

`uint32_t font_size`

Size of the font in pixels

`const char *path`

Path to font file

`const void *buffer`

Address of the font file in the memory

`uint32_t buffer_size`

Size of the font file buffer

14.17.4 fmt_txt

lv_font_fmt_txt.h

Enum

enum `lv_font_fmt_txt_cmap_type_t`

Format of font character map.

Values:

enumerator `LV_FONT_FMT_TXT_CMAP_FORMAT0_FULL`

enumerator `LV_FONT_FMT_TXT_CMAP_SPARSE_FULL`

enumerator `LV_FONT_FMT_TXT_CMAP_FORMAT0_TINY`

enumerator `LV_FONT_FMT_TXT_CMAP_SPARSE_TINY`

enum `lv_font_fmt_txt_bitmap_format_t`

Bitmap formats

Values:

enumerator `LV_FONT_FMT_TXT_PLAIN`

enumerator `LV_FONT_FMT_TXT_COMPRESSED`

enumerator `LV_FONT_FMT_TXT_COMPRESSED_NO_PREFILTER`

Funzioni

`const void *lv_font_get_bitmap_fmt_txt(lv_font_glyph_dsc_t *g_dsc, lv_draw_buf_t *draw_buf)`

Used as `get_glyph_bitmap` callback in lvgl's native font format if the font is uncompressed.

Parametri

- `g_dsc` -- the glyph descriptor including which font to use, which supply the `glyph_index` and format.
- `draw_buf` -- a draw buffer that can be used to store the bitmap of the glyph, it's OK not to use it.

Ritorna

pointer to an A8 bitmap (not necessarily bitmap_out) or NULL if unicode_letter not found

```
bool lv_font_get_glyph_dsc_fmt_txt(const lv_font_t *font, lv_font_glyph_dsc_t *dsc_out, uint32_t
                                    unicode_letter, uint32_t unicode_letter_next)
```

Used as get_glyph_dsc callback in lvgl's native font format if the font is uncompressed.

Parametri

- **font** -- pointer to font
- **dsc_out** -- store the result descriptor here
- **unicode_letter** -- a UNICODE letter code
- **unicode_letter_next** -- the unicode letter succeeding the letter under test

Ritorna

true: descriptor is successfully loaded into dsc_out. false: the letter was not found, no data is loaded to dsc_out

Variabili

```
const lv_font_class_t lv_builtin_font_class
```

```
struct lv_font_fmt_txt_glyph_dsc_t
```

#include <lv_font_fmt_txt.h> This describes a glyph.

Membri Pubblici

```
uint32_t bitmap_index
```

Start index of the bitmap. A font can be max 4 GB.

```
uint32_t adv_w
```

Draw the next glyph after this width. 28.4 format (real_value * 16 is stored).

```
uint16_t box_w
```

Width of the glyph's bounding box

```
uint16_t box_h
```

Height of the glyph's bounding box

```
int16_t ofs_x
```

x offset of the bounding box

```
int16_t ofs_y
```

y offset of the bounding box. Measured from the top of the line

```
struct lv_font_fmt_txt_cmap_t
```

#include <lv_font_fmt_txt.h> Map codepoints to a glyph_dscs Several formats are supported to optimize memory usage See https://github.com/lvgl/lv_font_conv/blob/master/doc/font_spec.md

Membri Pubblici

```
uint32_t range_start
```

First Unicode character for this range

uint16_t range_length

Number of Unicode characters related to this range. Last Unicode character = range_start + range_length - 1

uint16_t glyph_id_start

First glyph ID (array index of `glyph_dsc`) for this range

const uint16_t *unicode_list

const void *glyph_id_ofs_list

if(type == LV_FONT_FMT_TXT_CMAP_FORMAT0...) it's `uint8_t *` if(type == LV_FONT_FMT_TXT_CMAP_SPARSE...) it's `uint16_t *`

uint16_t list_length

Length of `unicode_list` and/or `glyph_id_ofs_list`

lv_font_fmt_txt_cmap_type_t type

Type of this character map

struct lv_font_fmt_txt_kern_pair_t

#include <lv_font_fmt_txt.h> A simple mapping of kern values from pairs

Membri Pubblici

const void *glyph_ids

const int8_t *values

uint32_t pair_cnt

uint32_t glyph_ids_size

0: `glyph_ids` is stored as `uint8_t`; 1: as `uint16_t`

struct lv_font_fmt_txt_kern_classes_t

#include <lv_font_fmt_txt.h> More complex but more optimal class based kern value storage

Membri Pubblici

const int8_t *class_pair_values

`left_class_cnt` * `right_class_cnt` value

const uint8_t *left_class_mapping

Map the `glyph_ids` to classes: index -> `glyph_id` -> `class_id`

const uint8_t *right_class_mapping

Map the `glyph_ids` to classes: index -> `glyph_id` -> `class_id`

uint8_t left_class_cnt

```

uint8_t right_class_cnt

struct lv_font_fmt_txt_dsc_t
#include <lv_font_fmt_txt.h> Describe store for additional data for fonts

Membri Pubblici

const uint8_t *glyph_bitmap
The bitmaps of all glyphs

const lv_font_fmt_txt_glyph_dsc_t *glyph_dsc
Describe the glyphs

const lv_font_fmt_txt_cmap_t *cmaps
Map the glyphs to Unicode characters. Array of lv_font_cmap_fmt_txt_t variables

const void *kern_dsc
Store kerning values. Can be lv_font_fmt_txt_kern_pair_t * or lv_font_kern_classes_fmt_txt_t *
depending on kern_classes

uint16_t kern_scale
Scale kern values in 12.4 format

uint16_t cmap_num
Number of cmap tables

uint16_t bpp
Bit per pixel: 1, 2, 3, 4, 8

uint16_t kern_classes
Type of kern_dsc

uint16_t bitmap_format
storage format of the bitmap from lv_font_fmt_txt_bitmap_format_t

uint8_t stride
Bytes to which each line is padded. 0: means no align and padding 1: e.g. with bpp=4 lines are aligned
to 1 byte, so there can be a 4 bits of padding 4, 8, 16, 32, 64: each line is padded to the given byte
boundaries

struct lv_builtin_font_src_t

Membri Pubblici

const lv_font_t *font_p
Pointer to built-in font

uint32_t size

```

lv_font_fmt_txt_private.h**Enum**enum **lv_font_fmt_rle_state_t***Values:*enumerator **RLE_STATE_SINGLE**enumerator **RLE_STATE_REPEATED**enumerator **RLE_STATE_COUNTER**struct **lv_font_fmt_rle_t****Membri Pubblici**uint32_t **rdp**const uint8_t ***in**uint8_t **bpp**uint8_t **prev_v**uint8_t **count***lv_font_fmt_rle_state_t* **state****14.17.5 font_manager****lv_font_manager.h****Funzioni**lv_font_manager_t ***lv_font_manager_create**(uint32_t recycle_cache_size)

Create main font manager.

Parametri

recycle_cache_size -- number of fonts that were recently deleted from the cache.

Ritorna

pointer to main font manager.

bool **lv_font_manager_delete**(lv_font_manager_t *manager)

Delete main font manager.

Parametri

manager -- pointer to main font manager.

Ritorna

return true if the deletion was successful.

bool **lv_font_manager_add_src**(lv_font_manager_t *manager, const char *name, const void *src, const lv_font_class_t *class_p)

Add font resource.

Parametri

- manager -- pointer to main font manager.

- **name** -- font name.
- **src** -- font source. Need to strictly correspond to the font class.
- **class_p** -- font class. eg. lv_freetype_font_class, lv_builtin_font_class.

Ritorna

return true if the add was successful.

```
bool lv_font_manager_add_src_static(lv_font_manager_t *manager, const char *name, const void *src, const
                                    lv_font_class_t *class_p)
```

Add font resource with static memory.

Parametri

- **manager** -- pointer to main font manager.
- **name** -- font name. It cannot be a local variable.
- **src** -- font source. Need to strictly correspond to the font class. And it cannot be a local variable.
- **class_p** -- font class. E.g. lv_freetype_font_class, lv_builtin_font_class.

Ritorna

return true if the add was successful.

```
bool lv_font_manager_remove_src(lv_font_manager_t *manager, const char *name)
```

Remove font resource.

Parametri

- **manager** -- pointer to main font manager.
- **name** -- font name.

Ritorna

return true if the remove was successful.

```
lv_font_t *lv_font_manager_create_font(lv_font_manager_t *manager, const char *font_family, uint32_t
                                       render_mode, uint32_t size, uint32_t style, lv_font_kerning_t
                                       kerning)
```

Create font.

Parametri

- **manager** -- pointer to main font manager.
- **font_family** -- font family name. Matches the font resource name, using commas to separate different names. E.g. "my_font_1,my_font_2".
- **render_mode** -- font render mode. see lv_freetype_font_render_mode_t.
- **size** -- font size in pixel.
- **style** -- font style. see lv_freetype_font_style_t.
- **kerning** -- kerning mode. see lv_font_kerning_t.

Ritorna

point to the created font.

```
void lv_font_manager_delete_font(lv_font_manager_t *manager, lv_font_t *font)
```

Delete font.

Parametri

- **manager** -- pointer to main font manager.
- **font** -- point to the font.

Ritorna

return true if the deletion was successful.

lv_font_manager_recycle.h

Typedef

```
typedef struct _lv_font_manager_recycle_t lv_font_manager_recycle_t
```

Funzioni

`lv_font_manager_recycle_t *lv_font_manager_recycle_create(uint32_t max_size)`

Create font recycle manager.

Parametri

- max_size** -- recycle size.

Ritorna

pointer to font recycle manager.

`void lv_font_manager_recycle_delete(lv_font_manager_recycle_t *manager)`

Delete font recycle manager.

Parametri

- manager** -- pointer to font recycle manager.

`lv_font_t *lv_font_manager_recycle_get_reuse(lv_font_manager_recycle_t *manager, const lv_font_info_t *ft_info)`

Get a reusable font.

Parametri

- **manager** -- pointer to font recycle manager.
- **ft_info** -- font info.

Ritorna

returns true on success.

`void lv_font_manager_recycle_set_reuse(lv_font_manager_recycle_t *manager, lv_font_t *font, const lv_font_info_t *ft_info)`

Set fonts to be reused.

Parametri

- **manager** -- pointer to font recycle manager.
- **ft_info** -- font info.

`void lv_font_recycle_remove_fonts(lv_font_manager_recycle_t *manager, const char *name)`

Remove fonts with name from recycle manager.

Parametri

- **manager** -- pointer to font recycle manager.
- **name** -- font name.

14.17.6 imgfont

`lv_imgfont.h`

Typedef

`typedef const void *(*lv_imgfont_get_path_cb_t)(const lv_font_t *font, uint32_t unicode, uint32_t unicode_next, int32_t *offset_y, void *user_data)`

Funzioni

`lv_font_t *lv_imgfont_create(uint16_t height, lv_imgfont_get_path_cb_t path_cb, void *user_data)`

Creates a image font with info parameter specified.

Parametri

- **height** -- font size
- **path_cb** -- a function to get the image path name of character.
- **user_data** -- pointer to user data

Ritorna

pointer to the new imgfont or NULL if create error.

`void lv_imgfont_destroy(lv_font_t *font)`

Destroy a image font that has been created.

Parametri

- font** -- pointer to image font handle.

14.18 indev

14.18.1 lv_gridnav.h

Enum

enum **lv_gridnav_ctrl_t**

Values:

enumerator **LV_GRIDNAV_CTRL_NONE**

enumerator **LV_GRIDNAV_CTRL_ROLLOVER**

If there is no next/previous object in a direction, the focus goes to the object in the next/previous row (on left/right keys) or first/last row (on up/down keys)

enumerator **LV_GRIDNAV_CTRL_SCROLL_FIRST**

If an arrow is pressed and the focused object can be scrolled in that direction then it will be scrolled instead of going to the next/previous object. If there is no more room for scrolling the next/previous object will be focused normally

enumerator **LV_GRIDNAV_CTRL_HORIZONTAL_MOVE_ONLY**

Only use left/right keys for grid navigation. Up/down key events will be sent to the focused object.

enumerator **LV_GRIDNAV_CTRL_VERTICAL_MOVE_ONLY**

Only use up/down keys for grid navigation. Left/right key events will be sent to the focused object.

Funzioni

void **lv_gridnav_add**(lv_obj_t *obj, *lv_gridnav_ctrl_t* ctrl)

Add grid navigation feature to an object. It expects the children to be arranged into a grid-like layout. Although it's not required to have pixel perfect alignment. This feature makes possible to use keys to navigate among the children and focus them. The keys other than arrows and press/release related events are forwarded to the focused child.

Parametri

- **obj** -- pointer to an object on which navigation should be applied.
- **ctrl** -- control flags from *lv_gridnav_ctrl_t*.

void **lv_gridnav_remove**(lv_obj_t *obj)

Remove the grid navigation support from an object

Parametri

- **obj** -- pointer to an object

void **lv_gridnav_set_focused**(lv_obj_t *cont, lv_obj_t *to_focus, *lv_anim_enable_t* anim_en)

Manually focus an object on gridnav container

Parametri

- **cont** -- pointer to a gridnav container
- **to_focus** -- pointer to an object to focus
- **anim_en** -- LV_ANIM_ON/OFF

14.18.2 lv_indev.h

TypeDef

typedef void (***lv_indev_read_cb_t**)(lv_indev_t *indev, *lv_indev_data_t* *data)

typedef *lv_key_t* (**lv_indev_key_remap_cb_t*)(*lv_indev_t* **indev*, *lv_key_t* *key*)

Indev key remapping callback

Enum

enum ***lv_indev_type_t***

Possible input device types

Values:

enumerator ***LV_INDEV_TYPE_NONE***

Uninitialized state

enumerator ***LV_INDEV_TYPE_POINTER***

Touch pad, mouse, external button

enumerator ***LV_INDEV_TYPE_KEYPAD***

Keypad or keyboard

enumerator ***LV_INDEV_TYPE_BUTTON***

External (hardware button) which is assigned to a specific point of the screen

enumerator ***LV_INDEV_TYPE_ENCODER***

Encoder with only Left, Right turn and a Button

enum ***lv_indev_state_t***

States for input devices

Values:

enumerator ***LV_INDEV_STATE_RELEASED***

enumerator ***LV_INDEV_STATE_PRESSED***

enum ***lv_indev_mode_t***

Values:

enumerator ***LV_INDEV_MODE_NONE***

enumerator ***LV_INDEV_MODE_TIMER***

enumerator ***LV_INDEV_MODE_EVENT***

enum ***lv_indev_gesture_type_t***

Values:

enumerator ***LV_INDEV_GESTURE_NONE***

enumerator ***LV_INDEV_GESTURE_PINCH***

enumerator **LV_INDEV_GESTURE_SWIPE**
 enumerator **LV_INDEV_GESTURE_ROTATE**
 enumerator **LV_INDEV_GESTURE_TWO_FINGERS_SWIPE**
 enumerator **LV_INDEV_GESTURE_SCROLL**
 enumerator **LV_INDEV_GESTURE_CNT**

Funzioni

lv_indev_t *lv_indev_create(void)

Create an indev

Ritorna

Pointer to the created indev or NULL when allocation failed

void lv_indev_delete(lv_indev_t *indev)

Remove the provided input device. Make sure not to use the provided input device afterwards anymore.

Parametri

indev -- pointer to delete

lv_indev_t *lv_indev_get_next(lv_indev_t *indev)

Get the next input device.

Parametri

indev -- pointer to the current input device. NULL to initialize.

Ritorna

the next input device or NULL if there are no more. Provide the first input device when the parameter is NULL

void lv_indev_read(lv_indev_t *indev)

Read data from an input device.

Parametri

indev -- pointer to an input device

void lv_indev_read_timer_cb(lv_timer_t *timer)

Called periodically to read the input devices

Parametri

timer -- pointer to a timer to read

void lv_indev_enable(lv_indev_t *indev, bool enable)

Enable or disable one or all input devices (default enabled)

Parametri

- **indev** -- pointer to an input device or NULL to enable/disable all of them
- **enable** -- true to enable, false to disable

lv_indev_t *lv_indev_active(void)

Get the currently processed input device. Can be used in action functions too.

Ritorna

pointer to the currently processed input device or NULL if no input device processing right now

void lv_indev_set_type(lv_indev_t *indev, lv_indev_type_t indev_type)

Set the type of an input device

Parametri

- **indev** -- pointer to an input device

- **indev_type** -- the type of the input device from `lv_indev_type_t` (`LV_INDEV_TYPE_...`)

`void lv_indev_set_read_cb(lv_indev_t *indev, lv_indev_read_cb_t read_cb)`

Set a callback function to read input device data to the indev

Parametri

- **indev** -- pointer to an input device
- **read_cb** -- pointer to callback function to read input device data

`void lv_indev_set_user_data(lv_indev_t *indev, void *user_data)`

Set user data to the indev

Parametri

- **indev** -- pointer to an input device
- **user_data** -- pointer to user data

`void lv_indev_set_driver_data(lv_indev_t *indev, void *driver_data)`

Set driver data to the indev

Parametri

- **indev** -- pointer to an input device
- **driver_data** -- pointer to driver data

`void lv_indev_set_display(lv_indev_t *indev, struct _lv_display_t *disp)`

Assign a display to the indev

Parametri

- **indev** -- pointer to an input device
- **disp** -- pointer to an display

`void lv_indev_set_long_press_time(lv_indev_t *indev, uint16_t long_press_time)`

Set long press time to indev

Parametri

- **indev** -- pointer to input device
- **long_press_time** -- time long press time in ms

`void lv_indev_set_long_press_repeat_time(lv_indev_t *indev, uint16_t long_press_repeat_time)`

Set long press repeat time to indev

Parametri

- **indev** -- pointer to input device
- **long_press_repeat_time** -- long press repeat time in ms

`void lv_indev_set_scroll_limit(lv_indev_t *indev, uint8_t scroll_limit)`

Set scroll limit to the input device

Parametri

- **indev** -- pointer to an input device
- **scroll_limit** -- the number of pixels to slide before actually drag the object

`void lv_indev_set_scroll_throw(lv_indev_t *indev, uint8_t scroll_throw)`

Set scroll throw slow-down to the indev. Greater value means faster slow-down

Parametri

- **indev** -- pointer to an input device
- **scroll_throw** -- the slow-down in [%]

`lv_indev_type_t lv_indev_get_type(const lv_indev_t *indev)`

Get the type of an input device

Parametri

- **indev** -- pointer to an input device

Ritorna

the type of the input device from `lv_hal_indev_type_t` (`LV_INDEV_TYPE_...`)

`lv_indev_read_cb_t lv_indev_get_read_cb(lv_indev_t *indev)`

Get the callback function to read input device data to the indev

Parametri
indev -- pointer to an input device

Ritorna
 Pointer to callback function to read input device data or NULL if indev is NULL

lv_indev_state_t lv_indev_get_state(const lv_indev_t *indev)

Get the indev state

Parametri
indev -- pointer to an input device

Ritorna
 Indev state or LV_INDEV_STATE_RELEASED if indev is NULL

lv_group_t *lv_indev_get_group(const lv_indev_t *indev)

Get the indev assigned group

Parametri
indev -- pointer to an input device

Ritorna
 Pointer to indev assigned group or NULL if indev is NULL

lv_display_t *lv_indev_get_display(const lv_indev_t *indev)

Get a pointer to the assigned display of the indev

Parametri
indev -- pointer to an input device

Ritorna
 pointer to the assigned display or NULL if indev is NULL

void *lv_indev_get_user_data(const lv_indev_t *indev)

Get a pointer to the user data of the indev

Parametri
indev -- pointer to an input device

Ritorna
 pointer to the user data or NULL if indev is NULL

void *lv_indev_get_driver_data(const lv_indev_t *indev)

Get a pointer to the driver data of the indev

Parametri
indev -- pointer to an input device

Ritorna
 pointer to the driver data or NULL if indev is NULL

bool lv_indev_get_press_moved(const lv_indev_t *indev)

Get whether indev is moved while pressed

Parametri
indev -- pointer to an input device

Ritorna
 true: indev is moved while pressed; false: indev is not moved while pressed

void lv_indev_reset(lv_indev_t *indev, lv_obj_t *obj)

Reset one or all input devices

Parametri

- **indev** -- pointer to an input device to reset or NULL to reset all of them
- **obj** -- pointer to an object which triggers the reset.

void lv_indev_stop_processing(lv_indev_t *indev)

Touch and key related events are sent to the input device first and to the widget after that. If this functions called in an indev event, the event won't be sent to the widget.

Parametri
indev -- pointer to an input device

void lv_indev_reset_long_press(lv_indev_t *indev)

Reset the long press state of an input device

Parametri

- indev** -- pointer to an input device

void lv_indev_set_cursor(lv_indev_t *indev, lv_obj_t *cur_obj)

Set a cursor for a pointer input device (for LV_INPUT_TYPE_POINTER and LV_INPUT_TYPE_BUTTON)

Parametri

- **indev** -- pointer to an input device
- **cur_obj** -- pointer to an object to be used as cursor

void lv_indev_set_group(lv_indev_t *indev, lv_group_t *group)

Set a destination group for a keypad input device (for LV_INDEV_TYPE_KEYPAD)

Parametri

- **indev** -- pointer to an input device
- **group** -- pointer to a group

void lv_indev_set_button_points(lv_indev_t *indev, const lv_point_t points[])

Set the an array of points for LV_INDEV_TYPE_BUTTON. These points will be assigned to the buttons to press a specific point on the screen

Parametri

- **indev** -- pointer to an input device
- **points** -- array of points

void lv_indev_get_point(const lv_indev_t *indev, lv_point_t *point)

Get the last point of an input device (for LV_INDEV_TYPE_POINTER and LV_INDEV_TYPE_BUTTON)

Parametri

- **indev** -- pointer to an input device
- **point** -- pointer to a point to store the result

lv_dir_t lv_indev_get_gesture_dir(const lv_indev_t *indev)

Get the current gesture direct

Parametri

- indev** -- pointer to an input device

Ritorna

current gesture direct

uint32_t lv_indev_get_key(const lv_indev_t *indev)

Get the last pressed key of an input device (for LV_INDEV_TYPE_KEYPAD)

Parametri

- indev** -- pointer to an input device

Ritorna

the last pressed key (0 on error)

uint8_t lv_indev_get_short_click_streak(const lv_indev_t *indev)

Get the counter for consecutive clicks within a short distance and time. The counter is updated before LV_EVENT_SHORT_CLICKED is fired.

Parametri

- indev** -- pointer to an input device

Ritorna

short click streak counter

lv_dir_t lv_indev_get_scroll_dir(const lv_indev_t *indev)

Check the current scroll direction of an input device (for LV_INDEV_TYPE_POINTER and LV_INDEV_TYPE_BUTTON)

Parametri

- indev** -- pointer to an input device

Ritorna

LV_DIR_NONE: no scrolling now LV_DIR_HOR/VER

lv_obj_t *lv_indev_get_scroll_obj(const lv_indev_t *indev)

Get the currently scrolled object (for LV_INDEV_TYPE_POINTER and LV_INDEV_TYPE_BUTTON)

Parametri

- indev** -- pointer to an input device

Ritorna

- pointer to the currently scrolled object or NULL if no scrolling by this indev

```
void lv_indev_get_vect(const lv_indev_t *indev, lv_point_t *point)
```

Get the movement vector of an input device (for LV_INDEV_TYPE_POINTER and LV_INDEV_TYPE_BUTTON)

Parametri

- indev** -- pointer to an input device
- point** -- pointer to a point to store the types.pointer.vector

```
lv_obj_t *lv_indev_get_cursor(lv_indev_t *indev)
```

Get the cursor object of an input device (for LV_INDEV_TYPE_POINTER only)

Parametri

- indev** -- pointer to an input device

Ritorna

- pointer to the cursor object

```
void lv_indev_wait_release(lv_indev_t *indev)
```

Do nothing until the next release

Parametri

- indev** -- pointer to an input device

```
lv_obj_t *lv_indev_get_active_obj(void)
```

Gets a pointer to the currently active object in the currently processed input device.

Ritorna

- pointer to currently active object or NULL if no active object

```
lv_timer_t *lv_indev_get_read_timer(lv_indev_t *indev)
```

Get a pointer to the indev read timer to modify its parameters with `lv_timer_...` functions.

Parametri

- indev** -- pointer to an input device

Ritorna

- pointer to the indev read refresher timer. (NULL on error)

```
void lv_indev_set_mode(lv_indev_t *indev, lv_indev_mode_t mode)
```

Set the input device's event model: event-driven mode or timer mode.

Parametri

- indev** -- pointer to an input device
- mode** -- the mode of input device

```
lv_indev_mode_t lv_indev_get_mode(lv_indev_t *indev)
```

Get the input device's running mode.

Parametri

- indev** -- pointer to an input device

Ritorna

- the running mode for the specified input device.

```
lv_obj_t *lv_indev_search_obj(lv_obj_t *obj, lv_point_t *point)
```

Search the most top, clickable object by a point

Parametri

- obj** -- pointer to a start object, typically the screen
- point** -- pointer to a point for searching the most top child

Ritorna

- pointer to the found object or NULL if there was no suitable object

```
void lv_indev_add_event_cb(lv_indev_t *indev, lv_event_cb_t event_cb, lv_event_code_t filter, void *user_data)
```

Add an event handler to the indev

Parametri

- indev** -- pointer to an indev

- **event_cb** -- an event callback
- **filter** -- event code to react or LV_EVENT_ALL
- **user_data** -- optional user_data

`uint32_t lv_indev_get_event_count(lv_indev_t *indev)`

Get the number of event attached to an indev

Parametri

indev -- pointer to an indev

Ritorna

 number of events

`lv_event_dsc_t *lv_indev_get_event_dsc(lv_indev_t *indev, uint32_t index)`

Get an event descriptor for an event

Parametri

- **indev** -- pointer to an indev
- **index** -- the index of the event

Ritorna

 the event descriptor

`bool lv_indev_remove_event(lv_indev_t *indev, uint32_t index)`

Remove an event

Parametri

- **indev** -- pointer to an indev
- **index** -- the index of the event to remove

Ritorna

 true: and event was removed; false: no event was removed

`uint32_t lv_indev_remove_event_cb_with_user_data(lv_indev_t *indev, lv_event_cb_t event_cb, void *user_data)`

Remove an event_cb with user_data

Parametri

- **indev** -- pointer to a indev
- **event_cb** -- the event_cb of the event to remove
- **user_data** -- user_data

Ritorna

 the count of the event removed

`lv_result_t lv_indev_send_event(lv_indev_t *indev, lv_event_code_t code, void *param)`

Send an event to an indev

Parametri

- **indev** -- pointer to an indev
- **code** -- an event code. LV_EVENT_...
- **param** -- optional param

Ritorna

 LV_RESULT_OK: indev wasn't deleted in the event.

`void lv_indev_set_key_remap_cb(lv_indev_t *indev, lv_indev_key_remap_cb_t remap_cb)`

Set key remapping callback (LV_INDEV_TYPE_KEYPAD)

Parametri

- **indev** -- pointer to an indev
- **remap_cb** -- remapping function callback. Use NULL to disable callback.

`struct lv_indev_data_t`

#include <lv_indev.h> Data structure passed to an input driver to fill

Membri Pubblici

lv_indev_gesture_type_t **gesture_type**[*LV_INDEV_GESTURE_CNT*]

void ***gesture_data**[*LV_INDEV_GESTURE_CNT*]

lv_indev_state_t **state**

LV_INDEV_STATE_RELEASED or LV_INDEV_STATE_PRESSED

lv_point_t **point**

For LV_INDEV_TYPE_POINTER the currently pressed point

uint32_t **key**

For LV_INDEV_TYPE_KEYPAD the currently pressed key

uint32_t **btn_id**

For LV_INDEV_TYPE_BUTTON the currently pressed button

int16_t **enc_diff**

For LV_INDEV_TYPE_ENCODER number of steps since the previous read

uint32_t **timestamp**

Initialized to *lv_tick_get()*. Driver may provide more accurate timestamp for buffered events

bool **continue_reading**

If set to true, the read callback is invoked again, unless the device is in event-driven mode

14.18.3 lv_indev_gesture.h

Typedef

typedef struct *lv_indev_gesture_recognizer* **lv_indev_gesture_recognizer_t**

typedef struct *lv_indev_touch_data* **lv_indev_touch_data_t**

typedef struct *lv_indev_gesture* **lv_indev_gesture_t**

typedef struct *lv_indev_gesture_configuration* **lv_indev_gesture_configuration_t**

typedef void (***lv_recognizer_func_t**)(*lv_indev_gesture_recognizer_t**, *lv_indev_touch_data_t**, uint16_t)

Enum

enum **lv_indev_gesture_state_t**

Values:

enumerator **LV_INDEV_GESTURE_STATE_NONE**

enumerator **LV_INDEV_GESTURE_STATE_ONGOING**

enumerator **LV_INDEV_GESTURE_STATE_RECOGNIZED**

enumerator **LV_INDEV_GESTURE_STATE_ENDED**

enumerator **LV_INDEV_GESTURE_STATE_CANCELED**

Funzioni

void lv_indev_gesture_init(lv_indev_t *indev)

Initialize this indev's recognizers. It specifies their recognizer functions

Parametri

- **indev** -- pointer to the indev containing the recognizers to initialize

void lv_indev_gesture_detect_pinch(lv_indev_gesture_recognizer_t *recognizer, lv_indev_touch_data_t *touches, uint16_t touch_cnt)

Pinch gesture recognizer function Will update the recognizer data

Parametri

- **recognizer** -- pointer to a gesture recognizer
- **touches** -- pointer to the first element of the collected touch events
- **touch_cnt** -- length of passed touch event array.

void lv_indev_gesture_detect_rotation(lv_indev_gesture_recognizer_t *recognizer, lv_indev_touch_data_t *touches, uint16_t touch_cnt)

Rotation gesture recognizer function Will update the recognizer data

Parametri

- **recognizer** -- pointer to a gesture recognizer
- **touches** -- pointer to the first element of the collected touch events
- **touch_cnt** -- length of passed touch event array.

void lv_indev_gesture_detect_two_fingersSwipe(lv_indev_gesture_recognizer_t *recognizer, lv_indev_touch_data_t *touches, uint16_t touch_cnt)

Two finger swipe gesture recognizer function Will update the recognizer data

Parametri

- **recognizer** -- pointer to a gesture recognizer
- **touches** -- pointer to the first element of the collected touch events
- **touch_cnt** -- length of passed touch event array.

void lv_indev_set_pinch_up_threshold(lv_indev_t *indev, float threshold)

Set the threshold for the pinch gesture scale up, when the scale factor of gesture reaches the threshold events get sent

Parametri

- **indev** -- pointer to the indev device containing the pinch recognizer
- **threshold** -- threshold for a pinch up gesture to be recognized

void lv_indev_set_pinch_down_threshold(lv_indev_t *indev, float threshold)

Set the threshold for the pinch gesture scale down, when the scale factor of gesture reaches the threshold events get sent

Parametri

- **indev** -- pointer to the indev device containing the pinch recognizer
- **threshold** -- threshold for a pinch down gesture to be recognized

void lv_indev_set_rotation_rad_threshold(lv_indev_t *indev, float threshold)

Set the rotation threshold in radian for the rotation gesture

Parametri

- **indev** -- pointer to the indev device containing the rotation recognizer

- **threshold** -- threshold in radian for a rotation gesture to be recognized

float lv_event_get_pinch_scale(lv_event_t *gesture_event)

Obtains the current scale of a pinch gesture

Parametri

gesture_event -- pointer to a gesture event

Ritorna

 the scale of the current gesture

float lv_event_get_rotation(lv_event_t *gesture_event)

Obtains the current angle in radian of a rotation gesture

Parametri

gesture_event -- pointer to a gesture event

Ritorna

 the rotation angle in radian of the current gesture

float lv_event_get_two_fingersSwipeDistance(lv_event_t *gesture_event)

Obtains the current distance in pixels of a two fingers swipe gesture, from the starting center

Parametri

gesture_event -- pointer to a gesture event

Ritorna

 the distance from the center, in pixels, of the current gesture

lv_dir_t lv_event_get_two_fingersSwipeDir(lv_event_t *gesture_event)

Obtains the current direction from the center of a two finger swipe

Parametri

gesture_event -- pointer to a gesture event

Ritorna

 the rotation angle in radian of the current gesture

void lv_indev_set_gesture_data(lv_indev_data_t *data, lv_indev_gesture_recognizer_t *recognizer, lv_indev_gesture_type_t type)

Sets the state of the recognizer to a indev data structure, it is usually called from the indev read callback

Parametri

- **data** -- the indev data
- **recognizer** -- pointer to a gesture recognizer

void lv_indev_get_gesture_center_point(lv_indev_gesture_recognizer_t *recognizer, lv_point_t *point)

Obtains the center point of a gesture

Parametri

- **gesture_event** -- pointer to a gesture recognizer event
- **point** -- pointer to a point

lv_indev_gesture_state_t lv_event_get_gesture_state(lv_event_t *gesture_event, lv_indev_gesture_type_t type)

Obtains the current state of the gesture recognizer attached to an event

Parametri

gesture_event -- pointer to a gesture recognizer event

Ritorna

 current state of the gesture recognizer

lv_indev_gesture_type_t lv_event_get_gesture_type(lv_event_t *gesture_event)

Obtains the current event type of the gesture recognizer attached to an event

Parametri

gesture_event -- pointer to a gesture recognizer event

Ritorna

 current event type of the gesture recognizer

void lv_indev_get_gesture_primary_point(lv_indev_gesture_recognizer_t *recognizer, lv_point_t *point)

Obtains the coordinates of the current primary point

Parametri

- **recognizer** -- pointer to a gesture recognizer
- **point** -- pointer to a point

`bool lv_indev_recognizer_is_active(lv_indev_gesture_recognizer_t *recognizer)`

Allows to determine if there is an are ongoing gesture

Parametri

recognizer -- pointer to a gesture recognizer

Ritorna

false if there are no contact points, or the gesture has ended - true otherwise

`void lv_indev_gesture_recognizers_update(lv_indev_t *indev, lv_indev_touch_data_t *touches, uint16_t touch_cnt)`

Update the recognizers. It execute the recognizers functions and checks for LV_GESTURE_STATE_RECOGNIZED or LV_GESTURE_STATE_ENDED gestures. To be called in the indev read_cb.

Parametri

- **indev** -- pointer to the indev containing from which the reconizer need an update
- **touches** -- indev touch data array, containing the last touch data from indev since the last recognizers update
- **touch_cnt** -- number of indev touch data in touches

`void lv_indev_gesture_recognizers_set_data(lv_indev_t *indev, lv_indev_data_t *data)`

Set the `lv_indev_data_t` struct from the recognizer data. To be called in the indev read_cb.

`struct lv_indev_touch_data`

Membri Pubblici

`lv_point_t point`

`lv_indev_state_t state`

`uint8_t id`

`uint32_t timestamp`

`struct lv_indev_gesture_recognizer`

Membri Pubblici

`lv_indev_gesture_type_t type`

`lv_indev_gesture_state_t state`

`lv_indev_gesture_t *info`

`float scale`

`float rotation`

`float distance`

```
float speed

lv_dir_t two_fingersSwipeDir

lv_indev_gesture_configuration_t *config

lv_recognizer_func_t recog_fn
```

14.18.4 lv_indev_gesture_private.h

Define

LV_GESTURE_MAX_POINTS

Typedef

```
typedef struct lv_indev_gesture_motion lv_indev_gesture_motion_t
```

```
struct lv_indev_gesture_motion
```

Membri Pubblici

int8_t finger

lv_point_t start_point

lv_point_t point

lv_indev_state_t state

```
struct lv_indev_gesture
```

Membri Pubblici

lv_indev_gesture_motion_t motions[2]

lv_point_t center

float scale

float p_scale

float scale_factors_x[2]

float scale_factors_y[2]

```

float delta_x

float delta_y

float p_delta_x

float p_delta_y

float rotation

float p_rotation

uint8_t finger_cnt

struct lv_indev_gesture_configuration

```

Membri Pubblici

```

float pinch_up_threshold

float pinch_down_threshold

float rotation_angle_rad_threshold

```

14.18.5 lv_indev_private.h**Define****LV_INDEV_VECT_HIST_SIZE****Funzioni****lv_obj_t *lv_indev_find_scroll_obj(lv_indev_t *indev)**

Find a scrollable object based on the current scroll vector in the indev. In handles scroll propagation to the parent if needed, and scroll directions too.

Parametri

- indev** -- pointer to an indev

Ritorna

- the found scrollable object or NULL if not found.

struct _lv_indev_t**Membri Pubblici****lv_indev_type_t type**

- Input device type

lv_indev_read_cb_t read_cb

- Function pointer to read input device data.

lv_indev_state_t state

Current state of the input device.

lv_indev_state_t prev_state

Previous state of the input device.

lv_indev_mode_t mode***uint8_t long_pr_sent******uint8_t reset_query******uint8_t enabled******uint8_t wait_until_release******uint8_t stop_processing_query******uint32_t timestamp***

Timestamp of last event

uint32_t pr_timestamp

Pressed time stamp

uint32_t longpr_rep_timestamp

Long press repeat time stamp

void *driver_data***void *user_data***

Pointer to the assigned display

lv_display_t *disp

Timer to periodically read the input device

lv_timer_t *read_timer

Number of pixels to slide before actually drag the object

uint8_t scroll_limit

Drag throw slow-down in [%]. Greater value means faster slow-down

uint8_t scroll_throw

At least this difference should be between two points to evaluate as gesture

uint8_t gesture_min_velocity

At least this difference should be to send a gesture

uint8_t gesture_limit

Long press time in milliseconds

uint16_t long_press_time

Repeated trigger period in long press [ms]

uint16_t long_press_repeat_time

Rotary diff count will be multiplied by this value and divided by 256

int32_t rotary_sensitivity***lv_point_t* act_point**

Current point of input device.

***lv_point_t* last_point**

Last point of input device.

***lv_point_t* last_raw_point**

Last point read from read_cb.

***lv_point_t* vect**

Difference between act_point and last_point.

lv_point_t* vect_hist[8]*uint32_t vect_hist_timestamp[8]****uint8_t vect_hist_index*****lv_point_t* scroll_sum*****lv_point_t* scroll_throw_vect*****lv_point_t* scroll_throw_vect_ori****lv_obj_t *act_obj****lv_obj_t *last_obj****lv_obj_t *scroll_obj****lv_obj_t *last_pressed****lv_obj_t *last_hovered*****lv_area_t* scroll_area**

```

lv_point_t gesture_sum

int32_t diff

uint8_t short_click_streak

lv_point_t last_short_click_point

uint32_t last_short_click_timestamp

uint8_t scroll_dir

uint8_t gesture_dir

uint8_t gesture_sent

uint8_t press_moved

uint8_t pressed

struct _lv_indev_t pointer

lv_indev_state_t last_state

uint32_t last_key

struct _lv_indev_t keypad

lv_obj_t*cursor  

    Cursor for LV_INPUT_TYPE_POINTER

lv_group_t*group  

    Keypad destination group

const lv_point_t *btn_points  

    Array points assigned to the button ()screen will be pressed here by the buttons

lv_event_list_t event_list

lv_anim_t*scroll_throw_anim  

    Key remapping callback

lv_indev_key_remap_cb_t key_remap_cb

lv_indev_gesture_recognizer_t recognizers[LV_INDEV_GESTURE_CNT]

```

```
lv_indev_gesture_type_t cur_gesture

void *gesture_data[LV_INDEV_GESTURE_CNT]

lv_indev_gesture_type_t gesture_type[LV_INDEV_GESTURE_CNT]
```

14.18.6 lv_indev_scroll.h

Funzioni

void lv_indev_scroll_handler(lv_indev_t *indev)
Handle scrolling. Called by LVGL during input device processing

Parametri

- indev** -- pointer to an input device

void lv_indev_scroll_throw_handler(lv_indev_t *indev)
Handle throwing after scrolling. Called by LVGL during input device processing

Parametri

- indev** -- pointer to an input device

int32_t lv_indev_scroll_throw_predict(lv_indev_t *indev, lv_dir_t dir)
Predict where would a scroll throw end

Parametri

- **indev** -- pointer to an input device
- **dir** -- LV_DIR_VER or LV_DIR_HOR

Ritorna

the difference compared to the current position when the throw would be finished

void lv_indev_scroll_get_snap_dist(lv_obj_t *obj, lv_point_t *p)
Get the distance of the nearest snap point

Parametri

- **obj** -- the object on which snap points should be found
- **p** -- save the distance of the found snap point there

14.19 layouts

14.19.1 lv_layout.h

Typedef

typedef void (*lv_layout_update_cb_t)(lv_obj_t*, void *user_data)

Enum

enum lv_layout_t

Values:

enumerator **LV_LAYOUT_NONE**

enumerator **LV_LAYOUT_FLEX**

enumerator **LV_LAYOUT_GRID**

enumerator **LV_LAYOUT_LAST**

Funzioni

```
uint32_t lv_layout_register(lv_layout_update_cb_t cb, void *user_data)
```

Register a new layout

Parametri

- **cb** -- the layout update callback
- **user_data** -- custom data that will be passed to cb

Ritorna

the ID of the new layout

14.19.2 lv_layout_private.h**Funzioni**

```
void lv_layout_init(void)
```

```
void lv_layout_deinit(void)
```

```
void lv_layout_apply(lv_obj_t *obj)
```

Update the layout of a widget

Parametri

obj -- pointer to a widget

```
struct lv_layout_dsc_t
```

Membri Pubblici

lv_layout_update_cb_t **cb**

void ***user_data**

14.19.3 flex

lv_flex.h

Define

LV_FLEX_COLUMN

LV_FLEX_WRAP

LV_FLEX_REVERSE

Enum

```
enum lv_flex_align_t
```

Values:

enumerator **LV_FLEX_ALIGN_START**

enumerator **LV_FLEX_ALIGN_END**

enumerator **LV_FLEX_ALIGN_CENTER**

enumerator **LV_FLEX_ALIGN_SPACE_EVENLY**

enumerator **LV_FLEX_ALIGN_SPACE_AROUND**

enumerator **LV_FLEX_ALIGN_SPACE_BETWEEN**

enum **lv_flex_flow_t**

Values:

enumerator **LV_FLEX_FLOW_ROW**

enumerator **LV_FLEX_FLOW_COLUMN**

enumerator **LV_FLEX_FLOW_ROW_WRAP**

enumerator **LV_FLEX_FLOW_ROW_REVERSE**

enumerator **LV_FLEX_FLOW_ROW_WRAP_REVERSE**

enumerator **LV_FLEX_FLOW_COLUMN_WRAP**

enumerator **LV_FLEX_FLOW_COLUMN_REVERSE**

enumerator **LV_FLEX_FLOW_COLUMN_WRAP_REVERSE**

Funzioni

void **lv_flex_init**(void)

Initialize a flex layout to default values

void **lv_obj_set_flex_flow**(lv_obj_t *obj, *lv_flex_flow_t* flow)

Set how the item should flow

Parametri

- **obj** -- pointer to an object. The parent must have flex layout else nothing will happen.
- **flow** -- an element of *lv_flex_flow_t*.

void **lv_obj_set_flex_align**(lv_obj_t *obj, *lv_flex_align_t* main_place, *lv_flex_align_t* cross_place, *lv_flex_align_t* track_cross_place)

Set how to place (where to align) the items and tracks

Parametri

- **obj** -- pointer to an object. The parent must have flex layout else nothing will happen.
- **main_place** -- where to place the items on main axis (in their track). Any value of *lv_flex_align_t*.
- **cross_place** -- where to place the item in their track on the cross axis. LV_FLEX_ALIGN_START/END/CENTER
- **track_cross_place** -- where to place the tracks in the cross direction. Any value of *lv_flex_align_t*.

void **lv_obj_set_flex_grow**(lv_obj_t *obj, uint8_t grow)

Sets the width or height (on main axis) to grow the object in order fill the free space

Parametri

- **obj** -- pointer to an object. The parent must have flex layout else nothing will happen.

- **grow** -- a value to set how much free space to take proportionally to other growing items.

14.19.4 grid

lv_grid.h

Define

LV_GRID_FR(x)

Can be used track size to make the track fill the free space.

Parametri

- **x** -- how much space to take proportionally to other FR tracks

Ritorna

a special track size

LV_GRID_CONTENT

LV_GRID_TEMPLATE_LAST

Enum

enum **lv_grid_align_t**

Values:

enumerator **LV_GRID_ALIGN_START**

enumerator **LV_GRID_ALIGN_CENTER**

enumerator **LV_GRID_ALIGN_END**

enumerator **LV_GRID_ALIGN_STRETCH**

enumerator **LV_GRID_ALIGN_SPACE_EVENLY**

enumerator **LV_GRID_ALIGN_SPACE_AROUND**

enumerator **LV_GRID_ALIGN_SPACE_BETWEEN**

Funzioni

void lv_grid_init(void)

void lv_obj_set_grid_dsc_array(lv_obj_t *obj, const int32_t col_dsc[], const int32_t row_dsc[])

void lv_obj_set_grid_align(lv_obj_t *obj, lv_grid_align_t column_align, lv_grid_align_t row_align)

void lv_obj_set_grid_cell(lv_obj_t *obj, lv_grid_align_t column_align, int32_t col_pos, int32_t col_span, lv_grid_align_t row_align, int32_t row_pos, int32_t row_span)

Set the cell of an object. The object's parent needs to have grid layout, else nothing will happen

Parametri

- **obj** -- pointer to an object
- **column_align** -- the vertical alignment in the cell. **LV_GRID_START/END/CENTER/STRETCH**

- **col_pos** -- column ID
- **col_span** -- number of columns to take (≥ 1)
- **row_align** -- the horizontal alignment in the cell. LV_GRID_START/END/CENTER/STRETCH
- **row_pos** -- row ID
- **row_span** -- number of rows to take (≥ 1)

```
int32_t lv_grid_fr(uint8_t x)
```

Just a wrapper to LV_GRID_FR for bindings.

14.20 libs

14.20.1 barcode

lv_barcode.h

Enum

enum **lv_barcode_encoding_t**

Values:

enumerator **LV_BARCODE_ENCODING_CODE128_GS1**

Code 128 with GS1 encoding. Strips [FCN1] and spaces.

enumerator **LV_BARCODE_ENCODING_CODE128_RAW**

Code 128 with raw encoding.

Funzioni

lv_obj_t *lv_barcode_create(lv_obj_t *parent)

Create an empty barcode (an **lv_canvas**) object.

Parametri

parent -- point to an object where to create the barcode

Ritorna

pointer to the created barcode object

void lv_barcode_set_dark_color(lv_obj_t *obj, lv_color_t color)

Set the dark color of a barcode object

Parametri

- **obj** -- pointer to barcode object
- **color** -- dark color of the barcode

void lv_barcode_set_light_color(lv_obj_t *obj, lv_color_t color)

Set the light color of a barcode object

Parametri

- **obj** -- pointer to barcode object
- **color** -- light color of the barcode

void lv_barcode_set_scale(lv_obj_t *obj, uint16_t scale)

Set the scale of a barcode object

Parametri

- **obj** -- pointer to barcode object
- **scale** -- scale factor

void lv_barcode_set_direction(lv_obj_t *obj, lv_dir_t direction)

Set the direction of a barcode object

Parametri

- **obj** -- pointer to barcode object

- **direction** -- draw direction (LV_DIR_HOR or LV_DIR_VER)

void lv_barcode_set_tiled(lv_obj_t *obj, bool tiled)

Set the tiled mode of a barcode object

Parametri

- **obj** -- pointer to barcode object
- **tiled** -- true: tiled mode, false: normal mode (default)

void lv_barcode_set_encoding(lv_obj_t *obj, lv_barcode_encoding_t encoding)

Set the encoding of a barcode object

Parametri

- **obj** -- pointer to barcode object
- **encoding** -- encoding (default is LV_BARCODE_CODE128_GS1)

lv_result_t lv_barcode_update(lv_obj_t *obj, const char *data)

Set the data of a barcode object

Parametri

- **obj** -- pointer to barcode object
- **data** -- data to display

Ritorna

LV_RESULT_OK: if no error; LV_RESULT_INVALID: on error

lv_color_t lv_barcode_get_dark_color(lv_obj_t *obj)

Get the dark color of a barcode object

Parametri

- **obj** -- pointer to barcode object

Ritorna

dark color of the barcode

lv_color_t lv_barcode_get_light_color(lv_obj_t *obj)

Get the light color of a barcode object

Parametri

- **obj** -- pointer to barcode object

Ritorna

light color of the barcode

uint16_t lv_barcode_get_scale(lv_obj_t *obj)

Get the scale of a barcode object

Parametri

- **obj** -- pointer to barcode object

Ritorna

scale factor

lv_barcode_encoding_t lv_barcode_get_encoding(const lv_obj_t *obj)

Get the encoding of a barcode object

Parametri

- **obj** -- pointer to barcode object

Ritorna

encoding

Variabili

const lv_obj_class_t **lv_barcode_class**

lv_barcode_private.h

struct **_lv_barcode_t**

Membri Pubblici

```
lv_canvas_t canvas  

lv_color_t dark_color  

lv_color_t light_color  

uint16_t scale  

lv_dir_t direction  

bool tiled  

lv_barcode_encoding_t encoding
```

14.20.2 bin_decoder**lv_bin_decoder.h****Funzioni**

void **lv_bin_decoder_init**(void)

Initialize the binary image decoder module

lv_result_t **lv_bin_decoder_info**(lv_image_decoder_t *decoder, lv_image_decoder_dsc_t *dsc,
lv_image_header_t *header)

Get info about a lvgl binary image

Parametri

- **decoder** -- the decoder where this function belongs
- **dsc** -- image descriptor containing the source and type of the image and other info.
- **header** -- store the image data here

Ritorna

LV_RESULT_OK: the info is successfully stored in header; LV_RESULT_INVALID:
unknown format or other error.

lv_result_t **lv_bin_decoder_get_area**(lv_image_decoder_t *decoder, lv_image_decoder_dsc_t *dsc, const
lv_area_t *full_area, *lv_area_t* *decoded_area)

lv_result_t **lv_bin_decoder_open**(lv_image_decoder_t *decoder, lv_image_decoder_dsc_t *dsc)

Open a lvgl binary image

Parametri

- **decoder** -- the decoder where this function belongs
- **dsc** -- pointer to decoder descriptor. **src**, **style** are already initialized in it.

Ritorna

LV_RESULT_OK: the info is successfully stored in header; LV_RESULT_INVALID:
unknown format or other error.

void **lv_bin_decoder_close**(lv_image_decoder_t *decoder, lv_image_decoder_dsc_t *dsc)

Close the pending decoding. Free resources etc.

Parametri

- **decoder** -- pointer to the decoder the function associated with
- **dsc** -- pointer to decoder descriptor

14.20.3 bmp

lv_bmp.h

Funzioni

void **lv_bmp_init**(void)

void **lv_bmp_deinit**(void)

14.20.4 ffmpeg

lv_ffmpeg.h

Enum

enum **lv_ffmpeg_player_cmd_t**

Values:

enumerator **LV_FFMPEG_PLAYER_CMD_START**

enumerator **LV_FFMPEG_PLAYER_CMD_STOP**

enumerator **LV_FFMPEG_PLAYER_CMD_PAUSE**

enumerator **LV_FFMPEG_PLAYER_CMD_RESUME**

enumerator **LV_FFMPEG_PLAYER_CMD_LAST**

Funzioni

void **lv_ffmpeg_init**(void)

Register FFMPEG image decoder

void **lv_ffmpeg_deinit**(void)

De-initialize FFMPEG image decoder

int **lv_ffmpeg_get_frame_num**(const char *path)

Get the number of frames contained in the file

Parametri

path -- image or video file name

Ritorna

Number of frames, less than 0 means failed

lv_obj_t ***lv_ffmpeg_player_create**(lv_obj_t *parent)

Create ffmpeg_player object

Parametri

parent -- pointer to an object, it will be the parent of the new player

Ritorna

pointer to the created ffmpeg_player

lv_result_t **lv_ffmpeg_player_set_src**(lv_obj_t *obj, const char *path)

Set the path of the file to be played.

Parametri

- **obj** -- pointer to a ffmpeg_player object

- **path** -- video file path

Ritorna

LV_RESULT_OK: no error; LV_RESULT_INVALID: can't get the info.

```
void lv_ffmpeg_player_set_cmd(lv_obj_t *obj, lv_ffmpeg_player_cmd_t cmd)
```

Set command control video player

Parametri

- **obj** -- pointer to a ffmpeg_player object
- **cmd** -- control commands

```
void lv_ffmpeg_player_set_auto_restart(lv_obj_t *obj, bool en)
```

Set the video to automatically replay

Parametri

- **obj** -- pointer to a ffmpeg_player object
- **en** -- true: enable the auto restart

```
void lv_ffmpeg_player_set_decoder(lv_obj_t *obj, const char *decoder_name)
```

Set the video decoder

Parametri

- **obj** -- pointer to a ffmpeg_player object
- **decoder_name** -- decoder name

Variabili

```
const lv_obj_class_t lv_ffmpeg_player_class
```

lv_ffmpeg_private.h

```
struct _lv_ffmpeg_player_t
```

Membri Pubblici

```
lv_image_t img
```

```
lv_timer_t *timer
```

```
lv_image_dsc_t imgdsc
```

```
bool auto_restart
```

```
struct ffmpeg_context_s *ffmpeg_ctx
```

```
const char *decoder_name
```

14.20.5 freetype

lv_freetype.h

Define

```
LV_FREETYPE_F26DOT6_TO_INT(x)
```

```
LV_FREETYPE_F26DOT6_TO_FLOAT(x)
```

```
FT_FONT_STYLE_NORMAL
```

```
FT_FONT_STYLE_ITALIC
```

FT_FONT_STYLE_BOLD

Typedef

typedef *lv_freetype_font_style_t* **LV_FT_FONT_STYLE**

typedef void ***lv_freetype_outline_t**

typedef const char **lv_freetype_font_src_t**

Enum

enum **lv_freetype_font_style_t**

Values:

enumerator **LV_FREETYPE_FONT_STYLE_NORMAL**

enumerator **LV_FREETYPE_FONT_STYLE_ITALIC**

enumerator **LV_FREETYPE_FONT_STYLE_BOLD**

enum **lv_freetype_font_render_mode_t**

Values:

enumerator **LV_FREETYPE_FONT_RENDER_MODE_BITMAP**

enumerator **LV_FREETYPE_FONT_RENDER_MODE_OUTLINE**

enum **lv_freetype_outline_type_t**

Values:

enumerator **LV_FREETYPE_OUTLINE_END**

enumerator **LV_FREETYPE_OUTLINE_MOVE_TO**

enumerator **LV_FREETYPE_OUTLINE_LINE_TO**

enumerator **LV_FREETYPE_OUTLINE_CUBIC_TO**

enumerator **LV_FREETYPE_OUTLINE_CONIC_TO**

enumerator **LV_FREETYPE_OUTLINE_BORDER_START**

Funzioni

`lv_result_t lv_freetype_init(uint32_t max_glyph_cnt)`

Initialize the freetype library.

Ritorna

LV_RESULT_OK on success, otherwise LV_RESULT_INVALID.

`void lv_freetype_uninit(void)`

Uninitialize the freetype library

`void lv_freetype_init_font_info(lv_font_info_t *font_info)`

Initialize a font info structure.

Parametri

`font_info` -- font info structure to be initialized.

`lv_font_t *lv_freetype_font_create_with_info(const lv_font_info_t *font_info)`

Create a freetype font with a font info structure.

Parametri

`font_info` -- font info structure.

Ritorna

Created font, or NULL on failure.

`lv_font_t *lv_freetype_font_create(const char *pathname, lv_freetype_font_render_mode_t render_mode, uint32_t size, lv_freetype_font_style_t style)`

Create a freetype font.

Parametri

- `pathname` -- font file path.
- `render_mode` -- font render mode(see @lv_freetype_font_render_mode_t for details).
- `size` -- font size.
- `style` -- font style(see lv_freetype_font_style_t for details).

Ritorna

Created font, or NULL on failure.

`void lv_freetype_font_delete(lv_font_t *font)`

Delete a freetype font.

Parametri

`font` -- freetype font to be deleted.

`void lv_freetype_outline_add_event(lv_event_cb_t event_cb, lv_event_code_t filter, void *user_data)`

Register a callback function to generate outlines for FreeType fonts.

Parametri

- `cb` -- The callback function to be registered.
- `user_data` -- User data to be passed to the callback function.

Ritorna

The ID of the registered callback function, or a negative value on failure.

`uint32_t lv_freetype_outline_get_scale(const lv_font_t *font)`

Get the scale of a FreeType font.

Parametri

`font` -- The FreeType font to get the scale of.

Ritorna

The scale of the FreeType font.

`bool lv_freetype_is_outline_font(const lv_font_t *font)`

Check if the font is an outline font.

Parametri

`font` -- The FreeType font.

Ritorna

Is outline font on success, otherwise false.

Variabili

```
const lv_font_class_t lv_freetype_font_class
```

lv_freetype_private.h

Define

```
FT_ERROR_MSG(msg, error_code)
```

```
LV_FREETYPE_FONT_DSC_MAGIC_NUM
```

```
LV_FREETYPE_FONT_DSC_HAS_MAGIC_NUM(dsc)
```

```
LV_ASSERT_FREETYPE_FONT_DSC(dsc)
```

```
FT_INT_TO_F26DOT6(x)
```

```
FT_F26DOT6_TO_INT(x)
```

```
FT_INT_TO_F16DOT16(x)
```

```
FT_F16DOT16_TO_INT(x)
```

Typedef

```
typedef struct lv_freetype_cache_node_t lv_freetype_cache_node_t
```

```
typedef struct lv_freetype_context_t lv_freetype_context_t
```

```
typedef struct lv_freetype_font_dsc_t lv_freetype_font_dsc_t
```

Funzioni

```
lv_freetype_context_t *lv_freetype_get_context(void)
```

Get the FreeType context.

Ritorna

A pointer to the FreeType context used by LittlevGL.

```
void lv_freetype_italic_transform(FT_Face face)
```

```
int32_t lv_freetype_italic_transform_on_pos(lv_point_t point)
```

```
lv_cache_t *lv_freetype_create_glyph_cache(uint32_t cache_size)
```

```
void lv_freetype_set_cbs_glyph(lv_freetype_font_dsc_t *dsc)
```

```
lv_cache_t *lv_freetype_create_draw_data_image(uint32_t cache_size)
```

```
void lv_freetype_set_cbs_image_font(lv_freetype_font_dsc_t *dsc)
```

```
lv_cache_t *lv_freetype_create_draw_data_outline(uint32_t cache_size)
```

```
void lv_freetype_set_cbs_outline_font(lv_freetype_font_dsc_t *dsc)
```

```
struct lv_freetype_outline_vector_t
```

Membri Pubblici

```
int32_t x
```

```
int32_t y
```

```
struct lv_freetype_outline_sizes_t
```

Membri Pubblici

```
int32_t segments_size
```

```
int32_t data_size
```

```
struct _lv_freetype_outline_event_param_t
```

Membri Pubblici

```
lv_freetype_outline_t outline
```

```
lv_freetype_outline_type_t type
```

```
lv_freetype_outline_vector_t to
```

```
lv_freetype_outline_vector_t control1
```

```
lv_freetype_outline_vector_t control2
```

```
lv_freetype_outline_sizes_t sizes
```

```
struct _lv_freetype_cache_node_t
```

Membri Pubblici

```
const char *pathname
```

```
lv_freetype_font_style_t style
```

```
lv_freetype_font_render_mode_t render_mode
```

```
uint32_t ref_size
```

Reference size for calculating outline glyph's real size.

```
FT_Face face
```

```
lv_mutex_t face_lock
```

```
bool face_has_kerning

lv_cache_t *glyph_cache

lv_cache_t *draw_data_cache

struct _lv_freetype_context_t
```

Membri Pubblici

```
FT_Library library

lv_ll_t face_id_ll

lv_event_cb_t event_cb

uint32_t max_glyph_cnt

lv_cache_t *cache_node_cache

struct _lv_freetype_font_dsc_t
```

Membri Pubblici

```
uint32_t magic_num

lv_font_t font

uint32_t size

lv_freetype_font_style_t style

lv_freetype_font_render_mode_t render_mode

lv_freetype_context_t *context

lv_freetype_cache_node_t *cache_node

lv_cache_entry_t *cache_node_entry

FTC_FaceID face_id

uint32_t outline_stroke_width

lv_font_kerning_t kerning
```

14.20.6 fsdrv

lv_fsdrv.h

Define

LV_FS_MAX_PATH_LEN

Funzioni

void **lv_fs_fatfs_init**(void)

void **lv_fs_stdio_init**(void)

void **lv_fs_posix_init**(void)

void **lv_fs_win32_init**(void)

void **lv_fs_memfs_init**(void)

void **lv_littlefs_set_handler**(struct lfs*)

void **lv_fs_littlefs_init**(void)

void **lv_fs_arduino_esp_littlefs_init**(void)

void **lv_fs_arduino_sd_init**(void)

void **lv_fs_uefi_init**(void)

void **lv_fs_frogfs_init**(void)

void **lv_fs_frogfs_deinit**(void)

lv_result_t **lv_fs_frogfs_register_blob**(const void *blob, const char *path_prefix)

Mount a frogfs blob at the path prefix. If there is a file "foo.txt" in the blob and the blob is registered with **path_prefix** as "my_blob", it can be opened later at path "my_blob/foo.txt".

Parametri

- **blob** -- a frogfs blob/image from mkfrogfs.py
- **path_prefix** -- a prefix that will be used to refer to this blob when accessing it.

Ritorna

LV_RESULT_OK or LV_RESULT_INVALID if there was an issue with the blob

void **lv_fs_frogfs_unregister_blob**(const char *path_prefix)

Unmount a frogfs blob that was previously mounted by **lv_fs_frogfs_register_blob**. All files and dirs should be closed before calling this.

Parametri

- **path_prefix** -- the path prefix that the blob was registered with

14.20.7 gif

lv_gif.h

Funzioni

lv_obj_t ***lv_gif_create**(lv_obj_t *parent)

Create a gif object

Parametri

- **parent** -- pointer to an object, it will be the parent of the new gif.

Ritorna

pointer to the gif obj

`void lv_gif_set_color_format(lv_obj_t *obj, lv_color_format_t color_format)`

Set the color format of the internally allocated framebuffer that the gif will be decoded to. The default is LV_COLOR_FORMAT_ARGB8888. Call this before `lv_gif_set_src` to avoid reallocating the framebuffer.

Parametri

- **obj** -- pointer to a gif object
- **color_format** -- the color format of the gif framebuffer

`void lv_gif_set_src(lv_obj_t *obj, const void *src)`

Set the gif data to display on the object

Parametri

- **obj** -- pointer to a gif object
- **src** -- 1) pointer to an `lv_image_dsc_t` descriptor (which contains gif raw data) or 2) path to a gif file (e.g. "S:/dir/anim.gif")

`void lv_gif_restart(lv_obj_t *obj)`

Restart a gif animation.

Parametri

- **obj** -- pointer to a gif obj

`void lv_gif_pause(lv_obj_t *obj)`

Pause a gif animation.

Parametri

- **obj** -- pointer to a gif obj

`void lv_gif_resume(lv_obj_t *obj)`

Resume a gif animation.

Parametri

- **obj** -- pointer to a gif obj

`bool lv_gif_is_loaded(lv_obj_t *obj)`

Checks if the GIF was loaded correctly.

Parametri

- **obj** -- pointer to a gif obj

`int32_t lv_gif_get_loop_count(lv_obj_t *obj)`

Get the loop count for the GIF.

Parametri

- **obj** -- pointer to a gif obj

`void lv_gif_set_loop_count(lv_obj_t *obj, int32_t count)`

Set the loop count for the GIF.

Parametri

- **obj** -- pointer to a gif obj
- **count** -- the loop count to set

Variabili

`const lv_obj_class_t lv_gif_class`

14.20.8 gltf

`gltf_data`

`lv_gltf_bind.h`

Define

`LV_GLTF_BIND_MAX_CHANNELS`

LV_GLTF_BIND_CHANNEL_0**LV_GLTF_BIND_CHANNEL_1****LV_GLTF_BIND_CHANNEL_2****LV_GLTF_BIND_CHANNEL_3**

Typedef

```
typedef struct _lv_gltf_bind lv_gltf_bind_t
```

Enum

enum **lv_gltf_bind_prop_t**

Values:

enumerator **LV_GLTF_BIND_PROP_VISIBILITY**

enumerator **LV_GLTF_BIND_PROP_POSITION**

enumerator **LV_GLTF_BIND_PROP_ROTATION**

enumerator **LV_GLTF_BIND_PROP_SCALE**

enumerator **LV_GLTF_BIND_PROP_BASE_COLOR**

enumerator **LV_GLTF_BIND_PROP_ALPHA_FACTOR**

enumerator **LV_GLTF_BIND_PROP_EMIS_COLOR**

enumerator **LV_GLTF_BIND_PROP_WORLD_POSITION**

enum **lv_gltf_bind_dir_t**

Values:

enumerator **LV_GLTF_BIND_DIR_READ**

enumerator **LV_GLTF_BIND_DIR_WRITE**

Funzioni

```
void lv_gltf_bind_set(lv_gltf_bind_t *bind, uint8_t channel, float data)
```

```
float lv_gltf_bind_get(lv_gltf_bind_t *bind, uint8_t channel)
```

```
void lv_gltf_bind_bind_clean(lv_gltf_bind_t *bind)
```

Reset the dirty flag for a given bind.

Parametri

- **bind** -- Pointer to the lv_gltf_bind_t to reset the dirty flag for.

```
lv_gltf_bind_t *lv_gltf_bind_add_by_index(lv_gltf_model_t *data, size_t index, lv_gltf_bind_prop_t
                                         which_prop, uint32_t data_mask, lv_gltf_bind_dir_t dir)
```

Add an bind to a GLTF data object by node index.

Parametri

- **gltf_data** -- Pointer to the lv_gltf_data_t object to which the bind will be added.
- **nodeIndex** -- The index of the node to bind.
- **which_prop** -- The property to bind.
- **data_mask** -- A mask indicating which data fields to bind.

Ritorna

Pointer to the newly created lv_gltf_bind_t object, or NULL if the operation failed.

```
lv_gltf_bind_t *lv_gltf_bind_add_by_ip(lv_gltf_model_t *data, const char *node_ip, lv_gltf_bind_prop_t
                                         which_prop, uint32_t data_mask, lv_gltf_bind_dir_t dir)
```

Add an bind to a GLTF data object by node IP address.

Parametri

- **gltf_data** -- Pointer to the lv_gltf_data_t object to which the bind will be added.
- **nodeIp** -- The IP address of the node to bind.
- **which_prop** -- The property to bind.
- **data_mask** -- A mask indicating which data fields to bind.

Ritorna

Pointer to the newly created lv_gltf_bind_t object, or NULL if the operation failed.

```
lv_gltf_bind_t *lv_gltf_bind_add_by_path(lv_gltf_model_t *data, const char *path, lv_gltf_bind_prop_t
                                         which_prop, uint32_t data_mask, lv_gltf_bind_dir_t dir)
```

Add an bind to a GLTF data object by node ID.

Parametri

- **gltf_data** -- Pointer to the lv_gltf_data_t object to which the bind will be added.
- **nodeId** -- The ID of the node to bind.
- **which_prop** -- The property to bind.
- **data_mask** -- A mask indicating which data fields to bind.

Ritorna

Pointer to the newly created lv_gltf_bind_t object, or NULL if the operation failed.

```
lv_result_t lv_gltf_bind_remove(lv_gltf_model_t *_data, lv_gltf_bind_t *bind)
```

Remove an bind from a GLTF data object.

Parametri

- **gltf_data** -- Pointer to the lv_gltf_data_t object from which the bind will be removed.
- **bind** -- The bind to be removed.
- **which_prop** -- The property to bind.
- **data_mask** -- A mask indicating which data fields to bind.

Ritorna

True on success, False on failure.

```
struct _lv_gltf_bind
```

Membri Pubblici

```
struct _lv_gltf_bind *next_bind
```

```
lv_gltf_bind_prop_t prop
```

`lv_gltf_bind_dir_t` **dir**

`uint32_t` **id**

`uint32_t` **data_mask**

`float` **data[4]**

`bool` **dirty**

lv_gltf_data_internal.h

Funzioni

`lv_gltf_uniform_locations_t` **lv_gltf_uniform_locations_create(GLuint program)**

`void lv_gltf_store_compiled_shader(lv_gltf_model_t *data, size_t identifier, lv_gltf_compiled_shader_t *shader)`

`lv_gltf_compiled_shader_t *lv_gltf_get_compiled_shader(lv_gltf_model_t *data, size_t identifier)`

`lv_gltf_model_t *lv_gltf_data_load_from_file(const char *file_path, lv_opengl_shader_manager_t *shader_manager)`

Load the gltf file at the specified filepath.

Parametri

- **gltf_path** -- The gltf filename
- **ret_data** -- Pointer to the data container that will be populated.
- **shaders** -- Pointer to the shader cache object this file uses.

`lv_gltf_model_t *lv_gltf_data_load_from_bytes(const uint8_t *data, size_t data_size, lv_opengl_shader_manager_t *shader_manager)`

Load the gltf file encoded within the supplied byte array.

Parametri

- **gltf_path** -- The gltf filename
- **gltf_data_size** -- if gltf_path is instead a byte array, pass the size of that array in through this variable (or 0 if it's a file path).
- **ret_data** -- Pointer to the data container that will be populated.
- **shaders** -- Pointer to the shader cache object this file uses.

`double lv_gltf_data_get_radius(lv_gltf_model_t *D)`

Retrieve the radius of the GLTF data object.

Parametri

`D` -- Pointer to the lv_gltf_data_t object from which to get the radius.

Ritorna

The radius of the GLTF data object.

`void lv_gltf_data_destroy(lv_gltf_model_t *_data)`

Destroy a GLTF data object and free associated resources.

Parametri

`_data` -- Pointer to the lv_gltf_data_t object to be destroyed.

`void lv_gltf_data_copy_bounds_info(lv_gltf_model_t *to, lv_gltf_model_t *from)`

Copy the bounds information from one GLTF data object to another.

Parametri

- **to** -- Pointer to the destination lv_gltf_data_t object.
- **from** -- Pointer to the source lv_gltf_data_t object.

```
void lv_gltf_data_rgb_to_bgr(uint8_t *pixel_buffer, size_t byte_total_count, bool has_alpha)
```

Swap the red and blue channels in a pixel buffer.

Parametri

- **pixel_buffer** -- Pointer to the pixel buffer containing the image data.
- **byte_total_count** -- The total number of bytes in the pixel buffer.
- **has_alpha** -- Flag indicating whether the pixel buffer includes an alpha channel.

```
struct IndirectDrawCommand
```

Membri Pubblici

GLuint **count**

GLuint **instanceCount**

GLuint **firstIndex**

GLint **baseVertex**

GLuint **baseInstance**

```
struct lv_gltf_primitive_t
```

Membri Pubblici

IndirectDrawCommand **draw**

GLenum **primitiveType**

GLenum **indexType**

GLuint **vertexArray**

GLuint **vertexBuffer**

GLuint **indexBuffer**

GLuint **materialUniformsIndex**

GLuint **albedoTexture**

GLuint **emissiveTexture**

GLuint **metalRoughTexture**

GLuint **occlusionTexture**

GLuint **normalTexture**

GLuint **diffuseTransmissionTexture**

GLuint **diffuseTransmissionColorTexture**

GLuint **transmissionTexture**

GLuint **transmissionTexcoordIndex**

GLint **baseColorTexcoordIndex**

GLint **emissiveTexcoordIndex**

GLint **metallicRoughnessTexcoordIndex**

GLint **occlusionTexcoordIndex**

GLint **normalTexcoordIndex**

GLint **diffuseTransmissionTexcoordIndex**

GLint **diffuseTransmissionColorTexcoordIndex**

GLint **clearcoatTexture**

GLint **clearcoatRoughnessTexture**

GLint **clearcoatNormalTexture**

GLint **clearcoatTexcoordIndex**

GLint **clearcoatRoughnessTexcoordIndex**

GLint **clearcoatNormalTexcoordIndex**

GLuint **thicknessTexture**

GLint **thicknessTexcoordIndex**

GLuint **diffuseTexture**

GLint **diffuseTexcoordIndex**

GLuint **specularGlossinessTexture**

GLint **specularGlossinessTexcoordIndex**
struct **lv_gltf_uniform_locations_t**

Membri Pubblici

GLint **camera**

GLint **view_projection_matrix**

GLint **model_matrix**

GLint **view_matrix**

GLint **projection_matrix**

GLint **env_intensity**

GLint **env_diffuse_sampler**

GLint **env_specular_sampler**

GLint **env_sheen_sampler**

GLint **env_ggx_lut_sampler**

GLint **env_charlie_lut_sampler**

GLint **env_mip_count**

GLint **exposure**

GLint **roughness_factor**

GLint **base_color_factor**

GLint **base_color_sampler**

GLint **base_color_uv_set**

GLint **base_color_uv_transform**

GLint **emissive_factor**

GLint **emissive_sampler**

GLint **emissive_uv_set**
GLint **emissive_uv_transform**
GLint **emissive_strength**
GLint **metallic_factor**
GLint **metallic_roughness_sampler**
GLint **metallic_roughness_uv_set**
GLint **metallic_roughness_uv_transform**
GLint **occlusion_strength**
GLint **occlusion_sampler**
GLint **occlusion_uv_set**
GLint **occlusion_uv_transform**
GLint **normal_scale**
GLint **normal_sampler**
GLint **normal_uv_set**
GLint **normal_uv_transform**
GLint **clearcoat_factor**
GLint **clearcoat_roughness_factor**
GLint **clearcoat_sampler**
GLint **clearcoat_uv_set**
GLint **clearcoat_uv_transform**
GLint **clearcoat_roughness_sampler**
GLint **clearcoat_roughness_uv_set**
GLint **clearcoat_roughness_uv_transform**

GLint **clearcoat_normal_scale**

GLint **clearcoat_normal_sampler**

GLint **clearcoat_normal_uv_set**

GLint **clearcoat_normal_uv_transform**

GLint **thickness**

GLint **thickness_sampler**

GLint **thickness_uv_set**

GLint **thickness_uv_transform**

GLint **diffuse_transmission_sampler**

GLint **diffuse_transmission_uv_set**

GLint **diffuse_transmission_uv_transform**

GLint **diffuse_transmission_color_sampler**

GLint **diffuse_transmission_color_uv_set**

GLint **diffuse_transmission_color_uv_transform**

GLint **sheen_color_factor**

GLint **sheen_roughness_factor**

GLint **specular_color_factor**

GLint **specular_factor**

GLint **diffuse_transmission_color_factor**

GLint **diffuse_transmission_factor**

GLint **ior**

GLint **alpha_cutoff**

GLint **dispersion**

```
GLint screen_size  
  
GLint transmission_factor  
  
GLint transmission_sampler  
  
GLint transmission_uv_set  
  
GLint transmission_uv_transform  
  
GLint transmission_framebuffer_sampler  
  
GLint transmission_framebuffer_size  
  
GLint attenuation_distance  
  
GLint attenuation_color  
  
GLint joints_sampler  
  
GLint diffuse_factor  
  
GLint glossiness_factor  
  
GLint diffuse_sampler  
  
GLint diffuse_uv_set  
  
GLint diffuse_uv_transform  
  
GLint specular_glossiness_sampler  
  
GLint specular_glossiness_uv_set  
  
GLint specular_glossiness_uv_transform  
  
struct lv_gltf_shaderset_t
```

Membri Pubblici

```
GLuint program  
  
uint32_t bg_program  
  
uint32_t vert
```

```
uint32_t frag

struct lv_gltf_compiled_shader_t
```

Membri Pubblici*lv_gltf_uniform_locations_t uniforms**lv_gltf_shaderset_t shaderset***lv_gltf_data_internal.hpp****lv_gltf_model.h****Funzioni****size_t lv_gltf_model_get_image_count(const lv_gltf_model_t *model)**

Get the number of images in the glTF model.

Images in glTF are used as sources for textures and can be stored either as external files or embedded as base64-encoded model within the glTF file.

Parametri**model** -- Pointer to the glTF model data structure**Ritorna**

Number of images in the model

size_t lv_gltf_model_get_texture_count(const lv_gltf_model_t *model)

Get the number of textures in the glTF model.

Textures define how images are sampled and applied to materials. Each texture references an image and may specify sampling parameters like filtering and wrapping modes.

Parametri**model** -- Pointer to the glTF model data structure**Ritorna**

Number of textures in the model

size_t lv_gltf_model_get_material_count(const lv_gltf_model_t *model)

Get the number of materials in the glTF model.

Materials define the visual appearance of mesh primitives, including properties like base color, metallic/roughness values, normal maps, and other surface characteristics.

Parametri**model** -- Pointer to the glTF model data structure**Ritorna**

Number of materials in the model

size_t lv_gltf_model_get_camera_count(const lv_gltf_model_t *model)

Get the number of cameras in the glTF model.

Cameras define viewpoints within the 3D scene and can be either perspective or orthographic. They are typically attached to nodes in the scene graph.

Parametri**model** -- Pointer to the glTF model data structure**Ritorna**

Number of cameras in the model

`size_t lv_gltf_model_get_node_count(const lv_gltf_model_t *model)`

Get the number of nodes in the glTF model.

Nodes form the scene graph hierarchy and can contain transformations, meshes, cameras, or other nodes as children. They define the spatial relationships between objects in the scene.

Parametri

`model` -- Pointer to the glTF model data structure

Ritorna

Number of nodes in the model

`size_t lv_gltf_model_get_mesh_count(const lv_gltf_model_t *model)`

Get the number of meshes in the glTF model.

Meshes contain the geometric model for 3D objects, including vertex positions, normals, texture coordinates, and indices. Each mesh can have multiple primitives with different materials.

Parametri

`model` -- Pointer to the glTF model data structure

Ritorna

Number of meshes in the model

`size_t lv_gltf_model_get_scene_count(const lv_gltf_model_t *model)`

Get the number of scenes in the glTF model.

Scenes define the root nodes of the scene graph. A glTF file can contain multiple scenes, though typically only one is designated as the default scene to be displayed.

Parametri

`model` -- Pointer to the glTF model data structure

Ritorna

Number of scenes in the model

`size_t lv_gltf_model_get_animation_count(const lv_gltf_model_t *model)`

Get the number of animations in the glTF model.

Animations define keyframe-based motion for nodes in the scene, including transformations like translation, rotation, and scaling over time.

Parametri

`model` -- Pointer to the glTF model data structure

Ritorna

Number of animations in the model

`lv_result_t lv_gltf_model_play_animation(lv_gltf_model_t *model, size_t index)`

Select and start playing an animation.

Parametri

- `model` -- Pointer to the glTF model structure
- `index` -- Animation number to start playing

Ritorna

`LV_RESULT_OK` if the animation was started else `LV_RESULT_INVALID`

`void lv_gltf_model_pause_animation(lv_gltf_model_t *model)`

Pause the current animation.

Parametri

`model` -- Pointer to the glTF model structure

`bool lv_gltf_model_is_animation_paused(lv_gltf_model_t *model)`

Check if an animation is currently being played.

Parametri

`model` -- Pointer to the glTF model structure

`size_t lv_gltf_model_get_animation(lv_gltf_model_t *model)`

Get the current selected animation. To see if it's playing see `lv_gltf_model_is_animation_paused`

Parametri

`model` -- Pointer to the glTF model structure

gltf_view**lv_gltf.h****Define****LV_GLTF_DEFAULT_CAMERA****LV_GLTF_ANIM_SPEED_TENTH****LV_GLTF_ANIM_SPEED_QUARTER****LV_GLTF_ANIM_SPEED_HALF****LV_GLTF_ANIM_SPEED_NORMAL****LV_GLTF_ANIM_SPEED_2X****LV_GLTF_ANIM_SPEED_3X****LV_GLTF_ANIM_SPEED_4X****Typedef**typedef [lv_3dplane_t](#) **lv_3dray_t****Enum****enum lv_gltf_aa_mode_t***Values:*enumerator **LV_GLTF_AA_MODE_OFF**enumerator **LV_GLTF_AA_MODE_ON**

Anti aliasing off

enumerator **LV_GLTF_AA_MODE_DYNAMIC**

Anti aliasing on

enum lv_gltf_bg_mode_t*Values:*enumerator **LV_GLTF_BG_MODE_SOLID**enumerator **LV_GLTF_BG_MODE_ENVIRONMENT**Solid background. Use `lv_obj_set_style_bg_color` to set the background color

Funzioni

`lv_obj_t *lv_gltf_create(lv_obj_t *parent)`

Create a GLTF viewer object

Parametri

- parent** -- pointer to the parent object

Ritorna

pointer to the created GLTF viewer object

`lv_gltf_model_t *lv_gltf_load_model_from_file(lv_obj_t *obj, const char *path)`

Load a GLTF model from a file into the viewer

Parametri

- obj** -- pointer to a GLTF viewer object
- path** -- file path to the GLTF model to load

Ritorna

pointer to the loaded GLTF model, or NULL on failure

`size_t lv_gltf_get_model_count(lv_obj_t *obj)`

Get the number of models loaded in the GLTF viewer

Parametri

- obj** -- pointer to a GLTF viewer object

Ritorna

the total number of models in the viewer

`lv_gltf_model_t *lv_gltf_get_model_by_index(lv_obj_t *obj, size_t id)`

Get a specific model by its index

Parametri

- obj** -- pointer to a GLTF viewer object
- id** -- index of the model to retrieve (0-based)

Ritorna

pointer to the model at the specified index, or NULL if index is invalid

`lv_gltf_model_t *lv_gltf_get_primary_model(lv_obj_t *obj)`

Get the primary model from the GLTF viewer. The primary model is the first model added to the viewer and can be used for camera selection and other primary operations

Parametri

- obj** -- pointer to a GLTF viewer object

Ritorna

pointer to the primary model, or NULL if no models are loaded

`void lv_gltf_set_yaw(lv_obj_t *obj, float yaw)`

Set the yaw (horizontal rotation) of the camera

Parametri

- obj** -- pointer to a GLTF viewer object
- yaw** -- yaw angle in degrees

`float lv_gltf_get_yaw(const lv_obj_t *obj)`

Get the yaw (horizontal rotation) of the camera

Parametri

- obj** -- pointer to a GLTF viewer object

Ritorna

yaw angle in degrees

`void lv_gltf_set_pitch(lv_obj_t *obj, float pitch)`

Set the pitch (vertical rotation) of the camera

Parametri

- obj** -- pointer to a GLTF viewer object
- pitch** -- pitch angle in degrees

`float lv_gltf_get_pitch(const lv_obj_t *obj)`

Get the pitch (vertical rotation) of the camera

Parametri

- **obj** -- pointer to a GLTF viewer object

Ritorna

- pitch angle in degrees

void **lv_gltf_set_distance**(lv_obj_t *obj, float value)

Set the camera distance from the focal point

Parametri

- **obj** -- pointer to a GLTF viewer object
- **value** -- distance value

float **lv_gltf_get_distance**(const lv_obj_t *obj)

Get the camera distance scale factor from the focal point

Parametri

- **obj** -- pointer to a GLTF viewer object

Ritorna

- distance scaling factor value

float **lv_gltf_get_world_distance**(const lv_obj_t *obj)

Get the camera distance from the focal point in world units

Parametri

- **obj** -- pointer to a GLTF viewer object

Ritorna

- world unit distance value

void **lv_gltf_set_fov**(lv_obj_t *obj, float value)

Set the field of view

Parametri

- **obj** -- pointer to a GLTF viewer object
- **value** -- vertical FOV in degrees. If zero, the view will be orthographic (non-perspective)

float **lv_gltf_get_fov**(const lv_obj_t *obj)

Get the field of view

Parametri

- **obj** -- pointer to a GLTF viewer object

Ritorna

- vertical FOV in degrees

void **lv_gltf_set_focal_x**(lv_obj_t *obj, float value)

Set the X coordinate of the camera focal point

Parametri

- **obj** -- pointer to a GLTF viewer object
- **value** -- X coordinate

float **lv_gltf_get_focal_x**(const lv_obj_t *obj)

Get the X coordinate of the camera focal point

Parametri

- **obj** -- pointer to a GLTF viewer object

Ritorna

- X coordinate

void **lv_gltf_set_focal_y**(lv_obj_t *obj, float value)

Set the Y coordinate of the camera focal point

Parametri

- **obj** -- pointer to a GLTF viewer object
- **value** -- Y coordinate

float **lv_gltf_get_focal_y**(const lv_obj_t *obj)

Get the Y coordinate of the camera focal point

Parametri

- obj** -- pointer to a GLTF viewer object

Ritorna

Y coordinate

```
void lv_gltf_set_focal_z(lv_obj_t *obj, float value)
```

Set the Z coordinate of the camera focal point

Parametri

- **obj** -- pointer to a GLTF viewer object
- **value** -- Z coordinate

```
float lv_gltf_get_focal_z(const lv_obj_t *obj)
```

Get the Z coordinate of the camera focal point

Parametri

- obj** -- pointer to a GLTF viewer object

Ritorna

Z coordinate

```
void lv_gltf_recenter(lv_obj_t *obj, lv_gltf_model_t *model)
```

Set the focal coordinates to the center point of the model object

Parametri

- **obj** -- pointer to a GLTF viewer object
- **model** -- a model attached to this viewer or NULL for the first model

```
void lv_gltf_set_camera(lv_obj_t *obj, uint32_t value)
```

Set the active camera index The camera is selected from the first GLTF model added to the viewer

 **Nota**

Values higher than the scene's camera count will be clamped to the maximum available camera index

Parametri

- **obj** -- pointer to a GLTF viewer object
- **value** -- camera index (0 for default camera, 1+ for scene camera index)

```
uint32_t lv_gltf_get_camera(const lv_obj_t *obj)
```

Get the active camera index

Parametri

- obj** -- pointer to a GLTF viewer object

Ritorna

active camera index

```
uint32_t lv_gltf_get_camera_count(const lv_obj_t *obj)
```

Get the number of cameras in the first GLTF model added to the viewer This count represents the valid range for the camera index parameter used with [lv_gltf_set_camera\(\)](#)

To get the camera count of other models, call lv_gltf_model_get_camera_count(model) directly with the specific model

Parametri

- obj** -- pointer to a GLTF viewer object

Ritorna

number of available cameras

```
void lv_gltf_set_animation_speed(lv_obj_t *obj, uint32_t value)
```

Set the animation speed ratio

The actual ratio is the value parameter / LV_GLTF_ANIM_SPEED_NORMAL Values greater than LV_GLTF_ANIM_SPEED_NORMAL will speed-up the animation Values less than LV_GLTF_ANIM_SPEED_NORMAL will slow down the animation

Parametri

- **obj** -- pointer to a GLTF viewer object
- **value** -- speed-up ratio of the animation

`uint32_t lv_gltf_get_animation_speed(const lv_obj_t *obj)`

Get the animation speed ratio

The actual ratio is the return value / LV_GLTF_ANIM_SPEED_NORMAL

Parametri

obj -- pointer to a GLTF viewer object

`void lv_gltf_set_background_mode(lv_obj_t *obj, lv_gltf_bg_mode_t value)`

Set the background mode

Parametri

- **obj** -- pointer to a GLTF viewer object
- **value** -- background mode

`lv_gltf_bg_mode_t lv_gltf_get_background_mode(const lv_obj_t *obj)`

Get the background mode

Parametri

obj -- pointer to a GLTF viewer object

Ritorna

background mode

`void lv_gltf_set_background_blur(lv_obj_t *obj, uint32_t value)`

Set the background blur amount

Parametri

- **obj** -- pointer to a GLTF viewer object
- **value** -- blur amount between 0 and 100

`uint32_t lv_gltf_get_background_blur(const lv_obj_t *obj)`

Get the background blur amount

Parametri

obj -- pointer to a GLTF viewer object

Ritorna

blur amount between 0 and 100

`void lv_gltf_set_env_brightness(lv_obj_t *obj, uint32_t value)`

Set the environmental brightness/power

Parametri

- **obj** -- pointer to a GLTF viewer object
- **value** -- brightness multiplier

`uint32_t lv_gltf_get_env_brightness(const lv_obj_t *obj)`

Get the environmental brightness/power

Parametri

obj -- pointer to a GLTF viewer object

Ritorna

brightness multiplier

`void lv_gltf_set_image_exposure(lv_obj_t *obj, float value)`

Set the image exposure level

Parametri

- **obj** -- pointer to a GLTF viewer object
- **value** -- exposure level (1.0 is default)

`float lv_gltf_get_image_exposure(const lv_obj_t *obj)`

Get the image exposure level

Parametri

obj -- pointer to a GLTF viewer object

Ritorna

exposure level

`void lv_gltf_set_antialiasing_mode(lv_obj_t *obj, lv_gltf_aa_mode_t value)`

Set the anti-aliasing mode

Parametri

- **obj** -- pointer to a GLTF viewer object
- **value** -- anti-aliasing mode

`lv_gltf_aa_mode_t lv_gltf_get_antialiasing_mode(const lv_obj_t *obj)`

Get the anti-aliasing mode

Parametri

- **obj** -- pointer to a GLTF viewer object

Ritorna

anti-aliasing mode

`lv_3dplane_t lv_gltf_get_current_view_plane(lv_obj_t *obj, float distance)`

Get a plane that faces the current view camera, centered some units in front of it

Parametri

- **obj** -- pointer to a GLTF viewer object
- **distance** -- distance in front of the camera to set the plane, in world units. see `lv_gltf_get_world_distance` to get the auto-distance

Ritorna

camera facing plane

`lv_3dplane_t lv_gltf_get_ground_plane(float elevation)`

Get a plane that faces upward, centered at a given height

Parametri

- **obj** -- pointer to a GLTF viewer object
- **elevation** -- elevation of the ground plane, in world units. this is usually zero

Ritorna

ground plane

`lv_3dray_t lv_gltf_get_ray_from_2d_coordinate(lv_obj_t *obj, const lv_point_t *screen_pos)`

Calculates a ray originating from the camera and passing through the specified mouse position on the screen.

Parametri

- **obj** -- pointer to a GLTF viewer object
- **screen_pos** -- screen co-ordinate, in pixels

Ritorna

mouse point ray

`lv_result_t lv_gltf_intersect_ray_with_plane(const lv_3dray_t *ray, const lv_3dplane_t *plane, lv_3dpoint_t *collision_point)`

Get the point that a given ray intersects with a specified plane at, if any

Parametri

- **ray** -- the intersection test ray
- **screen_y** -- the plane to test ray intersection with
- **collision_point** -- output `lv_3dpoint_t` holder, values are only valid if true is the return value

Ritorna

LV_RESULT_OK if intersection, LV_RESULT_INVALID if no intersection

`lv_result_t lv_gltf_world_to_screen(lv_obj_t *obj, const lv_3dpoint_t world_pos, lv_point_t *screen_pos)`

Get the screen position of a 3d point

Parametri

- **obj** -- pointer to a GLTF viewer object
- **world_pos** -- world position to convert
- **lv_point_t** -- the resulting point, in pixels. only valid if return value is true

Ritorna

LV_RESULT_OK if conversion valid, LV_RESULT_INVALID if no valid conversion

struct `lv_3dpoint_t`

Membri Pubblici`float x``float y``float z``struct lv_3dplane_t`**Membri Pubblici**`lv_3dpoint_t origin``lv_3dpoint_t direction``lv_gltf_view_internal.h`**Define**`LV_GLTF_DISTANCE_SCALE_FACTOR``struct lv_gltf_renwin_state_t`**Membri Pubblici**`uint32_t texture``uint32_t renderbuffer``unsigned framebuffer``struct lv_gltf_view_state_t`**Membri Pubblici**`lv_gltf_renwin_state_t render_state``lv_gltf_renwin_state_t opaque_render_state``uint64_t opaque_frame_buffer_width``uint64_t opaque_frame_buffer_height``uint32_t material_variant``bool render_state_ready`

```
bool render_opaque_buffer

struct lv_gltf_view_desc_t
```

Membri Pubblici

float pitch

float yaw

float distance

float fov

int32_t render_width

int32_t render_height

float focal_x

float focal_y

float focal_z

bool frame_was_antialiased

int32_t animation_speed_ratio

lv_gltf_aa_mode_t aa_mode

lv_gltf_bg_mode_t bg_mode

float blur_bg

float env_pow

How much to blur the environment background, between 0.0 and 1.0

float exposure

Environmental brightness, 1.8 by default

```
struct lv_opengl_state_t
```

Membri Pubblici

GLboolean blend_enabled

```
GLint blend_src  
  
GLint blend_dst  
  
GLint blend_equation  
  
GLfloat clear_depth  
  
GLfloat clear_color[4]  
  
struct lv_gltf_view_env_textures_t
```

Membri Pubblici

```
uint32_t diffuse  
  
uint32_t specular  
  
uint32_t sheen  
  
uint32_t ggxLut  
  
uint32_t charlie_lut  
  
uint32_t mip_count  
  
float ibl_intensity_scale  
  
float angle
```

assets

[lv_gltf_view_shader.h](#)

Funzioni

```
char *lv_gltf_view_shader_get_vertex(void)  
  
char *lv_gltf_view_shader_get_fragment(void)  
  
void lv_gltf_view_shader_get_src(lv_opengl_shader_portions_t *shaders)  
void lv_gltf_view_shader_get_env(lv_opengl_shader_portions_t *shaders)
```

ibl

[lv_gltf_ibl_sampler.h](#)

Funzioni

```
void lv_gltf_ibl_generate_env_textures(lv_gltf_view_env_textures_t *env, const char *env_file_path, float env_rotation)
```

```
struct lv_gltf_ibl_sampler_t
```

Membri Pubblici

```
uint32_t texture_size
```

```
float lod_bias
```

```
uint32_t lowest_mip_level
```

```
uint32_t input_texture_id
```

```
uint32_t cubemap_texture_id
```

```
uint32_t framebuffer
```

```
uint32_t mipmap_count
```

```
uint32_t lambertian_texture_id
```

```
uint32_t lambertian_sample_count
```

```
uint32_t ggx_sample_count
```

```
uint32_t ggx_texture_id
```

```
uint32_t sheen_texture_id
```

```
uint32_t sheen_sample_count
```

```
uint32_t ggxlut_texture_id
```

```
uint32_t lut_sample_count
```

```
uint32_t lut_resolution
```

```
uint32_t charlielut_texture_id
```

```
float scale_value
```

```
uint32_t mipmap_levels
```

```
lv_opengl_shader_manager_t shader_manager
```

```
uint32_t fullscreen_vertex_buffer
uint32_t fullscreen_tex_coord_buffer

struct lv_gltf_ibl_texture_t
```

Membri Pubblici

```
uint8_t *data
uint32_t internal_format
uint32_t format
uint32_t type

struct lv_gltf_ibl_image_t
```

Membri Pubblici

```
float *data
size_t data_len
uint32_t width
uint32_t height
```

math**lv_gltf_math.hpp**

GLTF math utilities and helper functions.

Define**M_PI****Funzioni**

```
fastgltf::math::fmat4x4 lv_gltf_math_look_at_rh(const fastgltf::math::fvec3 &eye, const fastgltf::math::fvec3
&center, const fastgltf::math::fvec3 &up) noexcept
fastgltf::math::fmat4x4 lv_gltf_math_perspective_rh(float fov, float ratio, float z_near, float z_far) noexcept
template<typename T>
fastgltf::math::quat<T> lv_gltf_math_euler_to_quaternion(T P, T Y, T R)
template<typename T>
fastgltf::math::vec<T, 3> lv_gltf_math_quaternion_to_euler(fastgltf::math::quat<T> q)
```

14.20.9 gstreamer

lv_gstreamer.h

Define

LV_GSTREAMER_FACTORY_URI_DECODE

LV_GSTREAMER_PROPERTY_URI_DECODE

LV_GSTREAMER_FACTORY_FILE

LV_GSTREAMER_PROPERTY_FILE

Enum

enum **lv_gstreamer_state_t**

These sources are untested. For most of them, URL_DECODE can probably be used instead

Values:

enumerator **LV_GSTREAMER_STATE_NULL**

enumerator **LV_GSTREAMER_STATE_READY**

enumerator **LV_GSTREAMER_STATE_PAUSED**

enumerator **LV_GSTREAMER_STATE_PLAYING**

Funzioni

lv_obj_t *lv_gstreamer_create(lv_obj_t *parent)

Create a gstreamer object

Parametri

parent -- pointer to an object, it will be the parent of the new gstreamer

Ritorna

pointer to the created gstreamer

lv_result_t lv_gstreamer_set_src(lv_obj_t *gstreamer, const char *factory_name, const char *property, const char *source)

Add a source to this gstreamer object

Parametri

- **gstreamer** -- pointer to a gstreamer object
- **factory_name** -- the factory name for the source of this gstreamer object. for common factory names, check **LV_GSTREAMER_FACTORY_XXX** defines
- **property** -- the property name for the gstreamer source object for common properties, see **LV_GSTREAMER_PROPERTY_XXX** defines Passing NULL will create the source object but not set its source
- **source** -- the property value for the gstreamer source object Passing NULL will create the source object but not set its source

Ritorna

LV_RESULT_OK if the source was correctly set else LV_RESULT_INVALID

`void lv_gstreamer_play(lv_obj_t *gstreamer)`

Play this gstreamer

Parametri

`gstreamer` -- pointer to a gstreamer object

`void lv_gstreamer_pause(lv_obj_t *gstreamer)`

Pause this gstreamer

Parametri

`gstreamer` -- pointer to a gstreamer object

`void lv_gstreamer_stop(lv_obj_t *gstreamer)`

Stop this gstreamer

Parametri

`gstreamer` -- pointer to a gstreamer object

`void lv_gstreamer_set_position(lv_obj_t *gstreamer, uint32_t position)`

Seek a position in this gstreamer

Parametri

- `gstreamer` -- pointer to a gstreamer object
- `position` -- position to seek to

`uint32_t lv_gstreamer_get_duration(lv_obj_t *gstreamer)`

Get the duration of this gstreamer

Parametri

`gstreamer` -- pointer to a gstreamer object

Ritorna

the duration (in ms) of the gstreamer object

`uint32_t lv_gstreamer_get_position(lv_obj_t *gstreamer)`

Get the position of this gstreamer

Parametri

`gstreamer` -- pointer to a gstreamer object

Ritorna

the position (in ms) of the gstreamer object

`lv_gstreamer_state_t lv_gstreamer_get_state(lv_obj_t *gstreamer)`

Get the state of this gstreamer

Parametri

`gstreamer` -- pointer to a gstreamer object

`void lv_gstreamer_set_volume(lv_obj_t *gstreamer, uint8_t volume)`

Set the volume of this gstreamer

Parametri

- `gstreamer` -- pointer to a gstreamer object
- `volume` -- the value to set in the range [0..100]. Higher values are clamped

`uint8_t lv_gstreamer_get_volume(lv_obj_t *gstreamer)`

Get the volume of this gstreamer

Parametri

`gstreamer` -- pointer to a gstreamer object

Ritorna

the volume for this gstreamer

`void lv_gstreamer_set_rate(lv_obj_t *gstreamer, uint32_t rate)`

Set the speed rate of this gstreamer

Parametri

- `gstreamer` -- pointer to a gstreamer object
- `rate` -- the rate factor. Example values:

– 256: 1x

- <256: slow down
- >256: speed up
- 128: 0.5x
- 512: 2x

Variabili

const lv_obj_class_t **lv_gstreamer_class**

lv_gstreamer_internal.h

TypeDef

typedef struct *lv_gstreamer_t* **lv_gstreamer_t**

struct **lv_gstreamer_t**

Membri Pubblici

lv_image_t **image**

lv_image_dsc_t **frame**

GstVideoInfo **video_info**

GstElement ***pipeline**

GstElement ***audio_convert**

GstElement ***video_convert**

GstElement ***audio_volume**

GstSample ***last_sample**

lv_timer_t ***gstreamer_timer**

GAsyncQueue ***frame_queue**

bool **is_video_info_valid**

struct **frame_data_t**

Membri Pubblici

uint8_t ***frame_data**

```
    uint32_t width
```

```
    uint32_t height
```

```
    uint32_t stride
```

```
    size_t data_size
```

14.20.10 libjpeg_turbo

lv_libjpeg_turbo.h

Funzioni

```
void lv_libjpeg_turbo_init(void)
```

Register the JPEG-Turbo decoder functions in LVGL

```
void lv_libjpeg_turbo_deinit(void)
```

14.20.11 libpng

lv_libpng.h

Funzioni

```
void lv_libpng_init(void)
```

Register the PNG decoder functions in LVGL

```
void lv_libpng_deinit(void)
```

14.20.12 libwebp

lv_libwebp.h

Funzioni

```
void lv_libwebp_init(void)
```

Register the WEBP decoder functions in LVGL

```
void lv_libwebp_deinit(void)
```

14.20.13 lodepng

lv_lodepng.h

Funzioni

```
void lv_lodepng_init(void)
```

Register the PNG decoder functions in LVGL

```
void lv_lodepng_deinit(void)
```

14.20.14 qrcode

lv_qrcode.h

Funzioni

`lv_obj_t *lv_qrcode_create(lv_obj_t *parent)`

Create an empty QR code (an `lv_canvas`) object.

Parametri

- `parent` -- point to an object where to create the QR code

Ritorna

pointer to the created QR code object

`void lv_qrcode_set_size(lv_obj_t *obj, int32_t size)`

Set QR code size.

Parametri

- `obj` -- pointer to a QR code object
- `size` -- width and height of the QR code

`void lv_qrcode_set_dark_color(lv_obj_t *obj, lv_color_t color)`

Set QR code dark color.

Parametri

- `obj` -- pointer to a QR code object
- `color` -- dark color of the QR code

`void lv_qrcode_set_light_color(lv_obj_t *obj, lv_color_t color)`

Set QR code light color.

Parametri

- `obj` -- pointer to a QR code object
- `color` -- light color of the QR code

`lv_result_t lv_qrcode_update(lv_obj_t *obj, const void *data, uint32_t data_len)`

Set the data of a QR code object

Parametri

- `obj` -- pointer to a QR code object
- `data` -- data to display
- `data_len` -- length of data in bytes

Ritorna

`LV_RESULT_OK`: if no error; `LV_RESULT_INVALID`: on error

`void lv_qrcode_set_data(lv_obj_t *obj, const char *data)`

Helper function to set the data of a QR code object

Parametri

- `obj` -- pointer to a QR code object
- `data` -- data to display as a string

`void lv_qrcode_set_quiet_zone(lv_obj_t *obj, bool enable)`

Enable or disable quiet zone. Quiet zone is the area around the QR code where no data is encoded.

Parametri

- `obj` -- pointer to a QR code object
- `enable` -- true: enable quiet zone; false: disable quiet zone

Variabili

`const lv_obj_class_t lv_qrcode_class`

lv_qrcode_private.h

`struct _lv_qrcode_t`

Membri Pubblici

`lv_canvas_t canvas`

`lv_color_t dark_color`

`lv_color_t light_color`

`int32_t quiet_zone`

14.20.15 rle

`lv_rle.h`

Funzioni

```
uint32_t lv_rle_decompress(const uint8_t *input, uint32_t input_buff_len, uint8_t *output, uint32_t
                           output_buff_len, uint8_t blk_size)
```

14.20.16 rlottie

`lv_rlottie.h`

Enum

enum `lv_rlottie_ctrl_t`

Values:

enumerator `LV_RLOTTIE_CTRL_FORWARD`

enumerator `LV_RLOTTIE_CTRL_BACKWARD`

enumerator `LV_RLOTTIE_CTRL_PAUSE`

enumerator `LV_RLOTTIE_CTRL_PLAY`

enumerator `LV_RLOTTIE_CTRL_LOOP`

Funzioni

`lv_obj_t *lv_rlottie_create_from_file(lv_obj_t *parent, int32_t width, int32_t height, const char *path)`

`lv_obj_t *lv_rlottie_create_from_raw(lv_obj_t *parent, int32_t width, int32_t height, const char *rlottie_desc)`

`void lv_rlottie_set_play_mode(lv_obj_t *rlottie, const lv_rlottie_ctrl_t ctrl)`

`void lv_rlottie_set_current_frame(lv_obj_t *rlottie, const size_t goto_frame)`

Variabili

const `lv_obj_class_t lv_rlottie_class`

lv_rlottie_private.h

```
struct _lv_rlottie_t
```

Membri Pubblici

```
lv_image_t img_ext
```

```
struct Lottie_Animation_S *animation
```

```
lv_timer_t *task
```

```
lv_image_dsc_t imgdsc
```

```
size_t total_frames
```

```
size_t current_frame
```

```
size_t framerate
```

```
uint32_t *allocated_buf
```

```
size_t allocated_buffer_size
```

```
size_t scanline_width
```

```
lv_rlottie_ctrl_t play_ctrl
```

```
size_t dest_frame
```

14.20.17 svg**lv_svg.h****Define**

```
LV_SVG_NODE_CHILD(n, i)
```

```
LV_SVG_NODE(n)
```

Typedef

```
typedef int8_t lv_svg_tag_t
```

```
typedef uint8_t lv_svg_attr_type_t
```

```
typedef uint8_t lv_svg_transform_type_t
```

```
typedef uint32_t lv_svg_aspect_ratio_t
```

```
typedef uint32_t lv_svg_color_t  
  
typedef uint8_t lv_svg_fill_rule_t  
  
typedef uint8_t lv_svg_line_cap_t  
  
typedef uint8_t lv_svg_line_join_t  
  
typedef uint8_t lv_svg_gradient_units_t  
  
typedef uint8_t lv_svg_attr_value_type_t  
  
typedef uint8_t lv_svg_attr_value_class_t
```

Enum

enum `lv_svg_tag_t`

Values:

enumerator `LV_SVG_TAG_INVALID`

enumerator `LV_SVG_TAG_CONTENT`

enumerator `LV_SVG_TAG_SVG`

enumerator `LV_SVG_TAG_USE`

enumerator `LV_SVG_TAG_G`

enumerator `LV_SVG_TAG_PATH`

enumerator `LV_SVG_TAG_RECT`

enumerator `LV_SVG_TAG_CIRCLE`

enumerator `LV_SVG_TAG_ELLIPSE`

enumerator `LV_SVG_TAG_LINE`

enumerator `LV_SVG_TAG_POLYLINE`

enumerator `LV_SVG_TAG_POLYGON`

enumerator `LV_SVG_TAG_SOLID_COLOR`

enumerator **LV_SVG_TAG_LINEAR_GRADIENT**

enumerator **LV_SVG_TAG_RADIAL_GRADIENT**

enumerator **LV_SVG_TAG_STOP**

enumerator **LV_SVG_TAG_DEFS**

enumerator **LV_SVG_TAG_IMAGE**

enumerator **LV_SVG_TAG_MPATH**

enumerator **LV_SVG_TAG_SET**

enumerator **LV_SVG_TAG_ANIMATE**

enumerator **LV_SVG_TAG_ANIMATE_COLOR**

enumerator **LV_SVG_TAG_ANIMATE_TRANSFORM**

enumerator **LV_SVG_TAG_ANIMATE_MOTION**

enumerator **LV_SVG_TAG_TEXT**

enumerator **LV_SVG_TAG_TSPAN**

enumerator **LV_SVG_TAG_TEXT_AREA**

enum **lv_svg_attr_type_t**

Values:

enumerator **LV_SVG_ATTR_INVALID**

enumerator **LV_SVG_ATTR_ID**

enumerator **LV_SVG_ATTR_XML_ID**

enumerator **LV_SVG_ATTR_VERSION**

enumerator **LV_SVG_ATTR_BASE_PROFILE**

enumerator **LV_SVG_ATTR_VIEWBOX**

enumerator **LV_SVG_ATTR_PRESERVE_ASPECT_RATIO**

enumerator **LV_SVG_ATTR_VIEWPORT_FILL**

enumerator **LV_SVG_ATTR_VIEWPORT_FILL_OPACITY**

enumerator **LV_SVG_ATTR_DISPLAY**

enumerator **LV_SVG_ATTR_VISIBILITY**

enumerator **LV_SVG_ATTR_X**

enumerator **LV_SVG_ATTR_Y**

enumerator **LV_SVG_ATTR_WIDTH**

enumerator **LV_SVG_ATTR_HEIGHT**

enumerator **LV_SVG_ATTR_RX**

enumerator **LV_SVG_ATTR_RY**

enumerator **LV_SVG_ATTR_CX**

enumerator **LV_SVG_ATTR_CY**

enumerator **LV_SVG_ATTR_R**

enumerator **LV_SVG_ATTR_X1**

enumerator **LV_SVG_ATTR_Y1**

enumerator **LV_SVG_ATTR_X2**

enumerator **LV_SVG_ATTR_Y2**

enumerator **LV_SVG_ATTR_POINTS**

enumerator **LV_SVG_ATTR_D**

enumerator **LV_SVG_ATTR_PATH_LENGTH**

enumerator **LV_SVG_ATTR_XLINK_HREF**

enumerator **LV_SVG_ATTR_FILL**

enumerator **LV_SVG_ATTR_FILL_RULE**

enumerator **LV_SVG_ATTR_FILL_OPACITY**

enumerator **LV_SVG_ATTR_STROKE**

enumerator **LV_SVG_ATTR_STROKE_WIDTH**

enumerator **LV_SVG_ATTR_STROKE_LINECAP**

enumerator **LV_SVG_ATTR_STROKE_LINEJOIN**

enumerator **LV_SVG_ATTR_STROKE_MITER_LIMIT**

enumerator **LV_SVG_ATTR_STROKE_DASH_ARRAY**

enumerator **LV_SVG_ATTR_STROKE_DASH_OFFSET**

enumerator **LV_SVG_ATTR_STROKE_OPACITY**

enumerator **LV_SVG_ATTR_SOLID_COLOR**

enumerator **LV_SVG_ATTR_SOLID_OPACITY**

enumerator **LV_SVG_ATTR_GRADIENT_UNITS**

enumerator **LV_SVG_ATTR_GRADIENT_STOP_OFFSET**

enumerator **LV_SVG_ATTR_GRADIENT_STOP_COLOR**

enumerator **LV_SVG_ATTR_GRADIENT_STOP_OPACITY**

enumerator **LV_SVG_ATTR_FONT_FAMILY**

enumerator **LV_SVG_ATTR_FONT_STYLE**

enumerator **LV_SVG_ATTR_FONT_VARIANT**

enumerator **LV_SVG_ATTR_FONT_WEIGHT**

enumerator **LV_SVG_ATTR_FONT_SIZE**

enumerator **LV_SVG_ATTR_TRANSFORM**

enumerator **LV_SVG_ATTR_TEXT_ANCHOR**

enumerator **LV_SVG_ATTR_ATTRIBUTE_NAME**

enumerator **LV_SVG_ATTR_ATTRIBUTE_TYPE**

enumerator **LV_SVG_ATTR_BEGIN**

enumerator **LV_SVG_ATTR_END**

enumerator **LV_SVG_ATTR_DUR**

enumerator **LV_SVG_ATTR_MIN**

enumerator **LV_SVG_ATTR_MAX**

enumerator **LV_SVG_ATTR_RESTART**

enumerator **LV_SVG_ATTR_REPEAT_COUNT**

enumerator **LV_SVG_ATTR_REPEAT_DUR**

enumerator **LV_SVG_ATTR_CALC_MODE**

enumerator **LV_SVG_ATTR_VALUES**

enumerator **LV_SVG_ATTR_KEY_TIMES**

enumerator **LV_SVG_ATTR_KEY_SPLINES**

enumerator **LV_SVG_ATTR_KEY_POINTS**

enumerator **LV_SVG_ATTR_FROM**

enumerator **LV_SVG_ATTR_TO**

enumerator **LV_SVG_ATTR_BY**

enumerator **LV_SVG_ATTR_ADDITIVE**

enumerator **LV_SVG_ATTR_ACCUMULATE**

enumerator **LV_SVG_ATTR_PATH**

enumerator **LV_SVG_ATTR_ROTATE**

enumerator **LV_SVG_ATTR_TRANSFORM_TYPE**

enum **_lv_svg_transform_type_t**

Values:

enumerator **LV_SVG_TRANSFORM_TYPE_MATRIX**

enumerator **LV_SVG_TRANSFORM_TYPE_TRANSLATE**

enumerator **LV_SVG_TRANSFORM_TYPE_ROTATE**

enumerator **LV_SVG_TRANSFORM_TYPE_SCALE**

enumerator **LV_SVG_TRANSFORM_TYPE_SKW_X**

enumerator **LV_SVG_TRANSFORM_TYPE_SKW_Y**

enum **lv_svg_anim_action_t**

Values:

enumerator **LV_SVG_ANIM_REMOVE**

enumerator **LV_SVG_ANIM_FREEZE**

enum **_lv_svg_anim_restart_type_t**

Values:

enumerator **LV_SVG_ANIM_RESTART_ALWAYS**

enumerator **LV_SVG_ANIM_RESTART_WHEN_NOT_ACTIVE**

enumerator **LV_SVG_ANIM_RESTART_NEVER**

enum **_lv_svg_anim_calc_mode_t**

Values:

enumerator **LV_SVG_ANIM_CALC_MODE_LINEAR**

enumerator **LV_SVG_ANIM_CALC_MODE_PACED**

enumerator **LV_SVG_ANIM_CALC_MODE_SPLINE**

enumerator **LV_SVG_ANIM_CALC_MODE_DISCRETE**

enum **_lv_svg_anim_additive_type_t**

Values:

enumerator **LV_SVG_ANIM_ADDITIVE_REPLACE**

enumerator **LV_SVG_ANIM_ADDITIVE_SUM**

enum **_lv_svg_anim_accumulate_type_t**

Values:

enumerator **LV_SVG_ANIM_ACCUMULATE_NONE**

enumerator **LV_SVG_ANIM_ACCUMULATE_SUM**

enum **_lv_svg_aspect_ratio_t**

Values:

enumerator **LV_SVG_ASPECT_RATIO_NONE**

enumerator **LV_SVG_ASPECT_RATIO_XMIN_YMIN**

enumerator **LV_SVG_ASPECT_RATIO_XMID_YMIN**

enumerator **LV_SVG_ASPECT_RATIO_XMAX_YMIN**

enumerator **LV_SVG_ASPECT_RATIO_XMIN_YMID**

enumerator **LV_SVG_ASPECT_RATIO_XMID_YMID**

enumerator **LV_SVG_ASPECT_RATIO_XMAX_YMID**

enumerator **LV_SVG_ASPECT_RATIO_XMIN_YMAX**

enumerator **LV_SVG_ASPECT_RATIO_XMID_YMAX**

enumerator **LV_SVG_ASPECT_RATIO_XMAX_YMAX**

enum **_lv_svg_aspect_ratio_opt_t**

Values:

enumerator **LV_SVG_ASPECT_RATIO_OPT_MEET**

enumerator **LV_SVG_ASPECT_RATIO_OPT_SLICE**

enum **_lv_svg_fill_rule_t**

Values:

enumerator **LV_SVG_FILL_NONZERO**

enumerator **LV_SVG_FILL_EVENODD**

enum **_lv_svg_line_cap_t**

Values:

enumerator **LV_SVG_LINE_CAP_BUTT**

enumerator **LV_SVG_LINE_CAP_SQUARE**

enumerator **LV_SVG_LINE_CAP_ROUND**

enum **_lv_svg_line_join_t**

Values:

enumerator **LV_SVG_LINE_JOIN_MITER**

enumerator **LV_SVG_LINE_JOIN_BEVEL**

enumerator **LV_SVG_LINE_JOIN_ROUND**

enum **_lv_svg_gradient_units_t**

Values:

enumerator **LV_SVG_GRADIENT_UNITS_OBJECT**

enumerator **LV_SVG_GRADIENT_UNITS_USER_SPACE**

enum **_lv_svg_path_cmd_t**

Values:

enumerator **LV_SVG_PATH_CMD_MOVE_TO**

enumerator **LV_SVG_PATH_CMD_LINE_TO**

enumerator **LV_SVG_PATH_CMD_CURVE_TO**

enumerator **LV_SVG_PATH_CMD_QUAD_TO**

enumerator **LV_SVG_PATH_CMD_ARC_TO**

enumerator **LV_SVG_PATH_CMD_CLOSE**

enum **_lv_svg_attr_value_type_t**

Values:

enumerator **LV_SVG_ATTR_VALUE_DATA**

enumerator **LV_SVG_ATTR_VALUE_PTR**

```
enum _lv_svg_attr_value_class_t
```

Values:

enumerator **LV_SVG_ATTR_VALUE_NONE**

enumerator **LV_SVG_ATTR_VALUE_INITIAL**

enumerator **LV_SVG_ATTR_VALUE_INHERIT**

Funzioni

*lv_svg_node_t *lv_svg_load_data(const char *svg_data, uint32_t data_len)*

Loading SVG data and creating the DOM tree.

Parametri

- **svg_data** -- pointer to the SVG data
- **data_len** -- the SVG data length

*lv_svg_node_t *lv_svg_node_create(lv_svg_node_t *parent)*

Create an SVG DOM node.

Parametri

- **parent** -- pointer to the parent node

Ritorna

true: an new SVG DOM node, false: NULL

*void lv_svg_node_delete(lv_svg_node_t *node)*

Delete an SVG DOM subtree.

Parametri

- **node** -- pointer to an SVG DOM subtree

struct lv_svg_point_t

Membri Pubblici

float x

float y

struct lv_svg_matrix_t

Membri Pubblici

float m[3][3]

union lv_svg_attr_value_t

Membri Pubblici

int32_t ival

uint32_t uval

```
float fval

char *sval

void *val

struct lv_svg_attr_values_list_t
```

Membri Pubblici

```
uint32_t length
```

```
uint8_t data[1]
```

```
struct lv_svg_attr_path_value_t
```

Membri Pubblici

```
uint32_t cmd
```

```
uint8_t data[1]
```

```
struct lv_svg_attr_t
```

Membri Pubblici

```
lv_svg_attr_type_t id
```

```
lv_svg_attr_value_type_t val_type
```

```
lv_svg_attr_value_class_t class_type
```

```
lv_svg_attr_value_t value
```

```
struct lv_svg_node_t
```

Membri Pubblici

```
lv_tree_node_t base
```

```
char *xml_id
```

```
lv_svg_tag_t type
```

```
lv_array_t attrs
```

```
struct lv_svg_render_obj *render_obj
```

lv_svg_decoder.h**Funzioni****void lv_svg_decoder_init(void)**

Register the SVG decoder functions in LVGL

void lv_svg_decoder_deinit(void)**lv_svg_parser.h****Enum****enum _lv_svg_parser_state_t***Values:* enumerator **LV_SVG_PARSER_PROCESS** enumerator **LV_SVG_PARSER_IGNORE****Funzioni****void _lv_svg_parser_init(_lv_svg_parser_t *parser)**

Initialize the SVG parser.

Parametri **parser** -- pointer to a parser object**void _lv_svg_parser_deinit(_lv_svg_parser_t *parser)**

Deinitialize the SVG parser.

Parametri **parser** -- pointer to a parser object**bool _lv_svg_parser_token(_lv_svg_parser_t *parser, const _lv_svg_token_t *token)**

Parse an SVG document.

Parametri

- **parser** -- pointer to a parser object
- **token** -- pointer to a token object

Ritorna

true: the parsing is finished, false: the parsing is not finished yet.

bool _lv_svg_parser_is_finish(_lv_svg_parser_t *parser)

Check if the parsing is finished.

Parametri **parser** -- pointer to a parser object**Ritorna**

true: the parsing is finished, false: the parsing is not finished yet.

void _lv_svg_dump_tree(lv_svg_node_t *root, int depth)

Dump the SVG tree.

Parametri

- **root** -- pointer to the root of the SVG tree
- **depth** -- the depth of the current node in the tree

struct _lv_svg_parser_t

Membri Pubblici

```
uint16_t state
char *ignore_name
uint32_t ignore_len
int32_t dpi
lv_svg_node_t *doc_root
lv_svg_node_t *cur_node
```

lv_svg_render.h**Define**

```
LV_SVG_RENDER_OBJ(n)
```

Typedef

```
typedef struct _lv_svg_render_obj lv_svg_render_obj_t
typedef struct _lv_svg_render_class lv_svg_render_class
typedef struct _lv_svg_render_hal lv_svg_render_hal_t
```

Funzioni

void lv_svg_render_init(const lv_svg_render_hal_t *hal)

Initialize the SVG render.

Parametri

hal -- pointer to a structure with rendering functions

lv_svg_render_obj_t *lv_svg_render_create(const lv_svg_node_t *svg_doc)

Create a new SVG render from an SVG document.

Parametri

svg_doc -- pointer to the SVG document

Ritorna

pointer to the new SVG render object

void lv_svg_render_delete(lv_svg_render_obj_t *render)

Delete an SVG render object.

Parametri

render -- pointer to the SVG render object to delete

uint32_t lv_svg_render_get_size(const lv_svg_render_obj_t *render)

Get size of render objects.

Parametri

render -- pointer to the SVG render object

Ritorna

the bytes of SVG render objects

`lv_result_t lv_svg_render_get_viewport_size(const lv_svg_render_obj_t *render, float *width, float *height)`

Get viewport's width and height of the render object.

Parametri

- **render** -- pointer to the SVG render object
- **width** -- pointer to save the width of the viewport of the SVG render object
- **height** -- pointer to save the height of the viewport of the SVG render object

Ritorna

`lv_result_t, LV_RESULT_OK if success, LV_RESULT_INVALID if fail`

`void lv_draw_svg_render(lv_draw_vector_dsc_t *dsc, const lv_svg_render_obj_t *render)`

Render an SVG object to a vector graphics.

Parametri

- **dsc** -- pointer to the vector graphics descriptor
- **render** -- pointer to the SVG render object to render

`void lv_draw_svg(lv_layer_t *layer, const lv_svg_node_t *svg_doc)`

Draw an SVG document to a layer.

Parametri

- **layer** -- pointer to the target layer
- **svg_doc** -- pointer to the SVG document to draw

`struct _lv_svg_render_obj`

Membri Pubblici

`struct _lv_svg_render_obj *next`

`lv_svg_tag_t tag`

`uint32_t flags`

`char *id`

`lv_vector_path_ctx_t dsc`

`lv_matrix_t matrix`

`struct _lv_svg_render_obj *head`

`char *fill_ref`

`char *stroke_ref`

`struct _lv_svg_render_class *cls`

`struct _lv_svg_render_class`

Membri Pubblici

`void (*set_paint_ref)(struct _lv_svg_render_obj *obj, lv_vector_path_ctx_t *dsc, const struct _lv_svg_render_obj *target_obj, bool fill)`

```

void (*init)(struct lv_svg_render_obj *obj, const lv_svg_node_t *node)

void (*render)(const struct lv_svg_render_obj *obj, lv_draw_vector_dsc_t *dsc, const lv_matrix_t *matrix)

void (*set_attr)(struct lv_svg_render_obj *obj, lv_vector_path_ctx_t *dsc, const lv_svg_attr_t *attr)

void (*get_bounds)(const struct lv_svg_render_obj *obj, lv_area_t *area)

void (*get_size)(const struct lv_svg_render_obj *obj, uint32_t *size)

void (*destroy)(struct lv_svg_render_obj *obj)

```

struct ***lv_svg_render_hal***

Membri Pubblici

```

void (*load_image)(const char *image_url, lv_draw_image_dsc_t *img_dsc)

const char *(*get_font_path)(const char *font_family)

```

lv_svg_token.h

Define

SVG_TOKEN_LEN(t)

Typedef

typedef bool (***svg_token_process**)(*lv_svg_token_t* *token, void *user_data)

Enum

enum ***lv_svg_token_type_t***

Values:

enumerator **LV_SVG_TOKEN_BEGIN**

enumerator **LV_SVG_TOKEN_END**

enumerator **LV_SVG_TOKEN_CONTENT**

Funzioni

bool ***lv_svg_tokenizer***(const char *svg_data, uint32_t len, *svg_token_process* cb, void *user_data)

Parse SVG data and call a callback for each token.

Parametri

- **svg_data** -- pointer to SVG data
- **len** -- length of the SVG data
- **cb** -- callback function to be called for each token
- **user_data** -- custom data to be passed to the callback function

Ritorna

true: SVG data successfully parsed, false: error occurred

```
struct _lv_svg_token_attr_t
```

Membri Pubblici

```
const char *name_start
```

```
const char *name_end
```

```
const char *value_start
```

```
const char *value_end
```

```
struct _lv_svg_token_t
```

Membri Pubblici

```
const char *start
```

```
const char *end
```

```
_lv_svg_token_type_t type
```

```
bool flat
```

```
_lv_svg_token_attr_t *cur_attr
```

```
lv_array_t attrs
```

14.20.18 tiny_ttf

lv_tiny_ttf.h

Funzioni

`lv_font_t *lv_tiny_ttf_create_data(const void *data, size_t data_size, int32_t font_size)`

Create a font from the specified data pointer with the specified line height.

Parametri

- **data** -- the data pointer
- **data_size** -- the data size
- **font_size** -- the font size in pixel

Ritorna

a font object

`lv_font_t *lv_tiny_ttf_create_data_ex(const void *data, size_t data_size, int32_t font_size, lv_font_kerning_t kerning, size_t cache_size)`

Create a font from the specified data pointer with the specified line height and the specified cache size.

Parametri

- **data** -- the data pointer
- **data_size** -- the data size

- **font_size** -- the font size in pixel
- **kerning** -- kerning value in pixel
- **cache_size** -- the cache size in count

Ritorna

void **lv_tiny_ttf_set_size**(lv_font_t *font, int32_t font_size)

Set the size of the font to a new font_size

 **Nota**

the font bitmap cache and glyph cache will be flushed.

Parametri

- **font** -- the font object
- **font_size** -- the font size in pixel

void **lv_tiny_ttf_destroy**(lv_font_t *font)

Destroy a font previously created with lv_tiny_ttf_create_xxxx()

Parametri

font -- the font object

Variabili

const lv_font_class_t **lv_tiny_ttf_font_class**

struct **lv_tiny_ttf_font_src_t**

Membri Pubblici

const char ***path**

Path to the font file

const void ***data**

Pointer to the font data

size_t **data_size**

Size of the font data

size_t **cache_size**

Size of the font cache

14.20.19 tjpgd

lv_tjpd.h

14.21 misc

14.21.1 lv_anim.h

Define

LV_ANIM_REPEAT_INFINITE

LV_ANIM_PLAYTIME_INFINITE

LV_ANIM_PAUSE_FOREVER

_PARA(a, x1, y1, x2, y2)

LV_ANIM_SET_EASE_IN_SINE(a)

LV_ANIM_SET_EASE_OUT_SINE(a)

LV_ANIM_SET_EASE_IN_OUT_SINE(a)

LV_ANIM_SET_EASE_IN_QUAD(a)

LV_ANIM_SET_EASE_OUT_QUAD(a)

LV_ANIM_SET_EASE_IN_OUT_QUAD(a)

LV_ANIM_SET_EASE_IN_CUBIC(a)

LV_ANIM_SET_EASE_OUT_CUBIC(a)

LV_ANIM_SET_EASE_IN_OUT_CUBIC(a)

LV_ANIM_SET_EASE_IN_QUART(a)

LV_ANIM_SET_EASE_OUT_QUART(a)

LV_ANIM_SET_EASE_IN_OUT_QUART(a)

LV_ANIM_SET_EASE_IN_QUINT(a)

LV_ANIM_SET_EASE_OUT_QUINT(a)

LV_ANIM_SET_EASE_IN_OUT_QUINT(a)

LV_ANIM_SET_EASE_IN_EXPO(a)

LV_ANIM_SET_EASE_OUT_EXPO(a)

LV_ANIM_SET_EASE_IN_OUT_EXPO(a)

LV_ANIM_SET_EASE_IN_CIRC(a)

LV_ANIM_SET_EASE_OUT_CIRC(a)

LV_ANIM_SET_EASE_IN_OUT_CIRC(a)

LV_ANIM_SET_EASE_IN_BACK(a)

LV_ANIM_SET_EASE_OUT_BACK(a)

LV_ANIM_SET_EASE_IN_OUT_BACK(a)

LV_ANIM_OFF

Can be used to indicate if animations are enabled or disabled in a case

LV_ANIM_ON

Typedef

`typedef bool lv_anim_enable_t`

`typedef int32_t (*lv_anim_path_cb_t)(const lv_anim_t*)`

Get the current value during an animation

`typedef void (*lv_anim_exec_xcb_t)(void*, int32_t)`

Generic prototype of "animator" functions. First parameter is the variable to animate. Second parameter is the value to set. Compatible with `lv_xxx_set_yyy(obj, value)` functions. The x in `_xcb_t` means it's not a fully generic prototype because it doesn't receive `lv_anim_t *` as its first argument

`typedef void (*lv_anim_custom_exec_cb_t)(lv_anim_t*, int32_t)`

Same as `lv_anim_exec_xcb_t` but receives `lv_anim_t *` as the first parameter. It's more consistent but less convenient. Might be used by binding generator functions.

`typedef void (*lv_anim_completed_cb_t)(lv_anim_t*)`

Callback to call when the animation is ready

`typedef void (*lv_anim_start_cb_t)(lv_anim_t*)`

Callback to call when the animation really stars (considering `delay`)

`typedef int32_t (*lv_anim_get_value_cb_t)(lv_anim_t*)`

Callback used when the animation values are relative to get the current value

`typedef void (*lv_anim_deleted_cb_t)(lv_anim_t*)`

Callback used when the animation is deleted

Funzioni

`void lv_anim_init(lv_anim_t *a)`

Initialize an animation variable. E.g.: `lv_anim_t a; lv_anim_init(&a); lv_anim_set_...(a); lv_anim_start(a);`
Parametri

`a` -- pointer to an `lv_anim_t` variable to initialize

`void lv_anim_set_var(lv_anim_t *a, void *var)`

Set a variable to animate

Parametri

- `a` -- pointer to an initialized `lv_anim_t` variable
- `var` -- pointer to a variable to animate

`void lv_anim_set_exec_cb(lv_anim_t *a, lv_anim_exec_xcb_t exec_cb)`

Set a function to animate var

Parametri

- `a` -- pointer to an initialized `lv_anim_t` variable
- `exec_cb` -- a function to execute during animation. LVGL's built-in functions can be used. E.g. `lv_obj_set_x`

`void lv_anim_set_duration(lv_anim_t *a, uint32_t duration)`

Set the duration of an animation

Parametri

- `a` -- pointer to an initialized `lv_anim_t` variable
- `duration` -- duration of the animation in milliseconds

void lv_anim_set_delay(lv_anim_t *a, uint32_t delay)

Set a delay before starting the animation

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable
- **delay** -- delay before the animation in milliseconds

void lv_anim_resume(lv_anim_t *a)

Resumes a paused animation

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable

void lv_anim_pause(lv_anim_t *a)

Pauses the animation

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable

void lv_anim_pause_for(lv_anim_t *a, uint32_t ms)

Pauses the animation for ms milliseconds

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable
- **ms** -- the pause time in milliseconds

bool lv_anim_is_paused(lv_anim_t *a)

Check if the animation is paused

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable

Ritorna

true if the animation is paused else false

void lv_anim_set_values(lv_anim_t *a, int32_t start, int32_t end)

Set the start and end values of an animation

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable
- **start** -- the start value
- **end** -- the end value

void lv_anim_set_custom_exec_cb(lv_anim_t *a, lv_anim_custom_exec_cb_t exec_cb)

Similar to `lv_anim_set_exec_cb` but `lv_anim_custom_exec_cb_t` receives `lv_anim_t *` as its first parameter instead of `void *`. This function might be used when LVGL is bound to other languages because it's more consistent to have `lv_anim_t *` as first parameter.

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable
- **exec_cb** -- a function to execute.

void lv_anim_set_path_cb(lv_anim_t *a, lv_anim_path_cb_t path_cb)

Set the path (curve) of the animation.

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable
- **path_cb** -- a function to set the current value of the animation.

void lv_anim_set_start_cb(lv_anim_t *a, lv_anim_start_cb_t start_cb)

Set a function call when the animation really starts (considering `delay`)

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable
- **start_cb** -- a function call when the animation starts

void lv_anim_set_get_value_cb(lv_anim_t *a, lv_anim_get_value_cb_t get_value_cb)

Set a function to use the current value of the variable and make start and end value relative to the returned current value.

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable
- **get_value_cb** -- a function call when the animation starts

`void lv_anim_set_completed_cb(lv_anim_t *a, lv_anim_completed_cb_t completed_cb)`

Set a function call when the animation is completed

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable
- **completed_cb** -- a function call when the animation is fully completed

`void lv_anim_set_deleted_cb(lv_anim_t *a, lv_anim_deleted_cb_t deleted_cb)`

Set a function call when the animation is deleted.

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable
- **deleted_cb** -- a function call when the animation is deleted

`void lv_anim_set_reverse_duration(lv_anim_t *a, uint32_t duration)`

Make the animation to play back to when the forward direction is ready

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable
- **duration** -- duration of playback animation in milliseconds. 0: disable playback

`void lv_anim_set_reverse_time(lv_anim_t *a, uint32_t duration)`

Legacy `lv_anim_set_reverse_time` API will be removed soon, use `lv_anim_set_reverse_duration` instead.

`void lv_anim_set_reverse_delay(lv_anim_t *a, uint32_t delay)`

Make the animation to play back to when the forward direction is ready

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable
- **delay** -- delay in milliseconds before starting the playback animation.

`void lv_anim_set_repeat_count(lv_anim_t *a, uint32_t cnt)`

Make the animation repeat itself.

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable
- **cnt** -- repeat count or `LV_ANIM_REPEAT_INFINITE` for infinite repetition. 0: to disable repetition.

`void lv_anim_set_repeat_delay(lv_anim_t *a, uint32_t delay)`

Set a delay before repeating the animation.

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable
- **delay** -- delay in milliseconds before repeating the animation.

`void lv_anim_set_early_apply(lv_anim_t *a, bool en)`

Set a whether the animation's should be applied immediately or only when the delay expired.

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable
- **en** -- true: apply the start value immediately in `lv_anim_start`; false: apply the start value only when `delay` ms is elapsed and the animations really starts

`void lv_anim_set_user_data(lv_anim_t *a, void *user_data)`

Set the custom user data field of the animation.

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable
- **user_data** -- pointer to the new user_data.

`void lv_anim_set_bezier3_param(lv_anim_t *a, int16_t x1, int16_t y1, int16_t x2, int16_t y2)`

Set parameter for cubic bezier path

Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable

- **x1** -- first control point X
- **y1** -- first control point Y
- **x2** -- second control point X
- **y2** -- second control point Y

`lv_anim_t *lv_anim_start(const lv_anim_t *a)`

Create an animation

Parametri

`a` -- an initialized 'anim_t' variable. Not required after call.

Ritorna

pointer to the created animation (different from the `a` parameter)

`uint32_t lv_anim_get_delay(const lv_anim_t *a)`

Get a delay before starting the animation

Parametri

`a` -- pointer to an initialized `lv_anim_t` variable

Ritorna

delay before the animation in milliseconds

`uint32_t lv_anim_get_playtime(const lv_anim_t *a)`

Get the time used to play the animation.

Parametri

`a` -- pointer to an animation.

Ritorna

the play time in milliseconds.

`uint32_t lv_anim_get_time(const lv_anim_t *a)`

Get the duration of an animation

Parametri

`a` -- pointer to an initialized `lv_anim_t` variable

Ritorna

the duration of the animation in milliseconds

`uint32_t lv_anim_get_repeat_count(const lv_anim_t *a)`

Get the repeat count of the animation.

Parametri

`a` -- pointer to an initialized `lv_anim_t` variable

Ritorna

the repeat count or `LV_ANIM_REPEAT_INFINITE` for infinite repetition. 0: disabled repetition.

`void *lv_anim_get_user_data(const lv_anim_t *a)`

Get the `user_data` field of the animation

Parametri

`a` -- pointer to an initialized `lv_anim_t` variable

Ritorna

the pointer to the custom `user_data` of the animation

`bool lv_anim_delete(void *var, lv_anim_exec_xcb_t exec_cb)`

Delete animation(s) of a variable with a given animator function

Parametri

- **var** -- pointer to variable
- **exec_cb** -- a function pointer which is animating 'var', or NULL to ignore it and delete all the animations of 'var'

Ritorna

true: at least 1 animation is deleted, false: no animation is deleted

`void lv_anim_delete_all(void)`

Delete all the animations

`lv_anim_t *lv_anim_get(void *var, lv_anim_exec_xcb_t exec_cb)`

Get the animation of a variable and its exec_cb.

Parametri

- **var** -- pointer to variable
- **exec_cb** -- a function pointer which is animating 'var', or NULL to return first matching 'var'

Ritorna

pointer to the animation.

`lv_timer_t *lv_anim_get_timer(void)`

Get global animation refresher timer.

Ritorna

pointer to the animation refresher timer.

`bool lv_anim_custom_delete(lv_anim_t *a, lv_anim_custom_exec_cb_t exec_cb)`

Delete an animation by getting the animated variable from a. Only animations with exec_cb will be deleted. This function exists because it's logical that all anim. functions receives an `lv_anim_t` as their first parameter. It's not practical in C but might make the API more consequent and makes easier to generate bindings.

Parametri

- **a** -- pointer to an animation.
- **exec_cb** -- a function pointer which is animating 'var', or NULL to ignore it and delete all the animations of 'var'

Ritorna

true: at least 1 animation is deleted, false: no animation is deleted

`lv_anim_t *lv_anim_custom_get(lv_anim_t *a, lv_anim_custom_exec_cb_t exec_cb)`

Get the animation of a variable and its exec_cb. This function exists because it's logical that all anim. functions receives an `lv_anim_t` as their first parameter. It's not practical in C but might make the API more consequent and makes easier to generate bindings.

Parametri

- **a** -- pointer to an animation.
- **exec_cb** -- a function pointer which is animating 'var', or NULL to return first matching 'var'

Ritorna

pointer to the animation.

`uint16_t lv_anim_count_running(void)`

Get the number of currently running animations

Ritorna

the number of running animations

`uint32_t lv_anim_speed(uint32_t speed)`

Store the speed as a special value which can be used as time in animations. It will be converted to time internally based on the start and end values. The return value can be used as a constant with multiple animations and let LVGL convert the speed to time based on the actual values. LIMITATION: the max time stored this way can be 10,000 ms.

Nota

internally speed is stored as 10 unit/sec

Parametri

speed -- the speed of the animation in with unit / sec resolution in 0..10k range

Ritorna

a special value which can be used as an animation time

`uint32_t lv_anim_speed_clamped(uint32_t speed, uint32_t min_time, uint32_t max_time)`

Store the speed as a special value which can be used as time in animations. It will be converted to time internally

based on the start and end values. The return value can be used as a constant with multiple animations and let LVGL convert the speed to time based on the actual values.

Nota

internally speed is stored as 10 unit/sec

Nota

internally min/max_time are stored with 10 ms unit

Parametri

- **speed** -- the speed of the animation in as unit / sec resolution in 0..10k range
- **min_time** -- the minimum time in 0..10k range
- **max_time** -- the maximum time in 0..10k range

Ritorna

a special value in where all three values are stored and can be used as an animation time

```
uint32_t lv_anim_resolve_speed(uint32_t speed, int32_t start, int32_t end)
```

Resolve the speed (created with `lv_anim_speed` or `lv_anim_speed_clamped`) to time based on start and end values.

Parametri

- **speed** -- return values of `lv_anim_speed` or `lv_anim_speed_clamped`
- **start** -- the start value of the animation
- **end** -- the end value of the animation

Ritorna

the time required to get from `start` to `end` with the given speed setting

```
uint32_t lv_anim_speed_to_time(uint32_t speed, int32_t start, int32_t end)
```

Calculate the time of an animation based on its speed, start and end values. It simpler than `lv_anim_speed` or `lv_anim_speed_clamped` as it converts speed, start, and end to a time immediately. As it's simpler there is no limit on the maximum time.

Parametri

- **speed** -- the speed of the animation
- **start** -- the start value
- **end** -- the end value

Ritorna

the time of the animation in milliseconds

```
void lv_anim_refr_now(void)
```

Manually refresh the state of the animations. Useful to make the animations running in a blocking process where `lv_timer_handler` can't run for a while. Shouldn't be used directly because it is called in `lv_refr_now()`.

```
int32_t lv_anim_path_linear(const lv_anim_t *a)
```

Calculate the current value of an animation applying linear characteristic

Parametri

`a` -- pointer to an animation

Ritorna

the current value to set

```
int32_t lv_anim_path_ease_in(const lv_anim_t *a)
```

Calculate the current value of an animation slowing down the start phase

Parametri

`a` -- pointer to an animation

Ritorna

the current value to set

```
int32_t lv_anim_path_ease_out(const lv_anim_t *a)
```

Calculate the current value of an animation slowing down the end phase

Parametri

a -- pointer to an animation

Ritorna

the current value to set

```
int32_t lv_anim_path_ease_in_out(const lv_anim_t *a)
```

Calculate the current value of an animation applying an "S" characteristic (cosine)

Parametri

a -- pointer to an animation

Ritorna

the current value to set

```
int32_t lv_anim_path_overshoot(const lv_anim_t *a)
```

Calculate the current value of an animation with overshoot at the end

Parametri

a -- pointer to an animation

Ritorna

the current value to set

```
int32_t lv_anim_path_bounce(const lv_anim_t *a)
```

Calculate the current value of an animation with 3 bounces

Parametri

a -- pointer to an animation

Ritorna

the current value to set

```
int32_t lv_anim_path_step(const lv_anim_t *a)
```

Calculate the current value of an animation applying step characteristic. (Set end value on the end of the animation)

Parametri

a -- pointer to an animation

Ritorna

the current value to set

```
int32_t lv_anim_path_custom_bezier3(const lv_anim_t *a)
```

A custom cubic bezier animation path, need to specify cubic-parameters in a->parameter.bezier3

Parametri

a -- pointer to an animation

Ritorna

the current value to set

```
struct lv_anim_bezier3_para_t
```

#include <lv_anim.h> Parameter used when path is custom_bezier

Membri Pubblici

```
int16_t x1
```

```
int16_t y1
```

```
int16_t x2
```

```
int16_t y2
```

```
struct _lv_anim_t
#include <lv_anim.h> Describes an animation
```

Membri Pubblici

void *var

Variable (Widget or other user-provided object) to animate

lv_anim_exec_xcb_t exec_cb

Function to execute to animate

lv_anim_custom_exec_cb_t custom_exec_cb

Function to execute to animate, same purpose as exec_cb but different parameters

lv_anim_start_cb_t start_cb

Call it when animation is starts (considering delay)

lv_anim_completed_cb_t completed_cb

Call it when animation is fully completed

lv_anim_deleted_cb_t deleted_cb

Call it when animation is deleted

lv_anim_get_value_cb_t get_value_cb

Get current value in relative mode

void *user_data

Custom user data

lv_anim_path_cb_t path_cb

Provides path (curve) of animation

int32_t start_value

Start value

int32_t current_value

Current value

int32_t end_value

End value

int32_t duration

Animation duration in ms

int32_t act_time

Ms elapsed since animation started. Set to negative to make delay.

uint32_t reverse_delay

Wait (in ms) after forward play ends and before reverse play begins.

uint32_t reverse_duration

Reverse animation duration in ms

uint32_t repeat_delay

Wait before repeating

uint32_t repeat_cnt

Repeat count for animation

union _lv_anim_t::_lv_anim_path_para_t parameter

uint32_t last_timer_run

uint32_t pause_time

The time when the animation was paused

uint32_t pause_duration

The amount of the time the animation must stay paused for

uint8_t is_paused

Indicates that the animation is paused

uint8_t reverse_play_in_progress

Reverse play is in progress

uint8_t run_round

When not equal to global.anim_state.anim_run_round (which toggles each time animation timer executes), indicates this animation needs to be updated.

uint8_t start_cb_called

Indicates that start_cb was already called

uint8_t early_apply

1: Apply start value immediately even if there is a delay

union _lv_anim_path_para_t

Membri Pubblici

***lv_anim_bezier3_para_t* bezier3**

Parameter used when path is custom_bezier

14.21.2 lv_anim_private.h

Funzioni

void lv_anim_core_init(void)

Init the animation module

```
void lv_anim_core_deinit(void)
    Deinit the animation module
void lv_anim_enable_vsync_mode(bool enable)

struct lv_anim_state_t
```

Membri Pubblici

```
bool anim_list_changed
bool anim_run_round
bool anim_vsync_registered
lv_timer_t *timer
lv_ll_t anim_ll
```

14.21.3 lv_anim_timeline.h**Define**

```
LV_ANIM_TIMELINE_PROGRESS_MAX
```

Typedef

```
typedef struct _lv_anim_timeline_dsc_t lv_anim_timeline_dsc_t
```

Funzioni

```
lv_anim_timeline_t *lv_anim_timeline_create(void)
```

Create an animation timeline.

Ritorna

pointer to the animation timeline.

```
void lv_anim_timeline_delete(lv_anim_timeline_t *at)
```

Delete animation timeline.

Parametri

at -- pointer to the animation timeline.

```
void lv_anim_timeline_add(lv_anim_timeline_t *at, uint32_t start_time, const lv_anim_t *a)
```

Add animation to the animation timeline.

Parametri

- **at** -- pointer to the animation timeline.
- **start_time** -- the time the animation started on the timeline, note that start_time will override the value of delay.
- **a** -- pointer to an animation.

```
uint32_t lv_anim_timeline_start(lv_anim_timeline_t *at)
```

Start the animation timeline.

Parametri

at -- pointer to the animation timeline.

Ritorna

total time spent in animation timeline.

```
void lv_anim_timeline_pause(lv_anim_timeline_t *at)
```

Pause the animation timeline.

Parametri

- at** -- pointer to the animation timeline.

```
void lv_anim_timeline_set_reverse(lv_anim_timeline_t *at, bool reverse)
```

Set the playback direction of the animation timeline.

Parametri

- **at** -- pointer to the animation timeline.
- **reverse** -- whether to play in reverse.

```
void lv_anim_timeline_set_delay(lv_anim_timeline_t *at, uint32_t delay)
```

Set the time to wait before starting the animation. Applies only when playing from the very start, or reverse from the very end.

Parametri

- **at** -- pointer to an animation timeline
- **delay** -- the delay time in milliseconds

```
void lv_anim_timeline_set_repeat_count(lv_anim_timeline_t *at, uint32_t cnt)
```

Make the animation timeline repeat itself.

Parametri

- **at** -- pointer to the animation timeline.
- **cnt** -- repeat count or `LV_ANIM_REPEAT_INFINITE` for infinite repetition. 0: to disable repetition.

```
void lv_anim_timeline_set_repeat_delay(lv_anim_timeline_t *at, uint32_t delay)
```

Set a delay before repeating the animation timeline.

Parametri

- **at** -- pointer to the animation timeline.
- **delay** -- delay in milliseconds before repeating the animation timeline.

```
void lv_anim_timeline_set_progress(lv_anim_timeline_t *at, uint16_t progress)
```

Set the progress of the animation timeline.

Parametri

- **at** -- pointer to the animation timeline.
- **progress** -- set value 0~65535 to map 0~100% animation progress.

```
void lv_anim_timeline_set_user_data(lv_anim_timeline_t *at, void *user_data)
```

Set the user_data of a an animation timeline

Parametri

- **at** -- pointer to the animation timeline.
- **user_data** -- pointer to any data. Only the pointer will be saved.

```
uint32_t lv_anim_timeline_get_playtime(lv_anim_timeline_t *at)
```

Get the time used to play the animation timeline.

Parametri

- at** -- pointer to the animation timeline.

Ritorna

total time spent in animation timeline.

```
bool lv_anim_timeline_get_reverse(lv_anim_timeline_t *at)
```

Get whether the animation timeline is played in reverse.

Parametri

- at** -- pointer to the animation timeline.

Ritorna

return true if it is reverse playback.

```
uint32_t lv_anim_timeline_get_delay(lv_anim_timeline_t *at)
```

Get the wait time when playing from the very start, or reverse from the very end.

Parametri

at -- pointer to an animation timeline

Ritorna

the remaining time in milliseconds

```
uint16_t lv_anim_timeline_get_progress(lv_anim_timeline_t *at)
```

Get the progress of the animation timeline.

Parametri

at -- pointer to the animation timeline.

Ritorna

return value 0~65535 to map 0~100% animation progress.

```
uint32_t lv_anim_timeline_get_repeat_count(lv_anim_timeline_t *at)
```

Get repeat count of the animation timeline.

Parametri

at -- pointer to the animation timeline.

```
uint32_t lv_anim_timeline_get_repeat_delay(lv_anim_timeline_t *at)
```

Get repeat delay of the animation timeline.

Parametri

at -- pointer to the animation timeline.

```
void *lv_anim_timeline_get_user_data(lv_anim_timeline_t *at)
```

Get the user_data of a an animation timeline

Parametri

at -- pointer to the animation timeline.

```
void lv_anim_timeline_merge(lv_anim_timeline_t *dest, const lv_anim_timeline_t *src, int32_t delay)
```

Merge (add) all animations of a timeline to another

Parametri

- **dest** -- merge animation into this timeline
- **src** -- merge the animations of this timeline
- **delay** -- add the animations with this extra delay

```
struct _lv_anim_timeline_dsc_t
```

Membri Pubblici

lv_anim_t anim

uint32_t start_time

uint8_t is_started

uint8_t is_completed

14.21.4 lv_anim_timeline_private.h

```
struct _lv_anim_timeline_t
```

Membri Pubblici

struct *lv_anim_timeline_dsc_t* *anim_dsc

Dynamically allocated anim dsc array

uint32_t anim_dsc_cnt

The length of anim dsc array

uint32_t act_time

Current time of the animation

bool reverse

Reverse playback

uint32_t delay

Delay before starting the animation from any ends

uint32_t repeat_count

Repeat count

uint32_t repeat_delay

Wait before repeat

void *user_data

For any custom data

14.21.5 lv_area.h

Define

LV_COORD_TYPE_SHIFT

LV_COORD_TYPE_MASK

LV_COORD_TYPE(x)

LV_COORD_PLAIN(x)

LV_COORD_TYPE_PX

LV_COORD_TYPE_SPEC

LV_COORD_TYPE_PX_NEG

LV_COORD_IS_PX(x)

LV_COORD_IS_SPEC(x)

LV_COORD_SET_SPEC(x)

LV_COORD_MAX

Max coordinate value

`LV_COORD_MIN`

`LV_SIZE_CONTENT`

`LV_PCT_STORED_MAX`

`LV_PCT_POS_MAX`

`LV_PCT(x)`

`LV_COORD_IS_PCT(x)`

`LV_COORD_GET_PCT(x)`

Enum

enum `lv_align_t`

Alignments

Values:

enumerator `LV_ALIGN_DEFAULT`

enumerator `LV_ALIGN_TOP_LEFT`

enumerator `LV_ALIGN_TOP_MID`

enumerator `LV_ALIGN_TOP_RIGHT`

enumerator `LV_ALIGN_BOTTOM_LEFT`

enumerator `LV_ALIGN_BOTTOM_MID`

enumerator `LV_ALIGN_BOTTOM_RIGHT`

enumerator `LV_ALIGN_LEFT_MID`

enumerator `LV_ALIGN_RIGHT_MID`

enumerator `LV_ALIGN_CENTER`

enumerator `LV_ALIGN_OUT_TOP_LEFT`

enumerator `LV_ALIGN_OUT_TOP_MID`

enumerator `LV_ALIGN_OUT_TOP_RIGHT`

enumerator `LV_ALIGN_OUT_BOTTOM_LEFT`

enumerator **LV_ALIGN_OUT_BOTTOM_MID**

enumerator **LV_ALIGN_OUT_BOTTOM_RIGHT**

enumerator **LV_ALIGN_OUT_LEFT_TOP**

enumerator **LV_ALIGN_OUT_LEFT_MID**

enumerator **LV_ALIGN_OUT_LEFT_BOTTOM**

enumerator **LV_ALIGN_OUT_RIGHT_TOP**

enumerator **LV_ALIGN_OUT_RIGHT_MID**

enumerator **LV_ALIGN_OUT_RIGHT_BOTTOM**

enum **lv_dir_t**

Values:

enumerator **LV_DIR_NONE**

enumerator **LV_DIR_LEFT**

enumerator **LV_DIR_RIGHT**

enumerator **LV_DIR_TOP**

enumerator **LV_DIR_BOTTOM**

enumerator **LV_DIR_HOR**

enumerator **LV_DIR_VER**

enumerator **LV_DIR_ALL**

Funzioni

void **lv_area_set**(*lv_area_t* *area_p, int32_t x1, int32_t y1, int32_t x2, int32_t y2)

Initialize an area

Parametri

- **area_p** -- pointer to an area
- **x1** -- left coordinate of the area
- **y1** -- top coordinate of the area
- **x2** -- right coordinate of the area
- **y2** -- bottom coordinate of the area

static inline void **lv_area_copy**(*lv_area_t* *dest, const *lv_area_t* *src)

Copy an area

Parametri

- **dest** -- pointer to the destination area
- **src** -- pointer to the source area

int32_t **lv_area_get_width**(const *lv_area_t* *area_p)

Get the width of an area

Parametri

area_p -- pointer to an area

Ritorna

the width of the area (if $x1 == x2 \rightarrow \text{width} = 1$)

int32_t **lv_area_get_height**(const *lv_area_t* *area_p)

Get the height of an area

Parametri

area_p -- pointer to an area

Ritorna

the height of the area (if $y1 == y2 \rightarrow \text{height} = 1$)

void **lv_area_set_width**(*lv_area_t* *area_p, int32_t w)

Set the width of an area

Parametri

- **area_p** -- pointer to an area
- **w** -- the new width of the area ($w == 1$ makes $x1 == x2$)

void **lv_area_set_height**(*lv_area_t* *area_p, int32_t h)

Set the height of an area

Parametri

- **area_p** -- pointer to an area
- **h** -- the new height of the area ($h == 1$ makes $y1 == y2$)

uint32_t **lv_area_get_size**(const *lv_area_t* *area_p)

Return with area of an area ($x * y$)

Parametri

area_p -- pointer to an area

Ritorna

size of area

void **lv_area_increase**(*lv_area_t* *area, int32_t w_extra, int32_t h_extra)

void **lv_area_move**(*lv_area_t* *area, int32_t x_ofs, int32_t y_ofs)

void **lv_area_align**(const *lv_area_t* *base, *lv_area_t* *to_align, *lv_align_t* align, int32_t ofs_x, int32_t ofs_y)

Align an area to another

Parametri

- **base** -- an area where the other will be aligned
- **to_align** -- the area to align
- **align** -- `LV_ALIGN_...`
- **ofs_x** -- X offset
- **ofs_y** -- Y offset

void **lv_point_transform**(*lv_point_t* *point, int32_t angle, int32_t scale_x, int32_t scale_y, const *lv_point_t* *pivot, bool zoom_first)

Transform a point

Parametri

- **point** -- pointer to a point
- **angle** -- angle with 0.1 resolutions (123 means 12.3°)
- **scale_x** -- horizontal zoom, 256 means 100%
- **scale_y** -- vertical zoom, 256 means 100%

- **pivot** -- pointer to the pivot point of the transformation
- **zoom_first** -- true: zoom first and rotate after that; else: opposite order

```
void lv_point_array_transform(lv_point_t *points, size_t count, int32_t angle, int32_t scale_x, int32_t scale_y,
                             const lv_point_t *pivot, bool zoom_first)
```

Transform an array of points

Parametri

- **points** -- pointer to an array of points
- **count** -- number of points in the array
- **angle** -- angle with 0.1 resolutions (123 means 12.3°)
- **scale_x** -- horizontal zoom, 256 means 100%
- **scale_y** -- vertical zoom, 256 means 100%
- **pivot** -- pointer to the pivot point of the transformation
- **zoom_first** -- true: zoom first and rotate after that; else: opposite order

```
lv_point_t lv_point_from_precise(const lv_point_precise_t *p)
```

```
lv_point_precise_t lv_point_to_precise(const lv_point_t *p)
```

```
void lv_point_set(lv_point_t *p, int32_t x, int32_t y)
```

```
void lv_point_precise_set(lv_point_precise_t *p, lv_value_precise_t x, lv_value_precise_t y)
```

```
void lv_point_swap(lv_point_t *p1, lv_point_t *p2)
```

```
void lv_point_precise_swap(lv_point_precise_t *p1, lv_point_precise_t *p2)
```

```
int32_t lv_pct(int32_t x)
```

Convert a percentage value to int32_t. Percentage values are stored in special range

Parametri

- x -- the percentage (0..1000)

Ritorna

a coordinate that stores the percentage

```
int32_t lv_pct_to_px(int32_t v, int32_t base)
```

```
struct lv_point_t
```

#include <lv_area.h> Represents a point on the screen.

Membri Pubblici

```
int32_t x
```

```
int32_t y
```

```
struct lv_point_precise_t
```

Membri Pubblici

```
lv_value_precise_t x
```

```
lv_value_precise_t y
```

```
struct lv_area_t
```

#include <lv_area.h> Represents an area of the screen.

Membri Pubblici

int32_t x1

int32_t y1

int32_t x2

int32_t y2

14.21.6 lv_area_private.h**Funzioni**void **lv_area_set_pos**(*lv_area_t* *area_p, int32_t x, int32_t y)

Set the position of an area (width and height will be kept)

Parametri

- **area_p** -- pointer to an area
- **x** -- the new x coordinate of the area
- **y** -- the new y coordinate of the area

bool **lv_area_intersect**(*lv_area_t* *res_p, const *lv_area_t* *a1_p, const *lv_area_t* *a2_p)

Get the common parts of two areas

Parametri

- **res_p** -- pointer to an area, the result will be stored here
- **a1_p** -- pointer to the first area
- **a2_p** -- pointer to the second area

Ritorna

false: the two area has NO common parts, res_p is invalid

int8_t **lv_area_diff**(*lv_area_t* res_p[], const *lv_area_t* *a1_p, const *lv_area_t* *a2_p)

Get resulting sub areas after removing the common parts of two areas from the first area

Parametri

- **res_p** -- pointer to an array of areas with a count of 4, the resulting areas will be stored here
- **a1_p** -- pointer to the first area
- **a2_p** -- pointer to the second area

Ritorna

number of results (max 4) or -1 if no intersect

void **lv_area_join**(*lv_area_t* *a_res_p, const *lv_area_t* *a1_p, const *lv_area_t* *a2_p)

Join two areas into a third which involves the other two

Parametri

- **a_res_p** -- pointer to an area, the result will be stored here
- **a1_p** -- pointer to the first area
- **a2_p** -- pointer to the second area

bool **lv_area_is_point_on**(const *lv_area_t* *a_p, const *lv_point_t* *p_p, int32_t radius)

Check if a point is on an area

Parametri

- **a_p** -- pointer to an area
- **p_p** -- pointer to a point
- **radius** -- radius of area (e.g. for rounded rectangle)

Ritorna

false:the point is out of the area

`bool lv_area_is_on(const lv_area_t *a1_p, const lv_area_t *a2_p)`

Check if two area has common parts

Parametri

- **a1_p** -- pointer to an area.
- **a2_p** -- pointer to another area

Ritorna

false: a1_p and a2_p has no common parts

`bool lv_area_is_in(const lv_area_t *ain_p, const lv_area_t *aholder_p, int32_t radius)`

Check if an area is fully on another

Parametri

- **ain_p** -- pointer to an area which could be in 'aholder_p'
- **aholder_p** -- pointer to an area which could involve 'ain_p'
- **radius** -- radius of aholder_p (e.g. for rounded rectangle)

Ritorna

true: ain_p is fully inside aholder_p

`bool lv_area_is_out(const lv_area_t *aout_p, const lv_area_t *aholder_p, int32_t radius)`

Check if an area is fully out of another

Parametri

- **aout_p** -- pointer to an area which could be in 'aholder_p'
- **aholder_p** -- pointer to an area which could involve 'ain_p'
- **radius** -- radius of aholder_p (e.g. for rounded rectangle)

Ritorna

true: aout_p is fully outside aholder_p

`bool lv_area_is_equal(const lv_area_t *a, const lv_area_t *b)`

Check if 2 area is the same

Parametri

- **a** -- pointer to an area
- **b** -- pointer to another area

14.21.7 lv_array.h

Array. The elements are dynamically allocated by the 'lv_mem' module.

Define

`LV_ARRAY_DEFAULT_CAPACITY`

`LV_ARRAY_DEFAULT_SHRINK_RATIO`

Funzioni

`void lv_array_init(lv_array_t *array, uint32_t capacity, uint32_t element_size)`

Init an array.

Parametri

- **array** -- pointer to an `lv_array_t` variable to initialize
- **capacity** -- the initial capacity of the array
- **element_size** -- the size of an element in bytes

`void lv_array_init_from_buf(lv_array_t *array, void *buf, uint32_t capacity, uint32_t element_size)`

Init an array from a buffer.

 **Nota**

The buffer must be large enough to store capacity elements. The array will not release the buffer and reallocate it. The user must ensure that the buffer is valid during the lifetime of the array. And release the buffer when the array is no longer needed.

Parametri

- **array** -- pointer to an `lv_array_t` variable to initialize
- **buf** -- pointer to a buffer to use as the array's data
- **capacity** -- the initial capacity of the array
- **element_size** -- the size of an element in bytes

`bool lv_array_resize(lv_array_t *array, uint32_t new_capacity)`

Resize the array to the given capacity.

Nota

if the new capacity is smaller than the current size, the array will be truncated.

Parametri

- **array** -- pointer to an `lv_array_t` variable
- **new_capacity** -- the new capacity of the array

`void lv_array_deinit(lv_array_t *array)`

Deinit the array, and free the allocated memory

Parametri

array -- pointer to an `lv_array_t` variable to deinitialize

`static inline uint32_t lv_array_size(const lv_array_t *array)`

Return how many elements are stored in the array.

Parametri

array -- pointer to an `lv_array_t` variable

Ritorna

the number of elements stored in the array

`static inline uint32_t lv_array_capacity(const lv_array_t *array)`

Return the capacity of the array, i.e. how many elements can be stored.

Parametri

array -- pointer to an `lv_array_t` variable

Ritorna

the capacity of the array

`static inline bool lv_array_is_empty(const lv_array_t *array)`

Return if the array is empty

Parametri

array -- pointer to an `lv_array_t` variable

Ritorna

true: array is empty; false: array is not empty

`static inline bool lv_array_is_full(const lv_array_t *array)`

Return if the array is full

Parametri

array -- pointer to an `lv_array_t` variable

Ritorna

true: array is full; false: array is not full

`void lv_array_copy(lv_array_t *target, const lv_array_t *source)`

Copy an array to another.

Nota

this will create a new array with the same capacity and size as the source array.

Parametri

- **target** -- pointer to an `lv_array_t` variable to copy to
- **source** -- pointer to an `lv_array_t` variable to copy from

`static inline void lv_array_clear(lv_array_t *array)`

Remove all elements in array.

Parametri

- **array** -- pointer to an `lv_array_t` variable

`void lv_array_shrink(lv_array_t *array)`

Shrink the memory capacity of array if necessary.

Parametri

- **array** -- pointer to an `lv_array_t` variable

`lv_result_t lv_array_remove(lv_array_t *array, uint32_t index)`

Remove the element at the specified position in the array.

Parametri

- **array** -- pointer to an `lv_array_t` variable
- **index** -- the index of the element to remove

Ritorna

`LV_RESULT_OK`: success, otherwise: error

`lv_result_t lv_array_erase(lv_array_t *array, uint32_t start, uint32_t end)`

Remove from the array either a single element or a range of elements ([start, end)).

Nota

This effectively reduces the container size by the number of elements removed.

Nota

When start equals to end, the function has no effect.

Parametri

- **array** -- pointer to an `lv_array_t` variable
- **start** -- the index of the first element to be removed
- **end** -- the index of the first element that is not to be removed

Ritorna

`LV_RESULT_OK`: success, otherwise: error

`lv_result_t lv_array_concat(lv_array_t *array, const lv_array_t *other)`

Concatenate two arrays. Adds new elements to the end of the array.

Nota

The destination array is automatically expanded as necessary.

Parametri

- **array** -- pointer to an `lv_array_t` variable
- **other** -- pointer to the array to concatenate

Ritorna

`LV_RESULT_OK`: success, otherwise: error

`lv_result_t lv_array_push_back(lv_array_t *array, const void *element)`

Push back element. Adds a new element to the end of the array. If the array capacity is not enough for the new element, the array will be resized automatically.

Nota

If the element is NULL, it will be added as an empty element.

Parametri

- **array** -- pointer to an `lv_array_t` variable
- **element** -- pointer to the element to add. NULL to push an empty element.

Ritorna

`LV_RESULT_OK`: success, otherwise: error

`lv_result_t lv_array_assign(lv_array_t *array, uint32_t index, const void *value)`

Assigns one content to the array, replacing its current content.

Parametri

- **array** -- pointer to an `lv_array_t` variable
- **index** -- the index of the element to replace
- **value** -- pointer to the elements to add

Ritorna

true: success; false: error

`void *lv_array_at(const lv_array_t *array, uint32_t index)`

Returns a pointer to the element at position n in the array.

Parametri

- **array** -- pointer to an `lv_array_t` variable
- **index** -- the index of the element to return

Ritorna

a pointer to the requested element, NULL if index is out of range

`static inline void *lv_array_front(const lv_array_t *array)`

Returns a pointer to the first element in the array.

Parametri

- **array** -- pointer to an `lv_array_t` variable

Ritorna

a pointer to the first element in the array

`static inline void *lv_array_back(const lv_array_t *array)`

Returns a pointer to the last element in the array.

Parametri

- **array** -- pointer to an `lv_array_t` variable

`struct _lv_array_t`

`#include <lv_array.h>` Description of a array

Membri Pubblici

`uint8_t *data`

`uint32_t size`

`uint32_t capacity`

`uint32_t element_size`

```
bool inner_alloc
```

14.21.8 lv_assert.h

Define

`LV_ASSERT(expr)`

`LV_ASSERT_MSG(expr, msg)`

`LV_ASSERT_FORMAT_MSG(expr, format, ...)`

`LV_ASSERT_NULL(p)`

`LV_ASSERT_MALLOC(p)`

`LV_ASSERT_MEM_INTEGRITY()`

14.21.9 lv_async.h

Typedef

`typedef void (*lv_async_cb_t)(void*)`

Type for async callback.

Funzioni

`lv_result_t lv_async_call(lv_async_cb_t async_xcb, void *user_data)`

Call an asynchronous function the next time `lv_timer_handler()` is run. This function is likely to return **before** the call actually happens!

Parametri

- **async_xcb** -- a callback which is the task itself. (the 'x' in the argument name indicates that it's not a fully generic function because it not follows the `func_name(object, callback, ...)` convention)
- **user_data** -- custom parameter

`lv_result_t lv_async_call_cancel(lv_async_cb_t async_xcb, void *user_data)`

Cancel an asynchronous function call

Parametri

- **async_xcb** -- a callback which is the task itself.
- **user_data** -- custom parameter

14.21.10 lv_bidi.h

Define

`LV_BIDI_LRO`

Special non printable strong characters. They can be inserted to texts to affect the run's direction

`LV_BIDI_RLO`

Enum

`enum lv_base_dir_t`

Values:

enumerator **LV_BASE_DIR_LTR**

enumerator **LV_BASE_DIR_RTL**

enumerator **LV_BASE_DIR_AUTO**

enumerator **LV_BASE_DIR_NEUTRAL**

enumerator **LV_BASE_DIR_WEAK**

Funzioni

`void lv_bidi_calculate_align(lv_text_align_t *align, lv_base_dir_t *base_dir, const char *txt)`

Get the real text alignment from the a text alignment, base direction and a text.

Parametri

- **align** -- LV_TEXT_ALIGN_..., write back the calculated align here (LV_TEXT_ALIGN_LEFT/RIGHT/CENTER)
- **base_dir** -- LV_BASE_DIR_..., write the calculated base dir here (LV_BASE_DIR_LTR/RTL)
- **txt** -- a text, used with LV_BASE_DIR_AUTO to determine the base direction

`void lv_bidi_set_custom_neutrals_static(const char *neutrals)`

Set custom neutrals string

Parametri

neutrals -- default " \t\n\r.,;`\"!%/?-=()[]{}<>@#&\$!"

14.21.11 lv_bidi_private.h

Funzioni

`void lv_bidi_process(const char *str_in, char *str_out, lv_base_dir_t base_dir)`

Convert a text to get the characters in the correct visual order according to Unicode Bidirectional Algorithm

Parametri

- **str_in** -- the text to process
- **str_out** -- store the result here. Has the be `strlen(str_in)` length
- **base_dir** -- LV_BASE_DIR_LTR or LV_BASE_DIR_RTL

`lv_base_dir_t lv_bidi_detect_base_dir(const char *txt)`

Auto-detect the direction of a text based on the first strong character

Parametri

txt -- the text to process

Ritorna

LV_BASE_DIR_LTR or LV_BASE_DIR_RTL

`uint16_t lv_bidi_get_logical_pos(const char *str_in, char **bidi_txt, uint32_t len, lv_base_dir_t base_dir, uint32_t visual_pos, bool *is_rtl)`

Get the logical position of a character in a line

Parametri

- **str_in** -- the input string. Can be only one line.
- **bidi_txt** -- internally the text is bidi processed which buffer can be get here. If not required anymore has to freed with `lv_free()` Can be NULL is unused
- **len** -- length of the line in character count
- **base_dir** -- base direction of the text: LV_BASE_DIR_LTR or LV_BASE_DIR_RTL
- **visual_pos** -- the visual character position which logical position should be get
- **is_rtl** -- tell the char at `visual_pos` is RTL or LTR context

Ritorna

the logical character position

```
uint16_t lv_bidi_get_visual_pos(const char *str_in, char **bidi_txt, uint16_t len, lv_base_dir_t base_dir,
                                uint32_t logical_pos, bool *is_rtl)
```

Get the visual position of a character in a line

Parametri

- **str_in** -- the input string. Can be only one line.
- **bidi_txt** -- internally the text is bidi processed which buffer can be get here. If not required anymore has to freed with `lv_free()`. Can be NULL if unused
- **len** -- length of the line in character count
- **base_dir** -- base direction of the text: LV_BASE_DIR_LTR or LV_BASE_DIR_RTL
- **logical_pos** -- the logical character position which visual position should be get
- **is_rtl** -- tell the char at `logical_pos` is RTL or LTR context

Ritorna

the visual character position

```
void lv_bidi_process_paragraph(const char *str_in, char *str_out, uint32_t len, lv_base_dir_t base_dir,
                               uint16_t *pos_conv_out, uint16_t pos_conv_len)
```

Bidi process a paragraph of text

Parametri

- **str_in** -- the string to process
- **str_out** -- store the result here
- **len** -- length of the text
- **base_dir** -- base dir of the text
- **pos_conv_out** -- an `uint16_t` array to store the related logical position of the character. Can be NULL if unused
- **pos_conv_len** -- length of `pos_conv_out` in element count

14.21.12 lv_circle_buf.h

Typedef

```
typedef bool (*lv_circle_buf_fill_cb_t)(void *buf, uint32_t buff_len, int32_t index, void *user_data)
```

Funzioni

```
lv_circle_buf_t *lv_circle_buf_create(uint32_t capacity, uint32_t element_size)
```

Create a circle buffer

Parametri

- **capacity** -- the maximum number of elements in the buffer
- **element_size** -- the size of an element in bytes

Ritorna

pointer to the created buffer

```
lv_circle_buf_t *lv_circle_buf_create_from_buf(void *buf, uint32_t capacity, uint32_t element_size)
```

Create a circle buffer from an existing buffer

Parametri

- **buf** -- pointer to a buffer
- **capacity** -- the maximum number of elements in the buffer
- **element_size** -- the size of an element in bytes

Ritorna

pointer to the created buffer

```
lv_circle_buf_t *lv_circle_buf_create_from_array(const lv_array_t *array)
```

Create a circle buffer from an existing array

Parametri

array -- pointer to an array

Ritorna

pointer to the created buffer

`lv_result_t lv_circle_buf_resize(lv_circle_buf_t *circle_buf, uint32_t capacity)`

Resize the buffer

Parametri

- `circle_buf` -- pointer to a buffer
- `capacity` -- the new capacity of the buffer

Ritorna

LV_RESULT_OK: the buffer is resized; LV_RESULT_INVALID: the buffer is not resized

`void lv_circle_buf_destroy(lv_circle_buf_t *circle_buf)`

Destroy a circle buffer

Parametri

`circle_buf` -- pointer to buffer

`uint32_t lv_circle_buf_size(const lv_circle_buf_t *circle_buf)`

Get the size of the buffer

Parametri

`circle_buf` -- pointer to buffer

Ritorna

the number of elements in the buffer

`uint32_t lv_circle_buf_capacity(const lv_circle_buf_t *circle_buf)`

Get the capacity of the buffer

Parametri

`circle_buf` -- pointer to buffer

Ritorna

the maximum number of elements in the buffer

`uint32_t lv_circle_buf_remain(const lv_circle_buf_t *circle_buf)`

Get the remaining space in the buffer

Parametri

`circle_buf` -- pointer to buffer

Ritorna

the number of elements that can be written to the buffer

`bool lv_circle_buf_is_empty(const lv_circle_buf_t *circle_buf)`

Check if the buffer is empty

Parametri

`circle_buf` -- pointer to buffer

Ritorna

true: the buffer is empty; false: the buffer is not empty

`bool lv_circle_buf_is_full(const lv_circle_buf_t *circle_buf)`

Check if the buffer is full

Parametri

`circle_buf` -- pointer to buffer

Ritorna

true: the buffer is full; false: the buffer is not full

`void lv_circle_buf_reset(lv_circle_buf_t *circle_buf)`

Reset the buffer

Parametri

`circle_buf` -- pointer to buffer

Ritorna

LV_RESULT_OK: the buffer is reset; LV_RESULT_INVALID: the buffer is not reset

`void *lv_circle_buf_head(const lv_circle_buf_t *circle_buf)`

Get the head of the buffer

Parametri
circle_buf -- pointer to buffer

Ritorna
pointer to the head of the buffer

void *lv_circle_buf_tail(const lv_circle_buf_t *circle_buf)

Get the tail of the buffer

Parametri
circle_buf -- pointer to buffer

Ritorna
pointer to the tail of the buffer

lv_result_t lv_circle_buf_read(lv_circle_buf_t *circle_buf, void *data)

Read a value

Parametri

- **circle_buf** -- pointer to buffer
- **data** -- pointer to a variable to store the read value

Ritorna
LV_RESULT_OK: the value is read; LV_RESULT_INVALID: the value is not read

lv_result_t lv_circle_buf_write(lv_circle_buf_t *circle_buf, const void *data)

Write a value

Parametri

- **circle_buf** -- pointer to buffer
- **data** -- pointer to the value to write

Ritorna
LV_RESULT_OK: the value is written; LV_RESULT_INVALID: the value is not written

uint32_t lv_circle_buf_fill(lv_circle_buf_t *circle_buf, uint32_t count, *lv_circle_buf_fill_cb_t* fill_cb, void *user_data)

Fill the buffer with values

Parametri

- **circle_buf** -- pointer to buffer
- **count** -- the number of values to fill
- **fill_cb** -- the callback function to fill the buffer
- **user_data** --

Ritorna
the number of values filled

lv_result_t lv_circle_buf_skip(lv_circle_buf_t *circle_buf)

Skip a value

Parametri
circle_buf -- pointer to buffer

Ritorna
LV_RESULT_OK: the value is skipped; LV_RESULT_INVALID: the value is not skipped

lv_result_t lv_circle_buf_peek(const lv_circle_buf_t *circle_buf, void *data)

Peek a value

Parametri

- **circle_buf** -- pointer to buffer
- **data** -- pointer to a variable to store the peeked value

Ritorna
LV_RESULT_OK: the value is peeked; LV_RESULT_INVALID: the value is not peeked

lv_result_t lv_circle_buf_peek_at(const lv_circle_buf_t *circle_buf, uint32_t index, void *data)

Peek a value at an index

Parametri

- **circle_buf** -- pointer to buffer
- **index** -- the index of the value to peek, if the index is greater than the size of the buffer, it will return loopy.

- **data** -- pointer to a variable to store the peeked value

Ritorna

LV_RESULT_OK: the value is peeked; LV_RESULT_INVALID: the value is not peeked

14.21.13 lv_color.h**Define**

LV_COLOR_NATIVE_WITH_ALPHA_SIZE

LV_OPA_MIN

Fully transparent if opa <= LV_OPA_MIN

LV_OPA_MAX

Fully cover if opa >= LV_OPA_MAX

LV_COLOR_FORMAT_GET_BPP(cf)

Get the pixel size of a color format in bits, bpp

 **Vedi anche**

[lv_color_format_get_bpp](#)

Parametri

- **cf** -- a color format (LV_COLOR_FORMAT_...)

Ritorna

the pixel size in bits

LV_COLOR_FORMAT_GET_SIZE(cf)

Get the pixel size of a color format in bytes

 **Vedi anche**

[lv_color_format_get_size](#)

Parametri

- **cf** -- a color format (LV_COLOR_FORMAT_...)

Ritorna

the pixel size in bytes

LV_COLOR_FORMAT_IS_ALPHA_ONLY(cf)

LV_COLOR_FORMAT_IS_INDEXED(cf)

LV_COLOR_FORMAT_IS_YUV(cf)

LV_COLOR_INDEXED_PALETTE_SIZE(cf)

LV_COLOR_MAKE(r8, g8, b8)

LV_OPA_MIX2(a1, a2)

LV_OPA_MIX3(a1, a2, a3)

Enum

enum **lv_opacity_level_t**

Opacity percentages.

Values:

enumerator **LV_OPA_TRANSP**

enumerator **LV_OPA_0**

enumerator **LV_OPA_10**

enumerator **LV_OPA_20**

enumerator **LV_OPA_30**

enumerator **LV_OPA_40**

enumerator **LV_OPA_50**

enumerator **LV_OPA_60**

enumerator **LV_OPA_70**

enumerator **LV_OPA_80**

enumerator **LV_OPA_90**

enumerator **LV_OPA_100**

enumerator **LV_OPA_COVER**

enum **lv_color_format_t**

Values:

enumerator **LV_COLOR_FORMAT_UNKNOWN**

enumerator **LV_COLOR_FORMAT_RAW**

enumerator **LV_COLOR_FORMAT_RAW_ALPHA**

enumerator **LV_COLOR_FORMAT_L8**

enumerator **LV_COLOR_FORMAT_I1**

enumerator **LV_COLOR_FORMAT_I2**

enumerator **LV_COLOR_FORMAT_I4**

enumerator **LV_COLOR_FORMAT_I8**

enumerator **LV_COLOR_FORMAT_A8**

enumerator **LV_COLOR_FORMAT_RGB565**

enumerator **LV_COLOR_FORMAT_ARGB8565**

Not supported by sw renderer yet.

enumerator **LV_COLOR_FORMAT_RGB565A8**

Color array followed by Alpha array

enumerator **LV_COLOR_FORMAT_AL88**

L8 with alpha >

enumerator **LV_COLOR_FORMAT_RGB565_SWAPPED**

enumerator **LV_COLOR_FORMAT_RGB888**

enumerator **LV_COLOR_FORMAT_ARGB8888**

enumerator **LV_COLOR_FORMAT_XRGB8888**

enumerator **LV_COLOR_FORMAT_ARGB8888_PREMULTIPLIED**

enumerator **LV_COLOR_FORMAT_A1**

enumerator **LV_COLOR_FORMAT_A2**

enumerator **LV_COLOR_FORMAT_A4**

enumerator **LV_COLOR_FORMAT_ARGB1555**

enumerator **LV_COLOR_FORMAT_ARGB4444**

enumerator **LV_COLOR_FORMAT_ARGB2222**

enumerator **LV_COLOR_FORMAT_YUV_START**

enumerator **LV_COLOR_FORMAT_I420**

enumerator **LV_COLOR_FORMAT_I422**

enumerator **LV_COLOR_FORMAT_I444**

enumerator **LV_COLOR_FORMAT_I400**

enumerator **LV_COLOR_FORMAT_NV21**

enumerator **LV_COLOR_FORMAT_NV12**

enumerator **LV_COLOR_FORMAT_YUY2**

enumerator **LV_COLOR_FORMAT_UYVY**

enumerator **LV_COLOR_FORMAT_YUV_END**

enumerator **LV_COLOR_FORMAT_PROPRIETARY_START**

enumerator **LV_COLOR_FORMAT_NEMA_TSC_START**

enumerator **LV_COLOR_FORMAT_NEMA_TSC4**

enumerator **LV_COLOR_FORMAT_NEMA_TSC6**

enumerator **LV_COLOR_FORMAT_NEMA_TSC6A**

enumerator **LV_COLOR_FORMAT_NEMA_TSC6AP**

enumerator **LV_COLOR_FORMAT_NEMA_TSC12**

enumerator **LV_COLOR_FORMAT_NEMA_TSC12A**

enumerator **LV_COLOR_FORMAT_NEMA_TSC_END**

enumerator **LV_COLOR_FORMAT_NATIVE**

enumerator **LV_COLOR_FORMAT_NATIVE_WITH_ALPHA**

Funzioni

`uint8_t lv_color_format_get_bpp(lv_color_format_t cf)`

Get the pixel size of a color format in bits, bpp

 **Vedi anche**

[LV_COLOR_FORMAT_GET_BPP](#)

Parametri

`cf` -- a color format (`LV_COLOR_FORMAT_...`)

Ritorna

the pixel size in bits

`uint8_t lv_color_format_get_size(lv_color_format_t cf)`

Get the pixel size of a color format in bytes

 **Vedi anche**

[LV_COLOR_FORMAT_GET_SIZE](#)

Parametri

- `cf` -- a color format (LV_COLOR_FORMAT_...)

Ritorna

the pixel size in bytes

`bool lv_color_format_has_alpha(lv_color_format_t src_cf)`

Check if a color format has alpha channel or not

Parametri

- `src_cf` -- a color format (LV_COLOR_FORMAT_...)

Ritorna

true: has alpha channel; false: doesn't have alpha channel

`lv_color32_t lv_color_to_32(lv_color_t color, lv_opa_t opa)`

Create an ARGB8888 color from RGB888 + alpha

Parametri

- `color` -- an RGB888 color
- `opa` -- the alpha value

Ritorna

the ARGB8888 color

`uint32_t lv_color_to_int(lv_color_t c)`

Convert an RGB888 color to an integer

Parametri

- `c` -- an RGB888 color

Ritorna

`c` as an integer

`bool lv_color_eq(lv_color_t c1, lv_color_t c2)`

Check if two RGB888 color are equal

Parametri

- `c1` -- the first color
- `c2` -- the second color

Ritorna

true: equal

`bool lv_color32_eq(lv_color32_t c1, lv_color32_t c2)`

Check if two ARGB8888 color are equal

Parametri

- `c1` -- the first color
- `c2` -- the second color

Ritorna

true: equal

`lv_color_t lv_color_hex(uint32_t c)`

Create a color from 0x000000..0xffffffff input

Parametri

- `c` -- the hex input

Ritorna

the color

`lv_color_t lv_color_make(uint8_t r, uint8_t g, uint8_t b)`

Create an RGB888 color

Parametri

- **r** -- the red channel (0..255)
- **g** -- the green channel (0..255)
- **b** -- the blue channel (0..255)

Ritorna

the color

```
lv_color32_t lv_color32_make(uint8_t r, uint8_t g, uint8_t b, uint8_t a)
```

Create an ARGB8888 color

Parametri

- **r** -- the red channel (0..255)
- **g** -- the green channel (0..255)
- **b** -- the blue channel (0..255)
- **a** -- the alpha channel (0..255)

Ritorna

the color

```
lv_color_t lv_color_hex3(uint32_t c)
```

Create a color from 0x000..0xffff input

Parametri

- **c** -- the hex input (e.g. 0x123 will be 0x112233)

Ritorna

the color

```
static inline bool lv_color_is_in_range(lv_color_t color, lv_color_t l_color, lv_color_t h_color)
```

Check if a color with an RGB888 color is within the color range defined by l_color and h_color.

Parametri

- **color** -- the color to check
- **l_color** -- the lower bound color
- **h_color** -- the upper bound color

Ritorna

true: pixel is within the color range

```
static inline lv_color_t lv_color16_to_color(lv_color16_t c)
```

Convert a RGB565 color to RGB888

Parametri

- **c** -- a RGB565 color on *lv_color16_t*

Ritorna

the color

```
uint16_t lv_color_to_u16(lv_color_t color)
```

Convert am RGB888 color to RGB565 stored in *uint16_t*

Parametri

- **color** -- and RGB888 color

Ritorna

color as RGB565 on *uin16_t*

```
uint32_t lv_color_to_u32(lv_color_t color)
```

Convert am RGB888 color to XRGB8888 stored in *uint32_t*

Parametri

- **color** -- and RGB888 color

Ritorna

color as XRGB8888 on *uin32_t* (the alpha channel is always set to 0xFF)

```
uint16_t lv_color_16_16_mix(uint16_t c1, uint16_t c2, uint8_t mix)
```

Mix two RGB565 colors

Parametri

- **c1** -- the first color (typically the foreground color)
- **c2** -- the second color (typically the background color)
- **mix** -- 0..255, or LV_OPA_0/10/20...

Ritorna

`mix == 0: c2 mix == 255: c1 mix == 128: 0.5 x c1 + 0.5 x c2`

`lv_color_t lv_color_lighten(lv_color_t c, lv_opa_t lvl)`

Mix white to a color

Parametri

- `c` -- the base color
- `lvl` -- the intensity of white (0: no change, 255: fully white)

Ritorna

the mixed color

`lv_color_t lv_color_darken(lv_color_t c, lv_opa_t lvl)`

Mix black to a color

Parametri

- `c` -- the base color
- `lvl` -- the intensity of black (0: no change, 255: fully black)

Ritorna

the mixed color

`lv_color_t lv_color_hsv_to_rgb(uint16_t h, uint8_t s, uint8_t v)`

Convert a HSV color to RGB

Parametri

- `h` -- hue [0..359]
- `s` -- saturation [0..100]
- `v` -- value [0..100]

Ritorna

the given RGB color in RGB (with LV_COLOR_DEPTH depth)

`lv_color_hsv_t lv_color_rgb_to_hsv(uint8_t r8, uint8_t g8, uint8_t b8)`

Convert a 32-bit RGB color to HSV

Parametri

- `r8` -- 8-bit red
- `g8` -- 8-bit green
- `b8` -- 8-bit blue

Ritorna

the given RGB color in HSV

`lv_color_hsv_t lv_color_to_hsv(lv_color_t color)`

Convert a color to HSV

Parametri

`color` -- color

Ritorna

the given color in HSV

`lv_color_t lv_color_white(void)`

A helper for white color

Ritorna

a white color

`lv_color_t lv_color_black(void)`

A helper for black color

Ritorna

a black color

`void lv_color_premultiply(lv_color32_t *c)`

`void lv_color16_premultiply(lv_color16_t *c, lv_opa_t a)`

`uint8_t lv_color_luminance(lv_color_t c)`

Get the luminance of a color: luminance = 0.3 R + 0.59 G + 0.11 B

Parametri`c` -- a color**Ritorna**

the brightness [0..255]

`uint8_t lv_color16_luminance(const lv_color16_t c)`

Get the luminance of a color16: luminance = 0.3 R + 0.59 G + 0.11 B

Parametri`c` -- a color**Ritorna**

the brightness [0..255]

`uint8_t lv_color24_luminance(const uint8_t *c)`

Get the luminance of a color24: luminance = 0.3 R + 0.59 G + 0.11 B

Parametri`c` -- a color**Ritorna**

the brightness [0..255]

`uint8_t lv_color32_luminance(lv_color32_t c)`

Get the luminance of a color32: luminance = 0.3 R + 0.59 G + 0.11 B

Parametri`c` -- a color**Ritorna**

the brightness [0..255]

`static inline uint16_t lv_color_swap_16(uint16_t c)`

Swap the endianness of an rgb565 color

Parametri`c` -- a color**Ritorna**

the swapped color

Variabili`const lv_color_filter_dsc_t lv_color_filter_shade``struct lv_color_t`**Membri Pubblici**`uint8_t blue``uint8_t green``uint8_t red``struct lv_color16_t`**Membri Pubblici**`uint16_t blue``uint16_t green`

```
uint16_t red

struct lv_color32_t
```

Membri Pubblici

```
    uint8_t blue

    uint8_t green

    uint8_t red

    uint8_t alpha
```

```
struct lv_color_hsv_t
```

Membri Pubblici

```
    uint16_t h

    uint8_t s

    uint8_t v
```

```
struct lv_color16a_t
```

Membri Pubblici

```
    uint8_t lumi

    uint8_t alpha
```

14.21.14 lv_color_op.h

Typedef

```
typedef lv_color_t (*lv_color_filter_cb_t)(const struct _lv_color_filter_desc_t*, lv_color_t, lv_opa_t)
```

Funzioni

```
lv_color_t lv_color_mix(lv_color_t c1, lv_color_t c2, uint8_t mix)
```

Mix two colors with a given ratio.

Parametri

- **c1** -- the first color to mix (usually the foreground)
- **c2** -- the second color to mix (usually the background)
- **mix** -- The ratio of the colors. 0: full c2, 255: full c1, 127: half c1 and half c2

Ritorna

the mixed color

`lv_color32_t lv_color_mix32(lv_color32_t fg, lv_color32_t bg)`

Nota

Use bg.alpha in the return value

Nota

Use fg.alpha as mix ratio

Parametri

- **fg** --
- **bg** --

Ritorna

`lv_color32_t lv_color_mix32_premultiplied(lv_color32_t fg, lv_color32_t bg)`

Blends two premultiplied ARGB8888 colors while maintaining correct alpha compositing.

This function correctly blends the foreground (fg) and background (bg) colors, ensuring that the output remains in a premultiplied alpha format.

Nota

If the foreground is fully opaque, it is returned as is.

Nota

If the foreground is fully transparent, the background is returned.

Parametri

- **fg** -- The foreground color in premultiplied ARGB8888 format.
- **bg** -- The background color in premultiplied ARGB8888 format.

Ritorna

The resulting blended color in premultiplied ARGB8888 format.

`uint8_t lv_color_brightness(lv_color_t c)`

Get the brightness of a color

Parametri

c -- a color

Ritorna

brightness in range [0..255]

`void lv_color_filter_dsc_init(lv_color_filter_dsc_t *dsc, lv_color_filter_cb_t cb)`

`lv_color32_t lv_color_over32(lv_color32_t fg, lv_color32_t bg)`

Blend two colors that have not been pre-multiplied using their alpha values

Parametri

- **fg** -- the foreground color
- **bg** -- the background color

Ritorna

result color

`struct _lv_color_filter_dsc_t`

Membri Pubblici

`lv_color_filter_cb_t filter_cb`

`void *user_data`

14.21.15 lv_color_op_private.h**14.21.16 lv_event.h****Typedef**

`typedef void (*lv_event_cb_t)(lv_event_t *e)`

Enum

`enum lv_event_code_t`

Type of event being sent to Widget

Values:

enumerator **LV_EVENT_ALL**

enumerator **LV_EVENT_PRESSED**

Input device events Widget has been pressed

enumerator **LV_EVENT_PRESSING**

Widget is being pressed (sent continuously while pressing)

enumerator **LV_EVENT_PRESS_LOST**

Widget is still being pressed but slid cursor/finger off Widget

enumerator **LV_EVENT_SHORT_CLICKED**

Widget was pressed for a short period of time, then released. Not sent if scrolled.

enumerator **LV_EVENT_SINGLE_CLICKED**

Sent for first short click within a small distance and short time

enumerator **LV_EVENT_DOUBLE_CLICKED**

Sent for second short click within small distance and short time

enumerator **LV_EVENT_TRIPLE_CLICKED**

Sent for third short click within small distance and short time

enumerator **LV_EVENT_LONG_PRESSED**

Object has been pressed for at least `long_press_time`. Not sent if scrolled.

enumerator **LV_EVENT_LONG_PRESSED_REPEAT**

Sent after `long_press_time` in every `long_press_repeat_time` ms. Not sent if scrolled.

enumerator LV_EVENT_CLICKED

Sent on release if not scrolled (regardless to long press)

enumerator LV_EVENT_RELEASED

Sent in every cases when Widget has been released

enumerator LV_EVENT_SCROLL_BEGIN

Scrolling begins. The event parameter is a pointer to the animation of the scroll. Can be modified

enumerator LV_EVENT_SCROLL_THROW_BEGIN**enumerator LV_EVENT_SCROLL_END**

Scrolling ends

enumerator LV_EVENT_SCROLL

Scrolling

enumerator LV_EVENT_GESTURE

A gesture is detected. Get gesture with `lv_indev_get_gesture_dir(lv_indev_active());`

enumerator LV_EVENT_KEY

A key is sent to Widget. Get key with `lv_indev_get_key(lv_indev_active());`

enumerator LV_EVENT_ROTARY

An encoder or wheel was rotated. Get rotation count with `lv_event_get_rotary_diff(e);`

enumerator LV_EVENT_FOCUSED

Widget received focus

enumerator LV_EVENT_DEFOCUSSED

Widget's focus has been lost

enumerator LV_EVENT_LEAVE

Widget's focus has been lost but is still selected

enumerator LV_EVENT_HIT_TEST

Perform advanced hit-testing

enumerator LV_EVENT_INDEV_RESET

Indev has been reset

enumerator LV_EVENT_HOVER_OVER

Indev hover over object

enumerator LV_EVENT_HOVER_LEAVE

Indev hover leave object

enumerator LV_EVENT_COVER_CHECK

Drawing events Check if Widget fully covers an area. The event parameter is `lv_cover_check_info_t *`.

enumerator LV_EVENT_REFR_EXT_DRAW_SIZE

Get required extra draw area around Widget (e.g. for shadow). The event parameter is `int32_t *` to store the size.

enumerator LV_EVENT_DRAW_MAIN_BEGIN

Starting the main drawing phase

enumerator LV_EVENT_DRAW_MAIN

Perform the main drawing

enumerator LV_EVENT_DRAW_MAIN_END

Finishing the main drawing phase

enumerator LV_EVENT_DRAW_POST_BEGIN

Starting the post draw phase (when all children are drawn)

enumerator LV_EVENT_DRAW_POST

Perform the post draw phase (when all children are drawn)

enumerator LV_EVENT_DRAW_POST_END

Finishing the post draw phase (when all children are drawn)

enumerator LV_EVENT_DRAW_TASK_ADDED

Adding a draw task. The `LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS` flag needs to be set

enumerator LV_EVENT_VALUE_CHANGED

Special events Widget's value has changed (i.e. slider moved)

enumerator LV_EVENT_INSERT

Text has been inserted into Widget. The event data is `char *` being inserted.

enumerator LV_EVENT_REFRESH

Notify Widget to refresh something on it (for user)

enumerator LV_EVENT_READY

A process has finished

enumerator LV_EVENT_CANCEL

A process has been cancelled

enumerator LV_EVENT_CREATE

Other events Object is being created

enumerator LV_EVENT_DELETE

Object is being deleted

enumerator LV_EVENT_CHILD_CHANGED

Child was removed, added, or its size, position were changed

enumerator LV_EVENT_CHILD_CREATED

Child was created, always bubbles up to all parents

enumerator LV_EVENT_CHILD_DELETED

Child was deleted, always bubbles up to all parents

enumerator LV_EVENT_SCREEN_UNLOAD_START

A screen unload started, fired immediately when scr_load is called

enumerator LV_EVENT_SCREEN_LOAD_START

A screen load started, fired when the screen change delay is expired

enumerator LV_EVENT_SCREEN_LOADED

A screen was loaded

enumerator LV_EVENT_SCREEN_UNLOADED

A screen was unloaded

enumerator LV_EVENT_SIZE_CHANGED

Object coordinates/size have changed

enumerator LV_EVENT_STYLE_CHANGED

Object's style has changed

enumerator LV_EVENT_LAYOUT_CHANGED

A child's position position has changed due to a layout recalculation

enumerator LV_EVENT_GET_SELF_SIZE

Get internal size of a widget

enumerator LV_EVENT_UPDATE_LAYOUT_COMPLETED

Sent after layout update completes

enumerator LV_EVENT_INVALIDATE_AREA

Events of optional LVGL components An area is invalidated (marked for redraw). `lv_event_get_param(e)` returns a pointer to an `lv_area_t` object with the coordinates of the area to be invalidated. The area can be freely modified if needed to adapt it a special requirement of the display. Usually needed with monochrome displays to invalidate N × 8 rows or columns in one pass.

enumerator LV_EVENT_RESOLUTION_CHANGED

Sent when the resolution changes due to `lv_display_set_resolution()` or `lv_display_set_rotation()`.

enumerator LV_EVENT_COLOR_FORMAT_CHANGED

Sent as a result of any call to `lv_display_set_color_format()`.

enumerator LV_EVENT_REFR_REQUEST

Sent when something happened that requires redraw.

enumerator LV_EVENT_REFR_START

Sent before a refreshing cycle starts. Sent even if there is nothing to redraw.

enumerator LV_EVENT_REFR_READY

Sent when refreshing has been completed (after rendering and calling flush callback). Sent even if no redraw happened.

enumerator LV_EVENT_RENDER_START

Sent just before rendering begins.

enumerator LV_EVENT_RENDER_READY

Sent after rendering has been completed.

enumerator LV_EVENT_FLUSH_START

Sent before flush callback is called.

enumerator LV_EVENT_FLUSH_FINISH

Sent after flush callback call has returned.

enumerator LV_EVENT_FLUSH_WAIT_START

Sent before flush wait callback is called.

enumerator LV_EVENT_FLUSH_WAIT_FINISH

Sent after flush wait callback call has returned.

enumerator LV_EVENT_VSYNC**enumerator LV_EVENT_VSYNC_REQUEST****enumerator LV_EVENT_TRANSLATION_LANGUAGE_CHANGED**

Sent when the translation language changed.

enumerator LV_EVENT_LAST**enumerator LV_EVENT_PREPROCESS**

Number of default events

enumerator LV_EVENT_MARKED_DELETING

This is a flag that can be set with an event so it's processed before the class default event processing

Funzioni

`lv_result_t lv_event_send(lv_event_list_t *list, lv_event_t *e, bool preprocess)`

Event callback. Events are used to notify the user of some action being taken on Widget. For details, see `::lv_event_t`.

`lv_event_dsc_t *lv_event_add(lv_event_list_t *list, lv_event_cb_t cb, lv_event_code_t filter, void *user_data)`

`bool lv_event_remove_dsc(lv_event_list_t *list, lv_event_dsc_t *dsc)`

`uint32_t lv_event_get_count(lv_event_list_t *list)`

`lv_event_dsc_t *lv_event_get_dsc(lv_event_list_t *list, uint32_t index)`

`lv_event_cb_t lv_event_dsc_get_cb(lv_event_dsc_t *dsc)`

`void *lv_event_dsc_get_user_data(lv_event_dsc_t *dsc)`

`bool lv_event_remove(lv_event_list_t *list, uint32_t index)`

`void lv_event_remove_all(lv_event_list_t *list)`

`void *lv_event_get_target(lv_event_t *e)`

Get Widget originally targeted by the event. It's the same even if event was bubbled.

Parametri

`e` -- pointer to the event descriptor

Ritorna

the target of the event_code

`void *lv_event_get_current_target(lv_event_t *e)`

Get current target of the event. It's the Widget for which the event handler being called. If the event is not bubbled it's the same as "normal" target.

Parametri

`e` -- pointer to the event descriptor

Ritorna

pointer to the current target of the event_code

`lv_event_code_t lv_event_get_code(lv_event_t *e)`

Get event code of an event.

Parametri

`e` -- pointer to the event descriptor

Ritorna

the event code. (E.g. LV_EVENT_CLICKED, LV_EVENT_FOCUSED, etc)

`void *lv_event_get_param(lv_event_t *e)`

Get parameter passed when event was sent.

Parametri

`e` -- pointer to the event descriptor

Ritorna

pointer to the parameter

`void *lv_event_get_user_data(lv_event_t *e)`

Get user_data passed when event was registered on Widget.

Parametri

`e` -- pointer to the event descriptor

Ritorna

pointer to the user_data

`void lv_event_stop_bubbling(lv_event_t *e)`

Stop event from bubbling. This is only valid when called in the middle of an event processing chain.

Parametri

`e` -- pointer to the event descriptor

`void lv_event_stop_trickling(lv_event_t *e)`

Stop event from trickling down to children. This is only valid when called in the middle of an event processing chain.

Parametri

`e` -- pointer to the event descriptor

```
void lv_event_stop_processing(lv_event_t *e)
```

Stop processing this event. This is only valid when called in the middle of an event processing chain.

Parametri

e -- pointer to the event descriptor

```
void lv_event_free_user_data_cb(lv_event_t *e)
```

Helper function typically used in LV_EVENT_DELETE to free the event's user_data

Parametri

e -- pointer to an event descriptor

```
uint32_t lv_event_register_id(void)
```

Register a new, custom event ID. It can be used the same way as e.g. LV_EVENT_CLICKED to send custom events

Example:

```
uint32_t LV_EVENT_MINE = 0;
...
e = lv_event_register_id();
...
lv_obj_send_event(obj, LV_EVENT_MINE, &some_data);
```

Ritorna

the new event id

```
const char *lv_event_code_get_name(lv_event_code_t code)
```

Get the name of an event code.

Parametri

code -- the event code

Ritorna

the name of the event code as a string

```
struct lv_event_list_t
```

Membri Pubblici

lv_array_t **array**

uint8_t **is_traversing**

True: the list is being nested traversed

uint8_t **has_marked_deleting**

True: the list has marked deleting objects when some of events are marked as deleting

14.21.17 lv_event_private.h

Funzioni

```
void lv_event_push(lv_event_t *e)
```

```
void lv_event_pop(lv_event_t *e)
```

```
lv_result_t lv_event_push_and_send(lv_event_list_t *event_list, lv_event_code_t code, void *original_target, void *param)
```

```
void lv_event_mark_deleted(void *target)
```

Nested events can be called and one of them might belong to an object that is being deleted. Mark this object's event_temp_data deleted to know that its lv_obj_send_event should return LV_RESULT_INVALID

Parametri

target -- pointer to an event target which was deleted

```
struct _lv_event_dsc_t
```

Membri Pubblici

```
lv_event_cb_t cb
```

```
void *user_data
```

```
uint32_t filter
```

```
struct _lv_event_t
```

Membri Pubblici

```
void *current_target
```

```
void *original_target
```

```
lv_event_code_t code
```

```
void *user_data
```

```
void *param
```

```
lv_event_t *prev
```

```
uint8_t deleted
```

```
uint8_t stop_processing
```

```
uint8_t stop_bubbling
```

```
uint8_t stop_trickling
```

14.21.18 lv_fs.h**Define**

```
LV_FS_MAX_FN_LENGTH
```

```
LV_FS_MAX_PATH_LENGTH
```

```
LV_FS_CACHE_FROM_BUFFER
```

Typedef

```
typedef struct _lv_fs_drv_t lv_fs_drv_t
```

Enum

enum **lv_fs_res_t**

Errors in the file system module.

Values:

enumerator **LV_FS_RES_OK**

enumerator **LV_FS_RES_HW_ERR**

enumerator **LV_FS_RES_FS_ERR**

enumerator **LV_FS_RES_NOT_EX**

enumerator **LV_FS_RES_FULL**

enumerator **LV_FS_RES_LOCKED**

enumerator **LV_FS_RES_DENIED**

enumerator **LV_FS_RES_BUSY**

enumerator **LV_FS_RES_TOUT**

enumerator **LV_FS_RES_NOT_IMP**

enumerator **LV_FS_RES_OUT_OF_MEM**

enumerator **LV_FS_RES_INV_PARAM**

enumerator **LV_FS_RES_UNKNOWN**

enum **lv_fs_mode_t**

File open mode.

Values:

enumerator **LV_FS_MODE_WR**

enumerator **LV_FS_MODE_RD**

enum **lv_fs_whence_t**

Seek modes.

Values:

enumerator **LV_FS_SEEK_SET**

Set the position from absolutely (from the start of file)

enumerator **LV_FS_SEEK_CUR**

Set the position from the current position

enumerator **LV_FS_SEEK_END**

Set the position from the end of the file

Funzioni

void lv_fs_drv_init(lv_fs_drv_t *drv)

Initialize a file system driver with default values. It is used to ensure all fields have known values and not memory junk. After it you can set the fields.

Parametri

drv -- pointer to driver variable to initialize

void lv_fs_drv_register(lv_fs_drv_t *drv)

Add a new drive

Parametri

drv -- pointer to an lv_fs_drv_t structure which is init with the corresponding function pointers. Only pointer is saved, so the driver should be static or dynamically allocated.

lv_fs_drv_t *lv_fs_get_drv(char letter)

Give a pointer to a driver from its letter

Parametri

letter -- the driver-identifier letter

Ritorna

pointer to a driver or NULL if not found

bool lv_fs_is_ready(char letter)

Test if a drive is ready or not. If the ready function was not initialized true will be returned.

Parametri

letter -- letter of the drive

Ritorna

true: drive is ready; false: drive is not ready

lv_fs_res_t lv_fs_open(lv_fs_file_t *file_p, const char *path, lv_fs_mode_t mode)

Open a file

Parametri

- **file_p** -- pointer to a **lv_fs_file_t** variable
- **path** -- path to the file beginning with the driver letter (e.g. S:/folder/file.txt)
- **mode** -- read: FS_MODE_RD, write: FS_MODE_WR, both: FS_MODE_RD | FS_MODE_WR

Ritorna

LV_FS_RES_OK or any error from **lv_fs_res_t** enum

void lv_fs_make_path_from_buffer(lv_fs_path_ex_t *path, char letter, const void *buf, uint32_t size, const char *ext)

Create a special object from buffer/ memory address which looks like a file and can be passed as path to **lv_fs_open** and other functions accepting a path.

For example

```
//Create a PNG file from t a buffer and use it
lv_fs_path_ex_t p;
lv_fs_make_path_from_buffer(&p, 'A', my_buf, my_buf_size, "png");
lv_image_set_src(image1, &p);
```

Parametri

- **path** -- path to a **lv_fs_path_ex** object
- **letter** -- the identifier letter of the driver. E.g. LV_FS_MEMFS_LETTER
- **buf** -- address of the memory buffer
- **size** -- size of the memory buffer in bytes

- **ext** -- the extension, e.g. "png", if NULL no extension will be added.

`lv_result_t lv_fs_get_buffer_from_path(lv_fs_path_ex_t *path, void **buffer, uint32_t *size)`

Get the buffer address and size from a path object

Parametri

- **path** -- pointer to an initialized `lv_fs_path_ex` data
- **buffer** -- pointer to a `void *` variable to store the address
- **size** -- pointer to an `uint32_t` data to store the size

Ritorna

`LV_RESULT_OK`: buffer and size are set; `LV_RESULT_INVALID`: an error happened.

`lv_fs_res_t lv_fs_close(lv_fs_file_t *file_p)`

Close an already opened file

Parametri

- **file_p** -- pointer to a `lv_fs_file_t` variable

Ritorna

`LV_FS_RES_OK` or any error from `lv_fs_res_t` enum

`lv_fs_res_t lv_fs_read(lv_fs_file_t *file_p, void *buf, uint32_t btr, uint32_t *br)`

Read from a file

Parametri

- **file_p** -- pointer to a `lv_fs_file_t` variable
- **buf** -- pointer to a buffer where the read bytes are stored
- **btr** -- Bytes To Read
- **br** -- the number of real read bytes (Bytes Read). NULL if unused.

Ritorna

`LV_FS_RES_OK` or any error from `lv_fs_res_t` enum

`lv_fs_res_t lv_fs_write(lv_fs_file_t *file_p, const void *buf, uint32_t btw, uint32_t *bw)`

Write into a file

Parametri

- **file_p** -- pointer to a `lv_fs_file_t` variable
- **buf** -- pointer to a buffer with the bytes to write
- **btw** -- Bytes To Write
- **bw** -- the number of real written bytes (Bytes Written). NULL if unused.

Ritorna

`LV_FS_RES_OK` or any error from `lv_fs_res_t` enum

`lv_fs_res_t lv_fs_seek(lv_fs_file_t *file_p, uint32_t pos, lv_fs_whence_t whence)`

Set the position of the 'cursor' (read write pointer) in a file

Parametri

- **file_p** -- pointer to a `lv_fs_file_t` variable
- **pos** -- the new position expressed in bytes index (0: start of file)
- **whence** -- tells from where to set position. See `lv_fs_whence_t`

Ritorna

`LV_FS_RES_OK` or any error from `lv_fs_res_t` enum

`lv_fs_res_t lv_fs_tell(lv_fs_file_t *file_p, uint32_t *pos)`

Give the position of the read write pointer

Parametri

- **file_p** -- pointer to a `lv_fs_file_t` variable
- **pos** -- pointer to store the position of the read write pointer

Ritorna

`LV_FS_RES_OK` or any error from 'fs_res_t'

`lv_fs_res_t lv_fs_get_size(lv_fs_file_t *file_p, uint32_t *size_res)`

Get the size in bytes of an open file. The file read/write position will not be affected.

Parametri

- **file_p** -- pointer to a `lv_fs_file_t` variable
- **size_res** -- pointer to store the file size

Ritorna

LV_FS_RES_OK or any error from `lv_fs_res_t`

`lv_fs_res_t lv_fs_path_get_size(const char *path, uint32_t *size_res)`

Get the size in bytes of a file at the given path.

Parametri

- **path** -- the path of the file
- **size_res** -- pointer to store the file size

Ritorna

LV_FS_RES_OK or any error from `lv_fs_res_t`

`lv_fs_res_t lv_fs_load_to_buf(void *buf, uint32_t buf_size, const char *path)`

Read the contents of a file at the given path into a buffer.

Parametri

- **buf** -- a buffer to read the contents of the file into
- **buf_size** -- the size of the buffer and the amount to read from the file
- **path** -- the path of the file

Ritorna

LV_FS_RES_OK on success, LV_FS_RES_UNKNOWN if fewer than `buf_size` bytes could be read from the file, or any error from `lv_fs_res_t`

`void *lv_fs_load_with_alloc(const char *path, uint32_t *size)`

Load a file into a memory buffer.

Parametri

- **filename** -- the path of the file
- **size** -- pointer to store the size of the loaded file

Ritorna

a pointer to the loaded file buffer, or NULL if an error occurred

`lv_fs_res_t lv_fs_dir_open(lv_fs_dir_t *raddir_p, const char *path)`

Initialize a 'fs_dir_t' variable for directory reading

Parametri

- **raddir_p** -- pointer to a '`lv_fs_dir_t`' variable
- **path** -- path to a directory

Ritorna

LV_FS_RES_OK or any error from `lv_fs_res_t` enum

`lv_fs_res_t lv_fs_dir_read(lv_fs_dir_t *raddir_p, char *fn, uint32_t fn_len)`

Read the next filename form a directory. The name of the directories will begin with '/'

Parametri

- **raddir_p** -- pointer to an initialized 'fs_dir_t' variable
- **fn** -- pointer to a buffer to store the filename
- **fn_len** -- length of the buffer to store the filename

Ritorna

LV_FS_RES_OK or any error from `lv_fs_res_t` enum

`lv_fs_res_t lv_fs_dir_close(lv_fs_dir_t *raddir_p)`

Close the directory reading

Parametri

- **raddir_p** -- pointer to an initialized 'fs_dir_t' variable

Ritorna

LV_FS_RES_OK or any error from `lv_fs_res_t` enum

`char *lv_fs_get_letters(char *buf)`

Fill a buffer with the letters of existing drivers

Parametri

- **buf** -- buffer to store the letters ('\0' added after the last letter)

Ritorna

the buffer

`const char *lv_fs_get_ext(const char *fn)`

Return with the extension of the filename

Parametri

`fn` -- string with a filename

Ritorna

pointer to the beginning extension or empty string if no extension

`char *lv_fs_up(char *path)`

Step up one level

Parametri

`path` -- pointer to a file name

Ritorna

the truncated file name

`const char *lv_fs_get_last(const char *path)`

Get the last element of a path (e.g. U:/folder/file -> file)

Parametri

`path` -- pointer to a file name

Ritorna

pointer to the beginning of the last element in the path

`int lv_fs_path_join(char *buf, size_t buf_sz, const char *base, const char *end)`

Concatenate two path components and automatically add/remove a separator as needed. buf, buf_sz, and the return value are analogous to lv_snprintf

Parametri

- `buf` -- the buffer to place the result in
- `buf_sz` -- the size of buf. At most buf_sz - 1 characters will be written to buf, and a null terminator
- `base` -- the first path component
- `end` -- the second path component

Ritorna

the number of characters (not including the null terminator) that would be written to buf, even if buf_sz-1 was smaller

`struct _lv_fs_drv_t`

Membri Pubblici

`char letter`

`uint32_t cache_size`

`bool (*ready_cb)(lv_fs_drv_t *drv)`

`void *(*open_cb)(lv_fs_drv_t *drv, const char *path, lv_fs_mode_t mode)`

`lv_fs_res_t (*close_cb)(lv_fs_drv_t *drv, void *file_p)`

`lv_fs_res_t (*read_cb)(lv_fs_drv_t *drv, void *file_p, void *buf, uint32_t btr, uint32_t *br)`

`lv_fs_res_t (*write_cb)(lv_fs_drv_t *drv, void *file_p, const void *buf, uint32_t btw, uint32_t *bw)`

`lv_fs_res_t (*seek_cb)(lv_fs_drv_t *drv, void *file_p, uint32_t pos, lv_fs_whence_t whence)`

```
lv_fs_res_t (*tell_cb)(lv_fs_drv_t *drv, void *file_p, uint32_t *pos_p)

void *(*dir_open_cb)(lv_fs_drv_t *drv, const char *path)

lv_fs_res_t (*dir_read_cb)(lv_fs_drv_t *drv, void *raddir_p, char *fn, uint32_t fn_len)

lv_fs_res_t (*dir_close_cb)(lv_fs_drv_t *drv, void *raddir_p)

void *user_data
    Custom file user data

struct lv_fs_file_t
```

Membri Pubblicivoid ***file_d***lv_fs_drv_t* ***drv***lv_fs_file_cache_t* ***cache**struct **lv_fs_dir_t****Membri Pubblici**void ***dir_d***lv_fs_drv_t* ***drv**struct **lv_fs_path_ex_t**

#include <lv_fs.h> Extended path object to specify buffer for memory-mapped files

Membri Pubblicichar **path**[64]

Store the driver letter address and size

14.21.19 lv_fs_private.h**Funzioni**void **lv_fs_init**(void)

Initialize the File system interface

void **lv_fs_deinit**(void)

Deinitialize the File system interface

struct **_lv_fs_file_cache_t**

Membri Pubblici

```
uint32_t start
uint32_t end
uint32_t file_position
void *buffer
```

14.21.20 lv_grad.h**Enum****enum lv_grad_dir_t**

The direction of the gradient.

Values:

enumerator LV_GRAD_DIR_NONE

No gradient (the `grad_color` property is ignored)

enumerator LV_GRAD_DIR_VER

Simple vertical (top to bottom) gradient

enumerator LV_GRAD_DIR_HOR

Simple horizontal (left to right) gradient

enumerator LV_GRAD_DIR_LINEAR

Linear gradient defined by start and end points. Can be at any angle.

enumerator LV_GRAD_DIR_RADIAL

Radial gradient defined by start and end circles

enumerator LV_GRAD_DIR_CONICAL

Conical gradient defined by center point, start and end angles

enum lv_grad_extend_t

Gradient behavior outside the defined range.

Values:

enumerator LV_GRAD_EXTEND_PAD

Repeat the same color

enumerator LV_GRAD_EXTEND_REPEAT

Repeat the pattern

enumerator LV_GRAD_EXTEND_REFLECT

Repeat the pattern mirrored

Funzioni

`void lv_grad_init_stops(lv_grad_dsc_t *grad, const lv_color_t colors[], const lv_opa_t opa[], const uint8_t fracs[], int num_stops)`

Initialize gradient color map from a table

Parametri

- **grad** -- pointer to a gradient descriptor
- **colors** -- color array
- **fracs** -- position array (0..255): if NULL, then colors are distributed evenly
- **opa** -- opacity array: if NULL, then LV_OPA_COVER is assumed
- **num_stops** -- number of gradient stops (1..LV_GRADIENT_MAX_STOPS)

`void lv_grad_horizontal_init(lv_grad_dsc_t *dsc)`

Helper function to initialize a horizontal gradient.

Parametri

dsc -- gradient descriptor

`void lv_grad_vertical_init(lv_grad_dsc_t *dsc)`

Helper function to initialize a vertical gradient.

Parametri

dsc -- gradient descriptor

`void lv_grad_linear_init(lv_grad_dsc_t *dsc, int32_t from_x, int32_t from_y, int32_t to_x, int32_t to_y, lv_grad_extend_t extend)`

Helper function to initialize linear gradient

Parametri

- **dsc** -- gradient descriptor
- **from_x** -- start x position: can be a coordinate or an `lv_pct()` value predefined constants LV_GRAD_LEFT, LV_GRAD_RIGHT, LV_GRAD_TOP, LV_GRAD_BOTTOM, LV_GRAD_CENTER can be used as well
- **from_y** -- start y position
- **to_x** -- end x position
- **to_y** -- end y position
- **extend** -- one of LV_GRAD_EXTEND_PAD, LV_GRAD_EXTEND_REPEAT or LV_GRAD_EXTEND_REFLECT

`void lv_grad_radial_init(lv_grad_dsc_t *dsc, int32_t center_x, int32_t center_y, int32_t to_x, int32_t to_y, lv_grad_extend_t extend)`

Helper function to initialize radial gradient

Parametri

- **dsc** -- gradient descriptor
- **center_x** -- center x position: can be a coordinate or an `lv_pct()` value predefined constants LV_GRAD_LEFT, LV_GRAD_RIGHT, LV_GRAD_TOP, LV_GRAD_BOTTOM, LV_GRAD_CENTER can be used as well
- **center_y** -- center y position
- **to_x** -- point on the end circle x position
- **to_y** -- point on the end circle y position
- **extend** -- one of LV_GRAD_EXTEND_PAD, LV_GRAD_EXTEND_REPEAT or LV_GRAD_EXTEND_REFLECT

`void lv_grad_radial_set_focal(lv_grad_dsc_t *dsc, int32_t center_x, int32_t center_y, int32_t radius)`

Set focal (starting) circle of a radial gradient

Parametri

- **dsc** -- gradient descriptor
- **center_x** -- center x position: can be a coordinate or an `lv_pct()` value predefined constants LV_GRAD_LEFT, LV_GRAD_RIGHT, LV_GRAD_TOP, LV_GRAD_BOTTOM, LV_GRAD_CENTER can be used as well
- **center_y** -- center y position
- **radius** -- radius of the starting circle (NOTE: this must be a scalar number, not percentage)

```
void lv_grad_conical_init(lv_grad_dsc_t *dsc, int32_t center_x, int32_t center_y, int32_t start_angle, int32_t end_angle, lv_grad_extend_t extend)
```

Helper function to initialize conical gradient

Parametri

- **dsc** -- gradient descriptor
- **center_x** -- center x position: can be a coordinate or an [lv_pct\(\)](#) value predefined constants LV_GRAD_LEFT, LV_GRAD_RIGHT, LV_GRAD_TOP, LV_GRAD_BOTTOM, LV_GRAD_CENTER can be used as well
- **center_y** -- center y position
- **start_angle** -- start angle in degrees
- **end_angle** -- end angle in degrees
- **extend** -- one of LV_GRAD_EXTEND_PAD, LV_GRAD_EXTEND_REPEAT or LV_GRAD_EXTEND_REFLECT

```
struct lv_grad_stop_t
```

#include <lv_grad.h> A gradient stop definition. This matches a color and a position in a virtual 0-255 scale.

Membri Pubblici

lv_color_t **color**

The stop color

lv_opa_t **opa**

The opacity of the color

uint8_t **frac**

The stop position in 1/255 unit

```
struct lv_grad_dsc_t
```

#include <lv_grad.h> A descriptor of a gradient.

Membri Pubblici

lv_grad_stop_t **stops[2]**

A gradient stop array

uint8_t **stops_count**

The number of used stops in the array

lv_grad_dir_t **dir**

The gradient direction. Any of LV_GRAD_DIR_NONE, LV_GRAD_DIR_VER, LV_GRAD_DIR_HOR, LV_GRAD_TYPE_LINEAR, LV_GRAD_TYPE_RADIAL, LV_GRAD_TYPE_CONICAL

lv_grad_extend_t **extend**

Behaviour outside the defined range. LV_GRAD_EXTEND_NONE, LV_GRAD_EXTEND_PAD, LV_GRAD_EXTEND_REPEAT, LV_GRAD_EXTEND_REFLECT

lv_point_t **start**

Linear gradient vector start point

lv_point_t **end**

Linear gradient vector end point

Center of the ending circle in local coordinates

struct ***lv_grad_dsc_t*** **linear*****lv_point_t*** **focal**

Center of the focal (starting) circle in local coordinates

lv_point_t **focal_extent**

Point on the circle (can be the same as the center)

lv_point_t **end_extent**

Point on the circle determining the radius of the gradient

struct ***lv_grad_dsc_t*** **radial*****lv_point_t*** **center**

Conical gradient center point

int16_t **start_angle**

Start angle 0..3600

int16_t **end_angle**

End angle 0..3600

struct ***lv_grad_dsc_t*** **conical**union ***lv_grad_dsc_t*** **params**void ***state****14.21.21 lv_iter.h****Typedef**

```
typedef lv_result_t (*lv_iter_next_cb)(void *instance, void *context, void *elem)
```

```
typedef void (*lv_iter_inspect_cb)(void *elem)
```

Funzioni

```
lv_iter_t *lv_iter_create(void *instance, uint32_t elem_size, uint32_t context_size, lv_iter_next_cb next_cb)
```

Create an iterator based on an instance, and then the next element of the iterator can be obtained through lv_iter_next, In order to obtain the next operation in a unified and abstract way.

Parametri

- **instance** -- The instance to be iterated
- **elem_size** -- The size of the element to be iterated in bytes
- **context_size** -- The size of the context to be passed to the next_cb in bytes
- **next_cb** -- The callback function to get the next element

Ritorna

The iterator object

```
void *lv_iter_get_context(const lv_iter_t *iter)
```

Get the context of the iterator. You can use it to store some temporary variables associated with current iterator..

Parametri

- **iter** -- lv_iter_t object create before

Ritorna

the iter context

```
void lv_iter_destroy(lv_iter_t *iter)
```

Destroy the iterator object, and release the context. Other resources allocated by the user are not released. The user needs to release it by itself.

Parametri

- **iter** -- lv_iter_t object create before

```
lv_result_t lv_iter_next(lv_iter_t *iter, void *elem)
```

Get the next element of the iterator.

Parametri

- **iter** -- lv_iter_t object create before
- **elem** -- The pointer to store the next element

Ritorna

LV_RESULT_OK: Get the next element successfully
LV_RESULT_INVALID: The next element is invalid

```
void lv_iter_make_peekable(lv_iter_t *iter, uint32_t capacity)
```

Make the iterator peekable, which means that the user can peek the next element without advancing the iterator.

Parametri

- **iter** -- lv_iter_t object create before
- **capacity** -- The capacity of the peek buffer

```
lv_result_t lv_iter_peek(lv_iter_t *iter, void *elem)
```

Peek the next element of the iterator without advancing the iterator.

Parametri

- **iter** -- lv_iter_t object create before
- **elem** -- The pointer to store the next element

Ritorna

LV_RESULT_OK: Peek the next element successfully
LV_RESULT_INVALID: The next element is invalid

```
lv_result_t lv_iter_peek_advance(lv_iter_t *iter)
```

Only advance the iterator without getting the next element.

Parametri

- **iter** -- lv_iter_t object create before

Ritorna

LV_RESULT_OK: Peek the next element successfully
LV_RESULT_INVALID: The next element is invalid

```
lv_result_t lv_iter_peek_reset(lv_iter_t *iter)
```

Reset the peek cursor to the next cursor.

Parametri

- **iter** -- lv_iter_t object create before

Ritorna

LV_RESULT_OK: Reset the peek buffer successfully
LV_RESULT_INVALID: The peek buffer is invalid

```
void lv_iter_inspect(lv_iter_t *iter, lv_iter_inspect_cb inspect_cb)
```

Inspect the element of the iterator. The callback function will be called for each element of the iterator.

Parametri

- **iter** -- `lv_iter_t` object create before
- **inspect_cb** -- The callback function to inspect the element

14.21.22 lv_ll.h

Handle linked lists. The nodes are dynamically allocated by the 'lv_mem' module.

Define

`LV_LL_READ(list, i)`

`LV_LL_READ_BACK(list, i)`

Typedef

`typedef uint8_t lv_ll_node_t`

Dummy type to make handling easier

Funzioni

`void lv_ll_init(lv_ll_t *ll_p, uint32_t node_size)`

Initialize linked list

Parametri

- **ll_p** -- pointer to `lv_ll_t` variable
- **node_size** -- the size of 1 node in bytes

`void *lv_ll_ins_head(lv_ll_t *ll_p)`

Add a new head to a linked list

Parametri

`ll_p` -- pointer to linked list

Ritorna

pointer to the new head

`void *lv_ll_ins_prev(lv_ll_t *ll_p, void *n_act)`

Insert a new node in front of the n_act node

Parametri

- **ll_p** -- pointer to linked list
- **n_act** -- pointer a node

Ritorna

pointer to the new node

`void *lv_ll_ins_tail(lv_ll_t *ll_p)`

Add a new tail to a linked list

Parametri

`ll_p` -- pointer to linked list

Ritorna

pointer to the new tail

`void lv_ll_remove(lv_ll_t *ll_p, void *node_p)`

Remove the node 'node_p' from 'll_p' linked list. It does not free the memory of node.

Parametri

- **ll_p** -- pointer to the linked list of 'node_p'
- **node_p** -- pointer to node in 'll_p' linked list

`void lv_ll_clear_custom(lv_ll_t *ll_p, void (*cleanup)(void*))`

`void lv_ll_clear(lv_ll_t *ll_p)`

Remove and free all elements from a linked list. The list remain valid but become empty.

Parametri

ll_p -- pointer to linked list

void lv_ll_chg_list(*lv_ll_t* *ll_ori_p, *lv_ll_t* *ll_new_p, void *node, bool head)

Move a node to a new linked list

Parametri

- **ll_ori_p** -- pointer to the original (old) linked list
- **ll_new_p** -- pointer to the new linked list
- **node** -- pointer to a node
- **head** -- true: be the head in the new list false be the tail in the new list

void *lv_ll_get_head(const *lv_ll_t* *ll_p)

Return with head node of the linked list

Parametri

ll_p -- pointer to linked list

Ritorna

pointer to the head of 'll_p'

void *lv_ll_get_tail(const *lv_ll_t* *ll_p)

Return with tail node of the linked list

Parametri

ll_p -- pointer to linked list

Ritorna

pointer to the tail of 'll_p'

void *lv_ll_get_next(const *lv_ll_t* *ll_p, const void *n_act)

Return with the pointer of the next node after 'n_act'

Parametri

- **ll_p** -- pointer to linked list
- **n_act** -- pointer a node

Ritorna

pointer to the next node

void *lv_ll_get_prev(const *lv_ll_t* *ll_p, const void *n_act)

Return with the pointer of the previous node after 'n_act'

Parametri

- **ll_p** -- pointer to linked list
- **n_act** -- pointer a node

Ritorna

pointer to the previous node

uint32_t lv_ll_get_len(const *lv_ll_t* *ll_p)

Return the length of the linked list.

Parametri

ll_p -- pointer to linked list

Ritorna

length of the linked list

void lv_ll_move_before(*lv_ll_t* *ll_p, void *n_act, void *n_after)

Move a node before another node in the same linked list

Parametri

- **ll_p** -- pointer to a linked list
- **n_act** -- pointer to node to move
- **n_after** -- pointer to a node which should be after n_act

bool lv_ll_is_empty(*lv_ll_t* *ll_p)

Check if a linked list is empty

Parametri

ll_p -- pointer to a linked list

Ritorna

true: the linked list is empty; false: not empty

```
struct lv_ll_t
#include <lv_ll.h> Description of a linked list
```

Membri Pubblici`uint32_t n_size``lv_ll_node_t *head``lv_ll_node_t *tail`**14.21.23 lv_log.h****Define****LV_LOG_LEVEL_TRACE**

Log detailed information.

LV_LOG_LEVEL_INFO

Log important events.

LV_LOG_LEVEL_WARN

Log if something unwanted happened but didn't caused problem.

LV_LOG_LEVEL_ERROR

Log only critical issues, when system may fail.

LV_LOG_LEVEL_USER

Log only custom log messages added by the user.

LV_LOG_LEVEL_NONE

Do not log anything.

LV_LOG_LEVEL_NUM

Number of log levels

LV_LOG_FILE**LV_LOG_LINE**`LV_LOG_TRACE(...)``LV_LOG_INFO(...)``LV_LOG_WARN(...)``LV_LOG_ERROR(...)``LV_LOG_USER(...)``LV_LOG(...)`

Typedef

```
typedef int8_t lv_log_level_t
```

```
typedef void (*lv_log_print_g_cb_t)(lv_log_level_t level, const char *buf)
```

Log print function. Receives a string buffer to print".

Funzioni

```
void lv_log_register_print_cb(lv_log_print_g_cb_t print_cb)
```

Register custom print/write function to call when a log is added. It can format its "File path", "Line number" and "Description" as required and send the formatted log message to a console or serial port.

Parametri

- print_cb** -- a function pointer to print a log

```
void lv_log(const char *format, ...)
```

Print a log message via printf if enabled with LV_LOG_PRINTF in lv_conf.h and/or a print callback if registered with lv_log_register_print_cb

Parametri

- format** -- printf-like format string
- ...** -- parameters for format

```
void lv_log_add(lv_log_level_t level, const char *file, int line, const char *func, const char *format, ...)
```

Add a log

Parametri

- level** -- the level of log. (From lv_log_level_t enum)
- file** -- name of the file when the log added
- line** -- line number in the source code where the log added
- func** -- name of the function when the log added
- format** -- printf-like format string
- ...** -- parameters for format

14.21.24 lv_lru.h**Typedef**

```
typedef void (*lv_lru_free_cb_t)(void *v)
```

```
typedef struct _lv_lru_item_t lv_lru_item_t
```

```
typedef struct _lv_lru_t lv_lru_t
```

Enum

```
enum lv_lru_res_t
```

Values:

enumerator **LV_LRU_OK**

enumerator **LV_LRU_MISSING_CACHE**

enumerator **LV_LRU_MISSING_KEY**

enumerator **LV_LRU_MISSING_VALUE**

enumerator **LV_LRU_LOCK_ERROR**

enumerator **LV_LRU_VALUE_TOO_LARGE**

Funzioni

lv_lru_t ***lv_lru_create**(size_t cache_size, size_t average_length, *lv_lru_free_cb_t* value_free, *lv_lru_free_cb_t* key_free)

void **lv_lru_delete**(*lv_lru_t* *cache)

lv_lru_res_t **lv_lru_set**(*lv_lru_t* *cache, const void *key, size_t key_length, void *value, size_t value_length)

lv_lru_res_t **lv_lru_get**(*lv_lru_t* *cache, const void *key, size_t key_size, void **value)

lv_lru_res_t **lv_lru_remove**(*lv_lru_t* *cache, const void *key, size_t key_size)

void **lv_lru_remove_lru_item**(*lv_lru_t* *cache)

remove the least recently used item

Todo:

we can optimise this by finding the n lru items, where n = required_space / average_length

struct **_lv_lru_t**

Membri Pubblici

lv_lru_item_t ****items**

uint64_t **access_count**

size_t **free_memory**

size_t **total_memory**

size_t **average_item_length**

size_t **hash_table_size**

uint32_t **seed**

lv_lru_free_cb_t **value_free**

lv_lru_free_cb_t **key_free**

lv_lru_item_t ***free_items**

14.21.25 lv_math.h

Define

LV_TRIGO_SIN_MAX

LV_TRIGO_SHIFT

LV_TRIGO_SHIFT to normalize

LV_BEZIER_VAL_SHIFT

$\log_2(LV_BEZIER_VAL_MAX)$: used to normalize up scaled values

LV_BEZIER_VAL_MAX

Max time in Bezier functions (not [0..1] to use integers)

LV_BEZIER_VAL_FLOAT(f)

Convert const float number cubic-bezier values to fix-point value

LV_ALIGN_UP(x, align)

Align up value x to align, align must be a power of two

LV_ROUND_UP(x, round)

Round up value x to round, round can be any integer number

LV_MIN(a, b)

LV_MIN3(a, b, c)

LV_MIN4(a, b, c, d)

LV_MAX(a, b)

LV_MAX3(a, b, c)

LV_MAX4(a, b, c, d)

LV_CLAMP(min, val, max)

LV_ABS(x)

LV_UDIV255(x)

LV_IS_SIGNED(t)

LV_UMAX_OF(t)

LV_SMAX_OF(t)

LV_MAX_OF(t)

Funzioni

int32_t lv_trigo_sin(int16_t angle)

Return with sinus of an angle

Parametri

angle --

Ritorna

sinus of 'angle'. sin(-90) = -32767, sin(90) = 32767

int32_t lv_trigo_cos(int16_t angle)

`int32_t lv_cubic_bezier(int32_t x, int32_t x1, int32_t y1, int32_t x2, int32_t y2)`

Calculate the y value of cubic-bezier(x1, y1, x2, y2) function as specified x.

Parametri

- **x** -- time in range of [0..LV_BEZIER_VAL_MAX]
- **x1** -- x of control point 1 in range of [0..LV_BEZIER_VAL_MAX]
- **y1** -- y of control point 1 in range of [0..LV_BEZIER_VAL_MAX]
- **x2** -- x of control point 2 in range of [0..LV_BEZIER_VAL_MAX]
- **y2** -- y of control point 2 in range of [0..LV_BEZIER_VAL_MAX]

Ritorna

the value calculated

`int32_t lv_bezier3(int32_t t, int32_t u0, uint32_t u1, int32_t u2, int32_t u3)`

Calculate a value of a Cubic Bezier function.

Parametri

- **t** -- time in range of [0..LV_BEZIER_VAL_MAX]
- **u0** -- must be 0
- **u1** -- control value 1 values in range of [0..LV_BEZIER_VAL_MAX]
- **u2** -- control value 2 in range of [0..LV_BEZIER_VAL_MAX]
- **u3** -- must be LV_BEZIER_VAL_MAX

Ritorna

the value calculated from the given parameters in range of [0..LV_BEZIER_VAL_MAX]

`uint16_t lv_atan2(int x, int y)`

Calculate the atan2 of a vector.

Parametri

- **x** --
- **y** --

Ritorna

the angle in degree calculated from the given parameters in range of [0..360]

`void lv_sqrt(uint32_t x, lv_sqrt_res_t *q, uint32_t mask)`

Get the square root of a number

Parametri

- **x** -- integer which square root should be calculated
- **q** -- store the result here. q->i: integer part, q->f: fractional part in 1/256 unit
- **mask** -- optional to skip some iterations if the magnitude of the root is known. Set to 0x8000 by default. If root < 16: mask = 0x80 If root < 256: mask = 0x800 Else: mask = 0x8000

`int32_t lv_sqrt32(uint32_t x)`

Alternative (fast, approximate) implementation for getting the square root of an integer.

Parametri

x -- integer which square root should be calculated

`static inline int32_t lv_sqr(int32_t x)`

Calculate the square of an integer (input range is 0..32767).

Parametri

x -- input

Ritorna

square

`int64_t lv_pow(int64_t base, int8_t exp)`

Calculate the integer exponents.

Parametri

- **base** --
- **exp** --

Ritorna

base raised to the power exponent

```
int32_t lv_map(int32_t x, int32_t min_in, int32_t max_in, int32_t min_out, int32_t max_out)
```

Get the mapped of a number given an input and output range

Parametri

- **x** -- integer which mapped value should be calculated
- **min_in** -- min input range
- **max_in** -- max input range
- **min_out** -- max output range
- **max_out** -- max output range

Ritorna

the mapped number

```
void lv_rand_set_seed(uint32_t seed)
```

Set the seed of the pseudo random number generator

Parametri

seed -- a number to initialize the random generator

```
uint32_t lv_rand(uint32_t min, uint32_t max)
```

Get a pseudo random number in the given range

Parametri

- **min** -- the minimum value
- **max** -- the maximum value

Ritorna

return the random number. min <= return_value <= max

```
struct lv_sqrt_res_t
```

Membri Pubblici

uint16_t i

uint16_t f

14.21.26 lv_matrix.h

Funzioni

```
void lv_matrix_identity(lv_matrix_t *matrix)
```

Set matrix to identity matrix

Parametri

matrix -- pointer to a matrix

```
void lv_matrix_translate(lv_matrix_t *matrix, float tx, float ty)
```

Translate the matrix to new position

Parametri

- **matrix** -- pointer to a matrix
- **tx** -- the amount of translate in x direction
- **ty** -- the amount of translate in y direction

```
void lv_matrix_scale(lv_matrix_t *matrix, float scale_x, float scale_y)
```

Change the scale factor of the matrix

Parametri

- **matrix** -- pointer to a matrix
- **scale_x** -- the scale factor for the X direction
- **scale_y** -- the scale factor for the Y direction

```
void lv_matrix_rotate(lv_matrix_t *matrix, float degree)
```

Rotate the matrix with origin

Parametri

- **matrix** -- pointer to a matrix
- **degree** -- angle to rotate

void lv_matrix_skew(lv_matrix_t *matrix, float skew_x, float skew_y)

Change the skew factor of the matrix

Parametri

- **matrix** -- pointer to a matrix
- **skew_x** -- the skew factor for x direction
- **skew_y** -- the skew factor for y direction

void lv_matrix_multiply(lv_matrix_t *matrix, const lv_matrix_t *mul)

Multiply two matrix and store the result to the first one

Parametri

- **matrix** -- pointer to a matrix
- **matrix2** -- pointer to another matrix

bool lv_matrix_inverse(lv_matrix_t *matrix, const lv_matrix_t *m)

Invert the matrix

Parametri

- **matrix** -- pointer to a matrix
- **m** -- pointer to another matrix (optional)

Ritorna

true: the matrix is invertible, false: the matrix is singular and cannot be inverted

lv_point_precise_t lv_matrix_transform_precise_point(const lv_matrix_t *matrix, const lv_point_precise_t *point)

Transform a point by a matrix

Parametri

- **matrix** -- pointer to a matrix
- **point** -- pointer to a point

Ritorna

the transformed point

lv_area_t lv_matrix_transform_area(const lv_matrix_t *matrix, const lv_area_t *area)

Transform an area by a matrix

Parametri

- **matrix** -- pointer to a matrix
- **area** -- pointer to an area

Ritorna

the transformed area

bool lv_matrix_is_identity(const lv_matrix_t *matrix)

Check if the matrix is identity

Parametri

matrix -- pointer to a matrix

Ritorna

true: the matrix is identity , false: the matrix is not identity

bool lv_matrix_is_identity_or_translation(const lv_matrix_t *matrix)

Check if the matrix is identity or translation matrix

Parametri

matrix -- pointer to a matrix

Ritorna

true: the matrix is identity or translation matrix, false: the matrix is not identity or translation matrix

struct _lv_matrix_t

Membri Pubblici

float m[3][3]

14.21.27 lv_palette.h

Enum

enum **lv_palette_t**

Values:

enumerator **LV_PALETTE_RED**

enumerator **LV_PALETTE_PINK**

enumerator **LV_PALETTE_PURPLE**

enumerator **LV_PALETTE_DEEP_PURPLE**

enumerator **LV_PALETTE_INDIGO**

enumerator **LV_PALETTE_BLUE**

enumerator **LV_PALETTE_LIGHT_BLUE**

enumerator **LV_PALETTE_CYAN**

enumerator **LV_PALETTE_TEAL**

enumerator **LV_PALETTE_GREEN**

enumerator **LV_PALETTE_LIGHT_GREEN**

enumerator **LV_PALETTE_LIME**

enumerator **LV_PALETTE_YELLOW**

enumerator **LV_PALETTE_AMBER**

enumerator **LV_PALETTE_ORANGE**

enumerator **LV_PALETTE_DEEP_ORANGE**

enumerator **LV_PALETTE_BROWN**

enumerator **LV_PALETTE_BLUE_GREY**

enumerator `LV_PALETTE_GREY`

enumerator `LV_PALETTE_LAST`

enumerator `LV_PALETTE_NONE`

Funzioni

`lv_color_t lv_palette_main(lv_palette_t p)`

`lv_color_t lv_palette_lighten(lv_palette_t p, uint8_t lvl)`

`lv_color_t lv_palette_darken(lv_palette_t p, uint8_t lvl)`

14.21.28 lv_profiler.h

Define

`LV_PROFILER_BEGIN`

`LV_PROFILER_END`

`LV_PROFILER_BEGIN_TAG(tag)`

`LV_PROFILER_END_TAG(tag)`

`LV_PROFILER_LAYOUT_BEGIN`

`LV_PROFILER_LAYOUT_END`

`LV_PROFILER_LAYOUT_BEGIN_TAG(tag)`

`LV_PROFILER_LAYOUT_END_TAG(tag)`

`LV_PROFILER_STYLE_BEGIN`

`LV_PROFILER_STYLE_END`

`LV_PROFILER_STYLE_BEGIN_TAG(tag)`

`LV_PROFILER_STYLE_END_TAG(tag)`

`LV_PROFILER_DRAW_BEGIN`

`LV_PROFILER_DRAW_END`

`LV_PROFILER_DRAW_BEGIN_TAG(tag)`

`LV_PROFILER_DRAW_END_TAG(tag)`

`LV_PROFILER_DECODER_BEGIN`

`LV_PROFILER_DECODER_END`

LV_PROFILER_DECODER_BEGIN_TAG(tag)

LV_PROFILER_DECODER_END_TAG(tag)

LV_PROFILER_REFR_BEGIN

LV_PROFILER_REFR_END

LV_PROFILER_REFR_BEGIN_TAG(tag)

LV_PROFILER_REFR_END_TAG(tag)

LV_PROFILER_INDEV_BEGIN

LV_PROFILER_INDEV_END

LV_PROFILER_INDEV_BEGIN_TAG(tag)

LV_PROFILER_INDEV_END_TAG(tag)

LV_PROFILER_FONT_BEGIN

LV_PROFILER_FONT_END

LV_PROFILER_FONT_BEGIN_TAG(tag)

LV_PROFILER_FONT_END_TAG(tag)

LV_PROFILER_CACHE_BEGIN

LV_PROFILER_CACHE_END

LV_PROFILER_CACHE_BEGIN_TAG(tag)

LV_PROFILER_CACHE_END_TAG(tag)

LV_PROFILER_FS_BEGIN

LV_PROFILER_FS_END

LV_PROFILER_FS_BEGIN_TAG(tag)

LV_PROFILER_FS_END_TAG(tag)

LV_PROFILER_TIMER_BEGIN

LV_PROFILER_TIMER_END

LV_PROFILER_TIMER_BEGIN_TAG(tag)

LV_PROFILER_TIMER_END_TAG(tag)

LV_PROFILER_EVENT_BEGIN

`LV_PROFILER_EVENT_END`

`LV_PROFILER_EVENT_BEGIN_TAG(tag)`

`LV_PROFILER_EVENT_END_TAG(tag)`

14.21.29 lv_profiler_builtin.h

14.21.30 lv_profiler_builtin_private.h

14.21.31 lv_rb.h

Typedef

`typedef int8_t lv_rb_compare_res_t`

`typedef lv_rb_compare_res_t (*lv_rb_compare_t)(const void *a, const void *b)`

Enum

`enum lv_rb_color_t`

Values:

enumerator `LV_RB_COLOR_RED`

enumerator `LV_RB_COLOR_BLACK`

Funzioni

`bool lv_rb_init(lv_rb_t *tree, lv_rb_compare_t compare, size_t node_size)`

`lv_rb_node_t *lv_rb_insert(lv_rb_t *tree, void *key)`

`lv_rb_node_t *lv_rb_find(lv_rb_t *tree, const void *key)`

`void *lv_rb_remove_node(lv_rb_t *tree, lv_rb_node_t *node)`

`void *lv_rb_remove(lv_rb_t *tree, const void *key)`

`bool lv_rb_drop_node(lv_rb_t *tree, lv_rb_node_t *node)`

`bool lv_rb_drop(lv_rb_t *tree, const void *key)`

`lv_rb_node_t *lv_rb_minimum(lv_rb_t *node)`

`lv_rb_node_t *lv_rb_maximum(lv_rb_t *node)`

`lv_rb_node_t *lv_rb_minimum_from(lv_rb_node_t *node)`

`lv_rb_node_t *lv_rb_maximum_from(lv_rb_node_t *node)`

`void lv_rb_destroy(lv_rb_t *tree)`

14.21.32 lv_rb_private.h

`struct _lv_rb_node_t`

Membri Pubblici

```
struct _lv_rb_node_t *parent

struct _lv_rb_node_t *left

struct _lv_rb_node_t *right

lv_rb_color_t color

void *data

struct _lv_rb_t
```

Membri Pubblici

```
lv_rb_node_t *root

lv_rb_compare_t compare

size_t size
```

14.21.33 lv_style.h**Define**

LV_STYLE_SENTINEL_VALUE

LV_STYLE_PROP_FLAG_NONE

No special behavior

LV_STYLE_PROP_FLAG_INHERITABLE

Inherited

LV_STYLE_PROP_FLAG_EXT_DRAW_UPDATE

Requires ext. draw size update when changed

LV_STYLE_PROP_FLAG_LAYOUT_UPDATE

Requires layout update when changed

LV_STYLE_PROP_FLAG_PARENT_LAYOUT_UPDATE

Requires layout update on parent when changed

LV_STYLE_PROP_FLAG_LAYER_UPDATE

Affects layer handling

LV_STYLE_PROP_FLAG_TRANSFORM

Affects the object's transformation

LV_STYLE_PROP_FLAG_ALL

Indicating all flags

LV_SCALE_NONE

Value for not zooming the image

LV_STYLE_CONST_INIT(var_name, prop_array)**LV_STYLE_CONST_PROPS_END****LV_GRAD_LEFT****LV_GRAD_RIGHT****LV_GRAD_TOP****LV_GRAD_BOTTOM****LV_GRAD_CENTER****LV_ASSERT_STYLE(style_p)****Enum****enum lv_blend_mode_t**

Possible options for blending opaque drawings

Values:

enumerator **LV_BLEND_MODE_NORMAL**

Simply mix according to the opacity value

enumerator **LV_BLEND_MODE_ADDITIVE**

Add the respective color channels

enumerator **LV_BLEND_MODE_SUBTRACTIVE**

Subtract the foreground from the background

enumerator **LV_BLEND_MODE_MULTIPLY**

Multiply the foreground and background

enumerator **LV_BLEND_MODE_DIFFERENCE**

Absolute difference between foreground and background

enum lv_text_decor_t

Some options to apply decorations on texts. 'OR'ed values can be used.

Values:

enumerator **LV_TEXT_DECOR_NONE**

enumerator **LV_TEXT_DECOR_UNDERLINE**

enumerator **LV_TEXT_DECOR_STRIKETHROUGH**

enum **lv_border_side_t**

Selects on which sides border should be drawn 'OR'ed values can be used.

Values:

enumerator **LV_BORDER_SIDE_NONE**

enumerator **LV_BORDER_SIDE_BOTTOM**

enumerator **LV_BORDER_SIDE_TOP**

enumerator **LV_BORDER_SIDE_LEFT**

enumerator **LV_BORDER_SIDE_RIGHT**

enumerator **LV_BORDER_SIDE_FULL**

enumerator **LV_BORDER_SIDE_INTERNAL**

FOR matrix-like objects (e.g. Button matrix)

enum **lv_style_id_t**

Enumeration of all built in style properties

Props are split into groups of 16. When adding a new prop to a group, ensure it does not overflow into the next one.

Values:

enumerator **LV_STYLE_PROP_INV**

enumerator **LV_STYLE_WIDTH**

enumerator **LV_STYLE_HEIGHT**

enumerator **LV_STYLE_LENGTH**

enumerator **LV_STYLE_MIN_WIDTH**

enumerator **LV_STYLE_MAX_WIDTH**

enumerator **LV_STYLE_MIN_HEIGHT**

enumerator **LV_STYLE_MAX_HEIGHT**

enumerator **LV_STYLE_X**

enumerator **LV_STYLE_Y**

enumerator **LV_STYLE_ALIGN**

enumerator **LV_STYLE_RADIUS**

enumerator **LV_STYLE_RADIAL_OFFSET**

enumerator **LV_STYLE_PAD_RADIAL**

enumerator **LV_STYLE_PAD_TOP**

enumerator **LV_STYLE_PAD_BOTTOM**

enumerator **LV_STYLE_PAD_LEFT**

enumerator **LV_STYLE_PAD_RIGHT**

enumerator **LV_STYLE_PAD_ROW**

enumerator **LV_STYLE_PAD_COLUMN**

enumerator **LV_STYLE_LAYOUT**

enumerator **LV_STYLE_MARGIN_TOP**

enumerator **LV_STYLE_MARGIN_BOTTOM**

enumerator **LV_STYLE_MARGIN_LEFT**

enumerator **LV_STYLE_MARGIN_RIGHT**

enumerator **LV_STYLE_BG_COLOR**

enumerator **LV_STYLE_BG_OPA**

enumerator **LV_STYLE_BG_GRAD_DIR**

enumerator **LV_STYLE_BG_MAIN_STOP**

enumerator **LV_STYLE_BG_GRAD_STOP**

enumerator **LV_STYLE_BG_GRAD_COLOR**

enumerator **LV_STYLE_BG_MAIN_OPA**

enumerator **LV_STYLE_BG_GRAD_OPA**

enumerator **LV_STYLE_BG_GRAD**

enumerator **LV_STYLE_BASE_DIR**

enumerator **LV_STYLE_BG_IMAGE_SRC**

enumerator **LV_STYLE_BG_IMAGE_OPA**

enumerator **LV_STYLE_BG_IMAGE_RECOLOR**

enumerator **LV_STYLE_BG_IMAGE_RECOLOR_OPA**

enumerator **LV_STYLE_BG_IMAGE_TILED**

enumerator **LV_STYLE_CLIP_CORNER**

enumerator **LV_STYLE_BORDER_WIDTH**

enumerator **LV_STYLE_BORDER_COLOR**

enumerator **LV_STYLE_BORDER_OPA**

enumerator **LV_STYLE_BORDER_SIDE**

enumerator **LV_STYLE_BORDER_POST**

enumerator **LV_STYLE_OUTLINE_WIDTH**

enumerator **LV_STYLE_OUTLINE_COLOR**

enumerator **LV_STYLE_OUTLINE_OPA**

enumerator **LV_STYLE_OUTLINE_PAD**

enumerator **LV_STYLE_SHADOW_WIDTH**

enumerator **LV_STYLE_SHADOW_COLOR**

enumerator **LV_STYLE_SHADOW_OPA**

enumerator **LV_STYLE_SHADOW_OFFSET_X**

enumerator **LV_STYLE_SHADOW_OFFSET_Y**

enumerator **LV_STYLE_SHADOW_SPREAD**

enumerator **LV_STYLE_IMAGE_OPA**

enumerator **LV_STYLE_IMAGE_RECOLOR**

enumerator **LV_STYLE_IMAGE_RECOLOR_OPA**

enumerator **LV_STYLE_LINE_WIDTH**

enumerator **LV_STYLE_LINE_DASH_WIDTH**

enumerator **LV_STYLE_LINE_DASH_GAP**

enumerator **LV_STYLE_LINE_ROUNDED**

enumerator **LV_STYLE_LINE_COLOR**

enumerator **LV_STYLE_LINE_OPA**

enumerator **LV_STYLE_ARC_WIDTH**

enumerator **LV_STYLE_ARC_ROUNDED**

enumerator **LV_STYLE_ARC_COLOR**

enumerator **LV_STYLE_ARC_OPA**

enumerator **LV_STYLE_ARC_IMAGE_SRC**

enumerator **LV_STYLE_TEXT_COLOR**

enumerator **LV_STYLE_TEXT_OPA**

enumerator **LV_STYLE_TEXT_FONT**

enumerator **LV_STYLE_TEXT_LETTER_SPACE**

enumerator **LV_STYLE_TEXT_LINE_SPACE**

enumerator **LV_STYLE_TEXT_DECOR**

enumerator **LV_STYLE_TEXT_ALIGN**

enumerator **LV_STYLE_TEXT_OUTLINE_STROKE_WIDTH**

enumerator **LV_STYLE_TEXT_OUTLINE_STROKE_OPA**

enumerator **LV_STYLE_TEXT_OUTLINE_STROKE_COLOR**

enumerator **LV_STYLE_OPA**

enumerator **LV_STYLE_OPA_LAYERED**

enumerator **LV_STYLE_COLOR_FILTER_DSC**

enumerator **LV_STYLE_COLOR_FILTER_OPA**

enumerator **LV_STYLE_ANIM**

enumerator **LV_STYLE_ANIM_DURATION**

enumerator **LV_STYLE_TRANSITION**

enumerator **LV_STYLE_BLEND_MODE**

enumerator **LV_STYLE_TRANSFORM_WIDTH**

enumerator **LV_STYLE_TRANSFORM_HEIGHT**

enumerator **LV_STYLE_TRANSLATE_X**

enumerator **LV_STYLE_TRANSLATE_Y**

enumerator **LV_STYLE_TRANSFORM_SCALE_X**

enumerator **LV_STYLE_TRANSFORM_SCALE_Y**

enumerator **LV_STYLE_TRANSFORM_ROTATION**

enumerator **LV_STYLE_TRANSFORM_PIVOT_X**

enumerator **LV_STYLE_TRANSFORM_PIVOT_Y**

enumerator **LV_STYLE_TRANSFORM_SKW_X**

enumerator **LV_STYLE_TRANSFORM_SKW_Y**

enumerator **LV_STYLE_BITMAP_MASK_SRC**

enumerator **LV_STYLE_ROTARY_SENSITIVITY**

enumerator **LV_STYLE_TRANSLATE_RADIAL**

enumerator **LV_STYLE_RECOLOR**

enumerator **LV_STYLE_RECOLOR_OPA**

enumerator **LV_STYLE_FLEX_FLOW**

enumerator **LV_STYLE_FLEX_MAIN_PLACE**

enumerator **LV_STYLE_FLEX_CROSS_PLACE**

enumerator **LV_STYLE_FLEX_TRACK_PLACE**

enumerator **LV_STYLE_FLEX_GROW**

enumerator **LV_STYLE_GRID_COLUMN_ALIGN**

enumerator **LV_STYLE_GRID_ROW_ALIGN**

enumerator **LV_STYLE_GRID_ROW_DSC_ARRAY**

enumerator **LV_STYLE_GRID_COLUMN_DSC_ARRAY**

enumerator **LV_STYLE_GRID_CELL_COLUMN_POS**

enumerator **LV_STYLE_GRID_CELL_COLUMN_SPAN**

enumerator **LV_STYLE_GRID_CELL_X_ALIGN**

enumerator **LV_STYLE_GRID_CELL_ROW_POS**

enumerator **LV_STYLE_GRID_CELL_ROW_SPAN**

enumerator **LV_STYLE_GRID_CELL_Y_ALIGN**

enumerator **LV_STYLE_IMAGE_COLORKEY**

enumerator **LV_STYLE_LAST_BUILT_IN_PROP**

enumerator **LV_STYLE_NUM_BUILT_IN_PROPS**

enumerator **LV_STYLE_PROP_ANY**

enumerator **LV_STYLE_PROP_CONST**

enum **lv_style_res_t**

Values:

enumerator **LV_STYLE_RES_NOT_FOUND**

enumerator **LV_STYLE_RES_FOUND**

Funzioni

void **lv_style_init**(*lv_style_t* *style)

Initialize a style

Nota

Do not call `lv_style_init` on styles that already have some properties because this function won't free the used memory, just sets a default state for the style. In other words be sure to initialize styles only once!

Parametri

style -- pointer to a style to initialize

void **lv_style_reset**(*lv_style_t* *style)

Clear all properties from a style and free all allocated memories.

Parametri

style -- pointer to a style

void **lv_style_copy**(*lv_style_t* *dst, const *lv_style_t* *src)

Copy all properties of a style to an other. It has the same affect calling the same `lv_set_style_...` functions on both styles. It means new memory will be allocated to store the properties in the destination style. After the copy the destination style is fully independent of the source and source can removed without affecting the destination style.

Parametri

- **dst** -- the destination to copy into (can not be a constant style)
- **src** -- the source style to copy from.

void **lv_style_merge**(*lv_style_t* *dst, const *lv_style_t* *src)

Copy all properties of a style to an other without resetting the dst style. It has the same effect as calling the same `lv_set_style_...` functions on both styles. It means new memory will be allocated to store the properties in the destination style. After the copy the destination style is fully independent of the source and source can removed without affecting the destination style.

Parametri

- **dst** -- the destination to copy into (cannot be a constant style)
- **src** -- the source style to copy from.

static inline bool **lv_style_is_const**(const *lv_style_t* *style)

Check if a style is constant

Parametri

style -- pointer to a style

Ritorna

true: the style is constant

lv_style_prop_t **lv_style_register_prop**(uint8_t flag)

Register a new style property for custom usage

Example:

```

lv_style_prop_t MY_PROP;
static inline void lv_style_set_my_prop(lv_style_t * style, lv_color_t value) {
    lv_style_value_t v = {.color = value}; lv_style_set_prop(style, MY_PROP, v); }

...
MY_PROP = lv_style_register_prop();
...
lv_style_set_my_prop(&style1, lv_palette_main(LV_PALETTE_RED));

```

Ritorna

a new property ID, or LV_STYLE_PROP_INV if there are no more available.

lv_style_prop_t lv_style_get_num_custom_props(void)

Get the number of custom properties that have been registered thus far.

bool lv_style_remove_prop(lv_style_t *style, lv_style_prop_t prop)

Remove a property from a style

Parametri

- **style** -- pointer to a style
- **prop** -- a style property ORed with a state.

Ritorna

true: the property was found and removed; false: the property wasn't found

void lv_style_set_prop(lv_style_t *style, lv_style_prop_t prop, lv_style_value_t value)

Set the value of property in a style. This function shouldn't be used directly by the user. Instead use `lv_style_set_<prop_name>()`. E.g. `lv_style_set_bg_color()`

Parametri

- **style** -- pointer to style
- **prop** -- the ID of a property (e.g. LV_STYLE_BG_COLOR)
- **value** -- `lv_style_value_t` variable in which a field is set according to the type of prop

lv_style_res_t lv_style_get_prop(const lv_style_t *style, lv_style_prop_t prop, lv_style_value_t *value)

Get the value of a property

 **Nota**

For performance reasons there are no sanity check on `style`

Parametri

- **style** -- pointer to a style
- **prop** -- the ID of a property
- **value** -- pointer to a `lv_style_value_t` variable to store the value

Ritorna

LV_RESULT_INVALID: the property wasn't found in the style (value is unchanged)

LV_RESULT_OK: the property was fond, and value is set accordingly

void lv_style_transition_dsc_init(lv_style_transition_dsc_t *tr, const lv_style_prop_t props[], lv_anim_path_cb_t path_cb, uint32_t time, uint32_t delay, void *user_data)

Initialize a transition descriptor.

Example:

```

const static lv_style_prop_t trans_props[] = { LV_STYLE_BG_OPA, LV_STYLE_BG_COLOR, 0 };
static lv_style_transition_dsc_t transl;
lv_style_transition_dsc_init(&transl, trans_props, NULL, 300, 0, NULL);

```

Parametri

- **tr** -- pointer to a transition descriptor to initialize
- **props** -- an array with the properties to transition. The last element must be zero.
- **path_cb** -- an animation path (ease) callback. If NULL liner path will be used.
- **time** -- duration of the transition in [ms]
- **delay** -- delay before the transition in [ms]

- **user_data** -- any custom data that will be saved in the transition animation and will be available when path_cb is called

`lv_style_value_t lv_style_prop_get_default(lv_style_prop_t prop)`

Get the default value of a property

Parametri

prop -- the ID of a property

Ritorna

the default value

```
static inline lv_style_res_t lv_style_get_prop_inlined(const lv_style_t *style, lv_style_prop_t prop,
                                                      lv_style_value_t *value)
```

Get the value of a property

Nota

For performance reasons there are no sanity check on style

Nota

This function is the same as `lv_style_get_prop` but inlined. Use it only on performance critical places

Parametri

- **style** -- pointer to a style
- **prop** -- the ID of a property
- **value** -- pointer to a `lv_style_value_t` variable to store the value

Ritorna

LV_RESULT_INVALID: the property wasn't found in the style (value is unchanged)

LV_RESULT_OK: the property was fond, and value is set accordingly

`bool lv_style_is_empty(const lv_style_t *style)`

Checks if a style is empty (has no properties)

Parametri

style -- pointer to a style

Ritorna

true if the style is empty

`static inline uint32_t lv_style_get_prop_group(lv_style_prop_t prop)`

Tell the group of a property. If the a property from a group is set in a style the $(1 \ll \text{group})$ bit of style->has_group is set. It allows early skipping the style if the property is not exists in the style at all.

Parametri

prop -- a style property

Ritorna

the group [0..30] 30 means all the custom properties with index > 120

`uint8_t lv_style_prop_lookup_flags(lv_style_prop_t prop)`

Get the flags of a built-in or custom property.

Parametri

prop -- a style property

Ritorna

the flags of the property

`static inline void lv_style_set_size(lv_style_t *style, int32_t width, int32_t height)`

`static inline void lv_style_set_pad_all(lv_style_t *style, int32_t value)`

`static inline void lv_style_set_pad_hor(lv_style_t *style, int32_t value)`

```
static inline void lv_style_set_pad_ver(lv_style_t *style, int32_t value)
static inline void lv_style_set_pad_gap(lv_style_t *style, int32_t value)
static inline void lv_style_set_margin_hor(lv_style_t *style, int32_t value)
static inline void lv_style_set_margin_ver(lv_style_t *style, int32_t value)
static inline void lv_style_set_margin_all(lv_style_t *style, int32_t value)
static inline void lv_style_set_transform_scale(lv_style_t *style, int32_t value)

static inline bool lv_style_prop_has_flag(lv_style_prop_t prop, uint8_t flag)
```

Check if the style property has a specified behavioral flag.

Do not pass multiple flags to this function as backwards-compatibility is not guaranteed for that.

Parametri

- **prop** -- Property ID
- **flag** -- Flag

Ritorna

true if the flag is set for this property

Variabili

```
const lv_style_prop_t lv_style_const_prop_id_inv
```

```
struct lv_image_colorkey_t
```

#include <lv_style.h> A image colorkey definition. The transparency within the color range of [low, high] will be set to LV_OPA_TRANSP If the "enable" flag is set to true.

Membri Pubblici

lv_color_t **low**

lv_color_t **high**

```
union lv_style_value_t
```

#include <lv_style.h> A common type to handle all the property types in the same way.

Membri Pubblici

int32_t num

Number integer number (opacity, enums, booleans or "normal" numbers)

const void *ptr

Constant pointers (font, cone text, etc)

lv_color_t **color**

Colors

```
struct lv_style_transition_dsc_t
```

#include <lv_style.h> Descriptor for style transitions

Membri Pubblici

```
const lv_style_prop_t *props
```

An array with the properties to animate.

```
void *user_data
```

A custom user data that will be passed to the animation's user_data

```
lv_anim_path_cb_t path_xcb
```

A path for the animation.

```
uint32_t time
```

Duration of the transition in [ms]

```
uint32_t delay
```

Delay before the transition in [ms]

```
struct lv_style_const_prop_t
```

#include <lv_style.h> Descriptor of a constant style property.

Membri Pubblici

```
lv_style_prop_t prop
```

```
lv_style_value_t value
```

```
struct lv_style_t
```

#include <lv_style.h> Descriptor of a style (a collection of properties and values).

Membri Pubblici

```
uint32_t sentinel
```

```
void *values_and_props
```

```
uint32_t has_group
```

```
uint8_t prop_cnt
```

255 means it's a constant style

14.21.34 lv_style_gen.h**Define**

```
LV_STYLE_CONST_WIDTH(val)
```

```
LV_STYLE_CONST_MIN_WIDTH(val)
```

```
LV_STYLE_CONST_MAX_WIDTH(val)
```

```
LV_STYLE_CONST_HEIGHT(val)
```

`LV_STYLE_CONST_MIN_HEIGHT(val)`
`LV_STYLE_CONST_MAX_HEIGHT(val)`
`LV_STYLE_CONST_LENGTH(val)`
`LV_STYLE_CONST_X(val)`
`LV_STYLE_CONST_Y(val)`
`LV_STYLE_CONST_ALIGN(val)`
`LV_STYLE_CONST_TRANSFORM_WIDTH(val)`
`LV_STYLE_CONST_TRANSFORM_HEIGHT(val)`
`LV_STYLE_CONST_TRANSLATE_X(val)`
`LV_STYLE_CONST_TRANSLATE_Y(val)`
`LV_STYLE_CONST_TRANSLATE_RADIAL(val)`
`LV_STYLE_CONST_TRANSFORM_SCALE_X(val)`
`LV_STYLE_CONST_TRANSFORM_SCALE_Y(val)`
`LV_STYLE_CONST_TRANSFORM_ROTATION(val)`
`LV_STYLE_CONST_TRANSFORM_PIVOT_X(val)`
`LV_STYLE_CONST_TRANSFORM_PIVOT_Y(val)`
`LV_STYLE_CONST_TRANSFORM_SKW_X(val)`
`LV_STYLE_CONST_TRANSFORM_SKW_Y(val)`
`LV_STYLE_CONST_PAD_TOP(val)`
`LV_STYLE_CONST_PAD_BOTTOM(val)`
`LV_STYLE_CONST_PAD_LEFT(val)`
`LV_STYLE_CONST_PAD_RIGHT(val)`
`LV_STYLE_CONST_PAD_ROW(val)`
`LV_STYLE_CONST_PAD_COLUMN(val)`
`LV_STYLE_CONST_PAD_RADIAL(val)`
`LV_STYLE_CONST_MARGIN_TOP(val)`
`LV_STYLE_CONST_MARGIN_BOTTOM(val)`
`LV_STYLE_CONST_MARGIN_LEFT(val)`
`LV_STYLE_CONST_MARGIN_RIGHT(val)`
`LV_STYLE_CONST_BG_COLOR(val)`
`LV_STYLE_CONST_BG_OPA(val)`
`LV_STYLE_CONST_BG_GRAD_COLOR(val)`
`LV_STYLE_CONST_BG_GRAD_DIR(val)`

`LV_STYLE_CONST_BG_MAIN_STOP(val)`
`LV_STYLE_CONST_BG_GRAD_STOP(val)`
`LV_STYLE_CONST_BG_MAIN_OPA(val)`
`LV_STYLE_CONST_BG_GRAD_OPA(val)`
`LV_STYLE_CONST_BG_GRAD(val)`
`LV_STYLE_CONST_BG_IMAGE_SRC(val)`
`LV_STYLE_CONST_BG_IMAGE_OPA(val)`
`LV_STYLE_CONST_BG_IMAGE_RECOLOR(val)`
`LV_STYLE_CONST_BG_IMAGE_RECOLOR_OPA(val)`
`LV_STYLE_CONST_BG_IMAGE_TILED(val)`
`LV_STYLE_CONST_BORDER_COLOR(val)`
`LV_STYLE_CONST_BORDER_OPA(val)`
`LV_STYLE_CONST_BORDER_WIDTH(val)`
`LV_STYLE_CONST_BORDER_SIDE(val)`
`LV_STYLE_CONST_BORDER_POST(val)`
`LV_STYLE_CONST_OUTLINE_WIDTH(val)`
`LV_STYLE_CONST_OUTLINE_COLOR(val)`
`LV_STYLE_CONST_OUTLINE_OPA(val)`
`LV_STYLE_CONST_OUTLINE_PAD(val)`
`LV_STYLE_CONST_SHADOW_WIDTH(val)`
`LV_STYLE_CONST_SHADOW_OFFSET_X(val)`
`LV_STYLE_CONST_SHADOW_OFFSET_Y(val)`
`LV_STYLE_CONST_SHADOW_SPREAD(val)`
`LV_STYLE_CONST_SHADOW_COLOR(val)`
`LV_STYLE_CONST_SHADOW_OPA(val)`
`LV_STYLE_CONST_IMAGE_OPA(val)`
`LV_STYLE_CONST_IMAGE_RECOLOR(val)`
`LV_STYLE_CONST_IMAGE_RECOLOR_OPA(val)`
`LV_STYLE_CONST_IMAGE_COLORKEY(val)`
`LV_STYLE_CONST_LINE_WIDTH(val)`
`LV_STYLE_CONST_LINE_DASH_WIDTH(val)`
`LV_STYLE_CONST_LINE_DASH_GAP(val)`
`LV_STYLE_CONST_LINE_ROUNDED(val)`

LV_STYLE_CONST_LINE_COLOR(val)
LV_STYLE_CONST_LINE_OPA(val)
LV_STYLE_CONST_ARC_WIDTH(val)
LV_STYLE_CONST_ARC_ROUNDED(val)
LV_STYLE_CONST_ARC_COLOR(val)
LV_STYLE_CONST_ARC_OPA(val)
LV_STYLE_CONST_ARC_IMAGE_SRC(val)
LV_STYLE_CONST_TEXT_COLOR(val)
LV_STYLE_CONST_TEXT_OPA(val)
LV_STYLE_CONST_TEXT_FONT(val)
LV_STYLE_CONST_TEXT_LETTER_SPACE(val)
LV_STYLE_CONST_TEXT_LINE_SPACE(val)
LV_STYLE_CONST_TEXT_DECOR(val)
LV_STYLE_CONST_TEXT_ALIGN(val)
LV_STYLE_CONST_TEXT_OUTLINE_STROKE_COLOR(val)
LV_STYLE_CONST_TEXT_OUTLINE_STROKE_WIDTH(val)
LV_STYLE_CONST_TEXT_OUTLINE_STROKE_OPA(val)
LV_STYLE_CONST_RADIUS(val)
LV_STYLE_CONST_RADIAL_OFFSET(val)
LV_STYLE_CONST_CLIP_CORNER(val)
LV_STYLE_CONST_OPA(val)
LV_STYLE_CONST_OPA_LAYERED(val)
LV_STYLE_CONST_COLOR_FILTER_DSC(val)
LV_STYLE_CONST_COLOR_FILTER_OPA(val)
LV_STYLE_CONST_RECOLOR(val)
LV_STYLE_CONST_RECOLOR_OPA(val)
LV_STYLE_CONST_ANIM(val)
LV_STYLE_CONST_ANIM_DURATION(val)
LV_STYLE_CONST_TRANSITION(val)
LV_STYLE_CONST_BLEND_MODE(val)
LV_STYLE_CONST_LAYOUT(val)
LV_STYLE_CONST_BASE_DIR(val)
LV_STYLE_CONST_BITMAP_MASK_SRC(val)
LV_STYLE_CONST_ROTARY_SENSITIVITY(val)

Funzioni

```
void lv_style_set_width(lv_style_t *style, int32_t value)
void lv_style_set_min_width(lv_style_t *style, int32_t value)
void lv_style_set_max_width(lv_style_t *style, int32_t value)
void lv_style_set_height(lv_style_t *style, int32_t value)
void lv_style_set_min_height(lv_style_t *style, int32_t value)
void lv_style_set_max_height(lv_style_t *style, int32_t value)
void lv_style_set_length(lv_style_t *style, int32_t value)
void lv_style_set_x(lv_style_t *style, int32_t value)
void lv_style_set_y(lv_style_t *style, int32_t value)
void lv_style_set_align(lv_style_t *style, lv_align_t value)
void lv_style_set_transform_width(lv_style_t *style, int32_t value)
void lv_style_set_transform_height(lv_style_t *style, int32_t value)
void lv_style_set_translate_x(lv_style_t *style, int32_t value)
void lv_style_set_translate_y(lv_style_t *style, int32_t value)
void lv_style_set_translate_radial(lv_style_t *style, int32_t value)
void lv_style_set_transform_scale_x(lv_style_t *style, int32_t value)
void lv_style_set_transform_scale_y(lv_style_t *style, int32_t value)
void lv_style_set_transform_rotation(lv_style_t *style, int32_t value)
void lv_style_set_transform_pivot_x(lv_style_t *style, int32_t value)
void lv_style_set_transform_pivot_y(lv_style_t *style, int32_t value)
void lv_style_set_transform_skew_x(lv_style_t *style, int32_t value)
void lv_style_set_transform_skew_y(lv_style_t *style, int32_t value)
void lv_style_set_pad_top(lv_style_t *style, int32_t value)
void lv_style_set_pad_bottom(lv_style_t *style, int32_t value)
void lv_style_set_pad_left(lv_style_t *style, int32_t value)
void lv_style_set_pad_right(lv_style_t *style, int32_t value)
void lv_style_set_pad_row(lv_style_t *style, int32_t value)
void lv_style_set_pad_column(lv_style_t *style, int32_t value)
void lv_style_set_pad_radial(lv_style_t *style, int32_t value)
void lv_style_set_margin_top(lv_style_t *style, int32_t value)
void lv_style_set_margin_bottom(lv_style_t *style, int32_t value)
void lv_style_set_margin_left(lv_style_t *style, int32_t value)
```

```
void lv_style_set_margin_right(lv_style_t *style, int32_t value)
void lv_style_set_bg_color(lv_style_t *style, lv_color_t value)
void lv_style_set_bg_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_bg_grad_color(lv_style_t *style, lv_color_t value)
void lv_style_set_bg_grad_dir(lv_style_t *style, lv_grad_dir_t value)
void lv_style_set_bg_main_stop(lv_style_t *style, int32_t value)
void lv_style_set_bg_grad_stop(lv_style_t *style, int32_t value)
void lv_style_set_bg_main_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_bg_grad_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_bg_grad(lv_style_t *style, const lv_grad_dsc_t *value)
void lv_style_set_bg_image_src(lv_style_t *style, const void *value)
void lv_style_set_bg_image_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_bg_image_recolor(lv_style_t *style, lv_color_t value)
void lv_style_set_bg_image_recolor_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_bg_image_tiled(lv_style_t *style, bool value)
void lv_style_set_border_color(lv_style_t *style, lv_color_t value)
void lv_style_set_border_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_border_width(lv_style_t *style, int32_t value)
void lv_style_set_border_side(lv_style_t *style, lv_border_side_t value)
void lv_style_set_border_post(lv_style_t *style, bool value)
void lv_style_set_outline_width(lv_style_t *style, int32_t value)
void lv_style_set_outline_color(lv_style_t *style, lv_color_t value)
void lv_style_set_outline_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_outline_pad(lv_style_t *style, int32_t value)
void lv_style_set_shadow_width(lv_style_t *style, int32_t value)
void lv_style_set_shadow_offset_x(lv_style_t *style, int32_t value)
void lv_style_set_shadow_offset_y(lv_style_t *style, int32_t value)
void lv_style_set_shadow_spread(lv_style_t *style, int32_t value)
void lv_style_set_shadow_color(lv_style_t *style, lv_color_t value)
void lv_style_set_shadow_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_image_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_image_recolor(lv_style_t *style, lv_color_t value)
void lv_style_set_image_recolor_opa(lv_style_t *style, lv_opa_t value)
```

```
void lv_style_set_image_colorkey(lv_style_t *style, const lv_image_colorkey_t *value)
void lv_style_set_line_width(lv_style_t *style, int32_t value)
void lv_style_set_line_dash_width(lv_style_t *style, int32_t value)
void lv_style_set_line_dash_gap(lv_style_t *style, int32_t value)
void lv_style_set_line_rounded(lv_style_t *style, bool value)
void lv_style_set_line_color(lv_style_t *style, lv_color_t value)
void lv_style_set_line_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_arc_width(lv_style_t *style, int32_t value)
void lv_style_set_arc_rounded(lv_style_t *style, bool value)
void lv_style_set_arc_color(lv_style_t *style, lv_color_t value)
void lv_style_set_arc_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_arc_image_src(lv_style_t *style, const void *value)
void lv_style_set_text_color(lv_style_t *style, lv_color_t value)
void lv_style_set_text_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_text_font(lv_style_t *style, const lv_font_t *value)
void lv_style_set_text_letter_space(lv_style_t *style, int32_t value)
void lv_style_set_text_line_space(lv_style_t *style, int32_t value)
void lv_style_set_text_decor(lv_style_t *style, lv_text_decor_t value)
void lv_style_set_text_align(lv_style_t *style, lv_text_align_t value)
void lv_style_set_text_outline_stroke_color(lv_style_t *style, lv_color_t value)
void lv_style_set_text_outline_stroke_width(lv_style_t *style, int32_t value)
void lv_style_set_text_outline_stroke_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_radius(lv_style_t *style, int32_t value)
void lv_style_set_radial_offset(lv_style_t *style, int32_t value)
void lv_style_set_clip_corner(lv_style_t *style, bool value)
void lv_style_set_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_opa_layered(lv_style_t *style, lv_opa_t value)
void lv_style_set_color_filter_dsc(lv_style_t *style, const lv_color_filter_dsc_t *value)
void lv_style_set_color_filter_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_recolor(lv_style_t *style, lv_color_t value)
void lv_style_set_recolor_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_anim(lv_style_t *style, const lv_anim_t *value)
void lv_style_set_anim_duration(lv_style_t *style, uint32_t value)
```

```
void lv_style_set_transition(lv_style_t *style, const lv_style_transition_dsc_t *value)
void lv_style_set_blend_mode(lv_style_t *style, lv_blend_mode_t value)
void lv_style_set_layout(lv_style_t *style, uint16_t value)
void lv_style_set_base_dir(lv_style_t *style, lv_base_dir_t value)
void lv_style_set_bitmap_mask_src(lv_style_t *style, const void *value)
void lv_style_set_rotary_sensitivity(lv_style_t *style, uint32_t value)
```

14.21.35 lv_style_private.h

14.21.36 lv_temp.h

14.21.37 lv_text.h

Enum

enum lv_text_flag_t

Options for text rendering.

Values:

enumerator **LV_TEXT_FLAG_NONE**

enumerator **LV_TEXT_FLAG_EXPAND**

enumerator **LV_TEXT_FLAG_FIT**

Max-width is already equal to the longest line. (Used to skip some calculation)

enumerator **LV_TEXT_FLAG_BREAK_ALL**

To prevent overflow, insert breaks between any two characters. Otherwise breaks are inserted at word boundaries, as configured via `LV_TXT_BREAK_CHARS` or according to `LV_TXT_LINE_BREAK_LONG_LEN`, `LV_TXT_LINE_BREAK_LONG_PRE_MIN_LEN`, and `LV_TXT_LINE_BREAK_LONG_POST_MIN_LEN`.

enumerator **LV_TEXT_FLAG_RECOLOR**

Enable parsing of recolor command

enum lv_text_align_t

Label align policy

Values:

enumerator **LV_TEXT_ALIGN_AUTO**

Align text auto

enumerator **LV_TEXT_ALIGN_LEFT**

Align text to left

enumerator **LV_TEXT_ALIGN_CENTER**

Align text to center

enumerator **LV_TEXT_ALIGN_RIGHT**

Align text to right

Funzioni

```
void lv_text_get_size(lv_point_t *size_res, const char *text, const lv_font_t *font, int32_t letter_space, int32_t line_space, int32_t max_width, lv_text_flag_t flag)
```

Get size of a text

Parametri

- **size_res** -- pointer to a 'point_t' variable to store the result
- **text** -- pointer to a text
- **font** -- pointer to font of the text
- **letter_space** -- letter space of the text
- **line_space** -- line space of the text
- **max_width** -- max width of the text (break the lines to fit this size). Set COORD_MAX to avoid
- **flag** -- settings for the text from *lv_text_flag_t*

14.21.38 lv_text_ap.h

Define

LV_UNDEF_ARABIC_PERSIAN_CHARS

LV_AP_ALPHABET_BASE_CODE

LV_AP_END_CHARS_LIST

Funzioni

```
uint32_t lv_text_ap_calc_bytes_count(const char *txt)
```

```
void lv_text_ap_proc(const char *txt, char *txt_out)
```

14.21.39 lv_text_private.h

Define

LV_TXT_ENC_UTF8

LV_TXT_ENC_ASCII

LV_TEXT_LEN_MAX

Enum

enum **lv_text_cmd_state_t**

State machine for text renderer.

Values:

enumerator **LV_TEXT_CMD_STATE_WAIT**

Waiting for command

enumerator **LV_TEXT_CMD_STATE_PAR**

Processing the parameter

enumerator **LV_TEXT_CMD_STATE_IN**

Processing the command

Funzioni

void lv_text_attributes_init(*lv_text_attributes_t* *attributes)

Initialize the text attributes descriptor

Parametri

attributes -- the text attributes descriptor to initialize

**void lv_text_get_size_attributes(*lv_point_t* *size_res, const char *text, const *lv_font_t* *font,
 lv_text_attributes_t *attributes)**

Get size of a text

Parametri

- **size_res** -- pointer to a 'point_t' variable to store the result
- **text** -- pointer to a text
- **font** -- pointer to font of the text
- **attributes** -- the text attributes, flags for line break behaviour, spacing etc

**int32_t lv_text_get_width(const char *txt, uint32_t length, const *lv_font_t* *font, const *lv_text_attributes_t*
 *attributes)**

Give the length of a text with a given font with text flags

Parametri

- **txt** -- a '\0' terminate string
- **length** -- length of 'txt' in byte count and not characters (À is 1 character but 2 bytes in UTF-8)
- **font** -- pointer to font of the text
- **attributes** -- the text attributes, flags for line break behaviour, spacing etc

Ritorna

length of a char_num long text

bool lv_text_is_cmd(*lv_text_cmd_state_t* *state, uint32_t c)

Check if c is command state

Parametri

- **state** --
- **c** --

Ritorna

True if c is state

**uint32_t lv_text_get_next_line(const char *txt, uint32_t len, const *lv_font_t* *font, int32_t *used_width,
 lv_text_attributes_t *attributes)**

Get the next line of text. Check line length and break chars too.

Parametri

- **txt** -- a '\0' terminated string
- **len** -- length of 'txt' in bytes
- **font** -- pointer to a font
- **used_width** -- When used_width != NULL, save the width of this line if flag == LV_TEXT_FLAG_NONE, otherwise save -1.
- **attributes** -- text attributes, flags to control line break behaviour, spacing etc

Ritorna

the index of the first char of the new line (in byte index not letter index. With UTF-8 they are different)

void lv_text_ins(char *txt_buf, uint32_t pos, const char *ins_txt)

Insert a string into another

Parametri

- **txt_buf** -- the original text (must be big enough for the result text and NULL terminated)
- **pos** -- position to insert (0: before the original text, 1: after the first char etc.)
- **ins_txt** -- text to insert, must be '\0' terminated

`void lv_text_cut(char *txt, uint32_t pos, uint32_t len)`

Delete a part of a string

Parametri

- **txt** -- string to modify, must be '\0' terminated and should point to a heap or stack frame, not read-only memory.
- **pos** -- position where to start the deleting (0: before the first char, 1: after the first char etc.)
- **len** -- number of characters to delete

`char *lv_text_set_text_vfmt(const char *fmt, va_list ap)`

Return a new formatted text. Memory will be allocated to store the text.

Parametri

- **fmt** -- printf-like format
- **ap** -- items to print

Ritorna

pointer to the allocated text string.

`void lv_text_encoded_letter_next_2(const char *txt, uint32_t *letter, uint32_t *letter_next, uint32_t *ofs)`

Decode two encoded character from a string.

Parametri

- **txt** -- pointer to '\0' terminated string
- **letter** -- the first decoded Unicode character or 0 on invalid data code
- **letter_next** -- the second decoded Unicode character or 0 on invalid data code
- **ofs** -- start index in 'txt' where to start. After the call it will point to the next encoded char in 'txt'. NULL to use txt[0] as index

`static inline bool lv_text_is_break_char(uint32_t letter)`

Test if char is break char or not (a text can broken here or not)

Parametri

letter -- a letter

Ritorna

false: 'letter' is not break char

`static inline bool lv_text_is_a_word(uint32_t letter)`

Test if char is break char or not (a text can broken here or not)

Parametri

letter -- a letter

Ritorna

false: 'letter' is not break char

`static inline bool lv_text_is_marker(uint32_t letter)`

Test if character can be treated as marker, and don't need to be rendered. Note, this is not a full list. Add your findings to the list.

Parametri

letter -- a letter

Ritorna

true if so

Variabili

`uint8_t (*const lv_text_encoded_size)(const char *txt)`

Give the size of an encoded character

Param txt

pointer to a character in a string

Return

length of the encoded character (1,2,3 ...). 0 in invalid

```
uint32_t (*const lv_text_unicode_to_encoded)(uint32_t letter_uni)
```

Convert a Unicode letter to encoded

Param letter_uni

a Unicode letter

Return

Encoded character in Little Endian to be compatible with C chars (e.g. 'Á', 'Ü')

```
uint32_t (*const lv_text_encoded_conv_wc)(uint32_t c)
```

Convert a wide character, e.g. 'Á' little endian to be compatible with the encoded format.

Param c

a wide character

Return

c in the encoded format

```
uint32_t (*const lv_text_encoded_next)(const char *txt, uint32_t *i_start)
```

Decode the next encoded character from a string.

Param txt

pointer to '\0' terminated string

Param i_start

start index in 'txt' where to start. After the call it will point to the next encoded char in 'txt'.

NULL to use txt[0] as index

Return

the decoded Unicode character or 0 on invalid data code

```
uint32_t (*const lv_text_encoded_prev)(const char *txt, uint32_t *i_start)
```

Get the previous encoded character form a string.

Param txt

pointer to '\0' terminated string

Param i_start

index in 'txt' where to start. After the call it will point to the previous encoded char in 'txt'.

Return

the decoded Unicode character or 0 on invalid data

```
uint32_t (*const lv_text_encoded_get_byte_id)(const char *txt, uint32_t utf8_id)
```

Convert a letter index (in the encoded text) to byte index. E.g. in UTF-8 "AÁRT" index of 'R' is 2 but start at byte 3 because 'Á' is 2 bytes long

Param txt

a '\0' terminated UTF-8 string

Param utf8_id

character index

Return

byte index of the 'enc_id'th letter

```
uint32_t (*const lv_text_encoded_get_char_id)(const char *txt, uint32_t byte_id)
```

Convert a byte index (in an encoded text) to character index. E.g. in UTF-8 "AÁRT" index of 'R' is 2 but start at byte 3 because 'Á' is 2 bytes long

Param txt

a '\0' terminated UTF-8 string

Param byte_id

byte index

Return

character index of the letter at 'byte_id'th position

```
uint32_t (*const lv_text_get_encoded_length)(const char *txt)
```

Get the number of characters (and NOT bytes) in a string. E.g. in UTF-8 "ÁBC" is 3 characters (but 4 bytes)

Param txt

a '\0' terminated char string

Return

number of characters

```
struct lv_text_attributes_t
```

Membri Pubblici

```
int32_t letter_space
```

Letter space between letters

```
int32_t line_space
```

Space between lines of text

```
int32_t max_width
```

Max width of the text (break the lines to fit this size). Set COORD_MAX to avoid

```
lv_text_flag_t text_flags
```

14.21.40 lv_timer.h

Define

```
LV_NO_TIMER_READY
```

Typedef

```
typedef void (*lv_timer_cb_t)(lv_timer_t*)
```

Timers execute this type of functions.

```
typedef void (*lv_timer_handler_resume_cb_t)(void *data)
```

Timer handler resume this type of function.

Funzioni

```
uint32_t lv_timer_handler(void)
```

Call it periodically to handle lv_timers.

Ritorna

time till it needs to be run next (in ms)

```
uint32_t lv_timer_handler_run_in_period(uint32_t period)
```

Call it in the super-loop of main() or threads. It will run [*lv_timer_handler\(\)*](#) with a given period in ms. You can use it with sleep or delay in OS environment. This function is used to simplify the porting.

Parametri

period -- the period for running [*lv_timer_handler\(\)*](#)

Ritorna

the time after which it must be called again

```
void lv_timer_periodic_handler(void)
```

Call it in the super-loop of main() or threads. It will automatically call `lv_timer_handler()` at the right time.
This function is used to simplify the porting.

```
void lv_timer_handler_set_resume_cb(lv_timer_handler_resume_cb_t cb, void *data)
```

Set the resume callback to the timer handler

Parametri

- **cb** -- the function to call when timer handler is resumed
- **data** -- pointer to a resume data

```
lv_timer_t *lv_timer_create_basic(void)
```

Create an "empty" timer. It needs to be initialized with at least `lv_timer_set_cb` and `lv_timer_set_period`
Ritorna

pointer to the created timer

```
lv_timer_t *lv_timer_create(lv_timer_cb_t timer_xcb, uint32_t period, void *user_data)
```

Create a new lv_timer

Parametri

- **timer_xcb** -- a callback to call periodically. (the 'x' in the argument name indicates that it's not a fully generic function because it not follows the `func_name(object, callback, ...)` convention)
- **period** -- call period in ms unit
- **user_data** -- custom parameter

Ritorna

pointer to the new timer

```
void lv_timer_delete(lv_timer_t *timer)
```

Delete a lv_timer

Parametri

timer -- pointer to an lv_timer

```
void lv_timer_pause(lv_timer_t *timer)
```

Pause a timer. It is typically safe to call from an interrupt handler or a different thread.

Parametri

timer -- pointer to an lv_timer

```
void lv_timer_resume(lv_timer_t *timer)
```

Resume a timer.

Parametri

timer -- pointer to an lv_timer

```
void lv_timer_set_cb(lv_timer_t *timer, lv_timer_cb_t timer_cb)
```

Set the callback to the timer (the function to call periodically)

Parametri

- **timer** -- pointer to a timer
- **timer_cb** -- the function to call periodically

```
void lv_timer_set_period(lv_timer_t *timer, uint32_t period)
```

Set new period for a lv_timer

Parametri

- **timer** -- pointer to a lv_timer
- **period** -- the new period

```
void lv_timer_ready(lv_timer_t *timer)
```

Make a lv_timer ready. It will not wait its period.

Parametri

timer -- pointer to a lv_timer.

```
void lv_timer_set_repeat_count(lv_timer_t *timer, int32_t repeat_count)
```

Set the number of times a timer will repeat.

Parametri

- **timer** -- pointer to a lv_timer.
- **repeat_count** -- -1 : infinity; 0 : stop ; n>0: residual times

void lv_timer_set_auto_delete(lv_timer_t *timer, bool auto_delete)

Set whether a lv_timer will be deleted automatically when it is called repeat_count times.

Parametri

- **timer** -- pointer to a lv_timer.
- **auto_delete** -- true: auto delete; false: timer will be paused when it is called repeat_count times.

void lv_timer_set_user_data(lv_timer_t *timer, void *user_data)

Set custom parameter to the lv_timer.

Parametri

- **timer** -- pointer to a lv_timer.
- **user_data** -- custom parameter

void lv_timer_reset(lv_timer_t *timer)

Reset a lv_timer. It will be called the previously set period milliseconds later.

Parametri

timer -- pointer to a lv_timer.

void lv_timer_enable(bool en)

Enable or disable the whole lv_timer handling

Parametri

en -- true: lv_timer handling is running, false: lv_timer handling is suspended

uint32_t lv_timer_get_idle(void)

Get idle percentage

Ritorna

the lv_timer idle in percentage

uint32_t lv_timer_get_time_until_next(void)

Get the time remaining until the next timer will run

Ritorna

the time remaining in ms

lv_timer_t *lv_timer_get_next(lv_timer_t *timer)

Iterate through the timers

Parametri

timer -- NULL to start iteration or the previous return value to get the next timer

Ritorna

the next timer or NULL if there is no more timer

void *lv_timer_get_user_data(lv_timer_t *timer)

Get the user_data passed when the timer was created

Parametri

timer -- pointer to the lv_timer

Ritorna

pointer to the user_data

bool lv_timer_get_paused(lv_timer_t *timer)

Get the pause state of a timer

Parametri

timer -- pointer to a lv_timer

Ritorna

true: timer is paused; false: timer is running

14.21.41 lv_timer_private.h

Funzioni

```
void lv_timer_core_init(void)
    Init the lv_timer module

void lv_timer_core_deinit(void)
    Deinit the lv_timer module

struct _lv_timer_t
    #include <lv_timer_private.h> Descriptor of a lv_timer
```

Membri Pubblici

```
uint32_t period
    How often the timer should run

uint32_t last_run
    Last time the timer ran

lv_timer_cb_t timer_cb
    Timer function

void *user_data
    Custom user data

int32_t repeat_count
    1: One time; -1 : infinity; n>0: residual times

volatile int paused

uint32_t auto_delete

struct lv_timer_state_t
```

Membri Pubblici

```
lv_ll_t timer_ll
    Linked list to store the lv_timers

bool lv_timer_run

uint8_t idle_last

bool timer_deleted

bool timer_created

volatile uint32_t timer_time_until_next
```

```

bool already_running

uint32_t periodic_last_tick

uint32_t busy_time

uint32_t idle_period_start

uint32_t run_cnt

lv_timer_handler_resume_cb_t resume_cb

void *resume_data

```

14.21.42 lv_tree.h

Tree. The tree nodes are dynamically allocated by the 'lv_mem' module.

Define

```
LV_TREE_NODE(n)
```

Typedef

```
typedef struct lv_tree_class_t lv_tree_class_t
```

```
typedef struct lv_tree_node_t lv_tree_node_t
```

```
typedef void (*lv_tree_constructor_cb_t)(const lv_tree_class_t *class_p, lv_tree_node_t *node)
```

```
typedef void (*lv_tree_destructor_cb_t)(const lv_tree_class_t *class_p, lv_tree_node_t *node)
```

```
typedef uint8_t lv_tree_walk_mode_t
```

```
typedef bool (*lv_tree_traverse_cb_t)(const lv_tree_node_t *node, void *user_data)
```

```
typedef bool (*lv_tree_before_cb_t)(const lv_tree_node_t *node, void *user_data)
```

```
typedef void (*lv_tree_after_cb_t)(const lv_tree_node_t *node, void *user_data)
```

Enum

```
enum lv_tree_walk_mode_t
```

Values:

enumerator **LV_TREE_WALK_PRE_ORDER**

enumerator **LV_TREE_WALK_POST_ORDER**

Funzioni

`lv_tree_node_t *lv_tree_node_create(const lv_tree_class_t *class_p, lv_tree_node_t *parent)`

Create a tree node.

Parametri

- **class_p** -- pointer to a class of the node
- **parent** -- pointer to the parent node (or NULL if it's the root node)

Ritorna

pointer to the new node

`void lv_tree_node_delete(lv_tree_node_t *node)`

Delete a tree node and all its children recursively.

Parametri

node -- pointer to the node to delete

`bool lv_tree_walk(const lv_tree_node_t *node, lv_tree_walk_mode_t mode, lv_tree_traverse_cb_t cb, lv_tree_before_cb_t bcb, lv_tree_after_cb_t acb, void *user_data)`

Walk the tree recursively and call a callback function on each node.

Parametri

- **node** -- pointer to the root node of the tree
- **mode** -- LV_TREE_WALK_PRE_ORDER or LV_TREE_WALK_POST_ORDER
- **cb** -- callback function to call on each node
- **bcb** -- callback function to call before visiting a node
- **acb** -- callback function to call after visiting a node
- **user_data** -- user data to pass to the callback functions

Ritorna

true: traversal is finished; false: traversal broken

Variabili

const `lv_tree_class_t lv_tree_node_class`

struct `_lv_tree_class_t`

#include <lv_tree.h> Describe the common methods of every object. Similar to a C++ class.

Membri Pubblici

const `lv_tree_class_t *base_class`

`uint32_t instance_size`

`lv_tree_constructor_cb_t constructor_cb`

`lv_tree_destructor_cb_t destructor_cb`

struct `_lv_tree_node_t`

#include <lv_tree.h> Description of a tree node

Membri Pubblici

`lv_tree_node_t *parent`

`lv_tree_node_t **children`

```
uint32_t child_cnt

uint32_t child_cap

const lv_tree_class_t *class_p
```

14.21.43 lv_utils.h

Funzioni

`void *lv_utils_bsearch(const void *key, const void *base, size_t n, size_t size, int (*cmp)(const void *pRef, const void *pElement))`

Searches base[0] to base[n - 1] for an item that matches *key.

i Nota

The function cmp must return negative if it's first argument (the search key) is less than its second (a table entry), zero if equal, and positive if greater.

i Nota

Items in the array must be in ascending order.

Parametri

- **key** -- Pointer to item being searched for
- **base** -- Pointer to first element to search
- **n** -- Number of elements
- **size** -- Size of each element
- **cmp** -- Pointer to comparison function (see `unicode_list_compare()` as a comparison function example)

Ritorna

a pointer to a matching item, or NULL if none exists.

`lv_result_t lv_draw_buf_save_to_file(const lv_draw_buf_t *draw_buf, const char *path)`

Save a draw buf to a file

Parametri

- **draw_buf** -- pointer to a draw buffer
- **path** -- path to the file to save

Ritorna

`LV_RESULT_OK`: success; `LV_RESULT_INVALID`: error

`static inline uint32_t lv_swap_bytes_32(uint32_t x)`

Reverse the order of the bytes in a 32-bit value.

Parametri

`x` -- a 32-bit value.

Ritorna

the value `x` with reversed byte-order.

`static inline uint16_t lv_swap_bytes_16(uint16_t x)`

Reverse the order of the bytes in a 16-bit value.

Parametri

`x` -- a 16-bit value.

Ritorna

the value `x` with reversed byte-order.

14.21.44 cache

lv_cache.h

Funzioni

`lv_cache_t *lv_cache_create(const lv_cache_class_t *cache_class, size_t node_size, size_t max_size, lv_cache_ops_t ops)`

Create a cache object with the given parameters.

Parametri

- **cache_class** -- The class of the cache. Currently only support one two builtin classes:
 - `lv_cache_class_lru_rb_count` for LRU-based cache with count-based eviction policy.
 - `lv_cache_class_lru_rb_size` for LRU-based cache with size-based eviction policy.
- **node_size** -- The node size is the size of the data stored in the cache..
- **max_size** -- The max size is the maximum amount of memory or count that the cache can hold.
 - `lv_cache_class_lru_rb_count`: max_size is the maximum count of nodes in the cache.
 - `lv_cache_class_lru_rb_size`: max_size is the maximum size of the cache in bytes.
- **ops** -- A set of operations that can be performed on the cache. See `lv_cache_ops_t` for details.

Ritorna

Returns a pointer to the created cache object on success, `NULL` on error.

`void lv_cache_destroy(lv_cache_t *cache, void *user_data)`

Destroy a cache object.

Parametri

- **cache** -- The cache object pointer to destroy.
- **user_data** -- A user data pointer that will be passed to the free callback.

`lv_cache_entry_t *lv_cache_acquire(lv_cache_t *cache, const void *key, void *user_data)`

Acquire a cache entry with the given key. If entry not in cache, it will return `NULL` (not found). If the entry is found, it's priority will be changed by the cache's policy. And the `lv_cache_entry_t::ref_cnt` will be incremented.

Parametri

- **cache** -- The cache object pointer to acquire the entry.
- **key** -- The key of the entry to acquire.
- **user_data** -- A user data pointer that will be passed to the create callback.

Ritorna

Returns a pointer to the acquired cache entry on success with `lv_cache_entry_t::ref_cnt` incremented, `NULL` on error.

`lv_cache_entry_t *lv_cache_acquire_or_create(lv_cache_t *cache, const void *key, void *user_data)`

Acquire a cache entry with the given key. If the entry is not in the cache, it will create a new entry with the given key. If the entry is found, it's priority will be changed by the cache's policy. And the `lv_cache_entry_t::ref_cnt` will be incremented. If you want to use this API to simplify the code, you should provide a `lv_cache_ops_t::create_cb` that creates a new entry with the given key. This API is a combination of `lv_cache_acquire()` and `lv_cache_add()`. The effect is the same as calling `lv_cache_acquire()` and `lv_cache_add()` separately. And the internal impact on cache is also consistent with these two APIs.

Parametri

- **cache** -- The cache object pointer to acquire the entry.
- **key** -- The key of the entry to acquire or create.
- **user_data** -- A user data pointer that will be passed to the create callback.

Ritorna

Returns a pointer to the acquired or created cache entry on success with `lv_cache_entry_t::ref_cnt` incremented, NULL on error.

```
lv_cache_entry_t *lv_cache_add(lv_cache_t *cache, const void *key, void *user_data)
```

Add a new cache entry with the given key and data. If the cache is full, the cache's policy will be used to evict an entry.

Parametri

- **cache** -- The cache object pointer to add the entry.
- **key** -- The key of the entry to add.
- **user_data** -- A user data pointer that will be passed to the create callback.

Ritorna

Returns a pointer to the added cache entry on success with `lv_cache_entry_t::ref_cnt` incremented, NULL on error.

```
void lv_cache_release(lv_cache_t *cache, lv_cache_entry_t *entry, void *user_data)
```

Release a cache entry. The `lv_cache_entry_t::ref_cnt` will be decremented. If the `lv_cache_entry_t::ref_cnt` is zero, it will issue an error. If the entry passed to this function is the last reference to the data and the entry is marked as invalid, the cache's policy will be used to evict the entry.

Parametri

- **cache** -- The cache object pointer to release the entry.
- **entry** -- The cache entry pointer to release.
- **user_data** -- A user data pointer that will be passed to the free callback.

```
void lv_cache_reserve(lv_cache_t *cache, uint32_t reserved_size, void *user_data)
```

Reserve a certain amount of memory/count in the cache. This function is useful when you want to reserve a certain amount of memory/count in advance, for example, when you know that you will need it later. When the current cache size is max than the reserved size, the function will evict entries until the reserved size is reached.

Parametri

- **cache** -- The cache object pointer to reserve.
- **reserved_size** -- The amount of memory/count to reserve.
- **user_data** -- A user data pointer that will be passed to the free callback.

```
void lv_cache_drop(lv_cache_t *cache, const void *key, void *user_data)
```

Drop a cache entry with the given key. If the entry is not in the cache, nothing will happen to it. If the entry is found, it will be removed from the cache and its data will be freed when the last reference to it is released.

Nota

The data will not be freed immediately but when the last reference to it is released. But this entry will not be found by `lv_cache_acquire()`. If you want cache a same key again, you should use `lv_cache_add()` or `lv_cache_acquire_or_create()`.

Parametri

- **cache** -- The cache object pointer to drop the entry.
- **key** -- The key of the entry to drop.
- **user_data** -- A user data pointer that will be passed to the free callback.

```
void lv_cache_drop_all(lv_cache_t *cache, void *user_data)
```

Drop all cache entries. All entries will be removed from the cache and their data will be freed when the last reference to them is released.

Nota

If some entries are still referenced by other objects, it will issue an error. And this case shouldn't happen in normal cases..

Parametri

- **cache** -- The cache object pointer to drop all entries.
- **user_data** -- A user data pointer that will be passed to the free callback.

`bool lv_cache_evict_one(lv_cache_t *cache, void *user_data)`

Evict one entry from the cache. The eviction policy will be used to select the entry to evict.

Parametri

- **cache** -- The cache object pointer to evict an entry.
- **user_data** -- A user data pointer that will be passed to the free callback.

Ritorna

Returns true if an entry is evicted, false if no entry is evicted.

`void lv_cache_set_max_size(lv_cache_t *cache, size_t max_size, void *user_data)`

Set the maximum size of the cache. If the current cache size is greater than the new maximum size, the cache's policy will be used to evict entries until the new maximum size is reached. If set to 0, the cache will be disabled.

Nota

But this behavior will happen only new entries are added to the cache.

Parametri

- **cache** -- The cache object pointer to set the maximum size.
- **max_size** -- The new maximum size of the cache.
- **user_data** -- A user data pointer that will be passed to the free callback.

`size_t lv_cache_get_max_size(lv_cache_t *cache, void *user_data)`

Get the maximum size of the cache.

Parametri

- **cache** -- The cache object pointer to get the maximum size.
- **user_data** -- A user data pointer that will be passed to the free callback.

Ritorna

Returns the maximum size of the cache.

`size_t lv_cache_get_size(lv_cache_t *cache, void *user_data)`

Get the current size of the cache.

Parametri

- **cache** -- The cache object pointer to get the current size.
- **user_data** -- A user data pointer that will be passed to the free callback.

Ritorna

Returns the current size of the cache.

`size_t lv_cache_get_free_size(lv_cache_t *cache, void *user_data)`

Get the free size of the cache.

Parametri

- **cache** -- The cache object pointer to get the free size.
- **user_data** -- A user data pointer that will be passed to the free callback.

Ritorna

Returns the free size of the cache.

`bool lv_cache_is_enabled(lv_cache_t *cache)`

Return true if the cache is enabled. Disabled cache means that when the max_size of the cache is 0. In this case, all cache operations will be no-op.

Parametri

cache -- The cache object pointer to check if it's disabled.

Ritorna

Returns true if the cache is enabled, false otherwise.

`void lv_cache_set_compare_cb(lv_cache_t *cache, lv_cache_compare_cb_t compare_cb, void *user_data)`

Set the compare callback of the cache.

Parametri

- **cache** -- The cache object pointer to set the compare callback.
- **compare_cb** -- The compare callback to set.
- **user_data** -- A user data pointer.

`void lv_cache_set_create_cb(lv_cache_t *cache, lv_cache_create_cb_t alloc_cb, void *user_data)`

Set the create callback of the cache.

Parametri

- **cache** -- The cache object pointer to set the create callback.
- **alloc_cb** -- The create callback to set.
- **user_data** -- A user data pointer.

`void lv_cache_set_free_cb(lv_cache_t *cache, lv_cache_free_cb_t free_cb, void *user_data)`

Set the free callback of the cache.

Parametri

- **cache** -- The cache object pointer to set the free callback.
- **free_cb** -- The free callback to set.
- **user_data** -- A user data pointer.

`void lv_cache_set_name(lv_cache_t *cache, const char *name)`

Give a name for a cache object. Only the pointer of the string is saved.

Parametri

- **cache** -- The cache object pointer to set the name.
- **name** -- The name of the cache.

`const char *lv_cache_get_name(lv_cache_t *cache)`

Get the name of a cache object.

Parametri

cache -- The cache object pointer to get the name.

Ritorna

Returns the name of the cache.

`lv_iter_t *lv_cache_iter_create(lv_cache_t *cache)`

Create an iterator for the cache object. The iterator is used to iterate over all cache entries.

Parametri

cache -- The cache object pointer to create the iterator.

Ritorna

Returns a pointer to the created iterator on success, NULL on error.

lv_cache_entry.h

Funzioni

`uint32_t lv_cache_entry_get_size(const uint32_t node_size)`

Get the size of a cache entry.

Parametri

node_size -- The size of the node in the cache.

Ritorna

The size of the cache entry.

`int32_t lv_cache_entry_get_ref(lv_cache_entry_t *entry)`

Get the reference count of a cache entry.

Parametri

entry -- The cache entry to get the reference count of.

Ritorna

The reference count of the cache entry.

`uint32_t lv_cache_entry_get_node_size(lv_cache_entry_t *entry)`

Get the node size of a cache entry. Which is the same size with `lv_cache_entry_get_size()`'s `node_size` parameter.

Parametri

entry -- The cache entry to get the node size of.

Ritorna

The node size of the cache entry.

```
bool lv_cache_entry_is_invalid(lv_cache_entry_t *entry)
```

Check if a cache entry is invalid.

Parametri

entry -- The cache entry to check.

Ritorna

True: the cache entry is invalid. False: the cache entry is valid.

```
void *lv_cache_entry_get_data(lv_cache_entry_t *entry)
```

Get the data of a cache entry.

Parametri

entry -- The cache entry to get the data of.

Ritorna

The pointer to the data of the cache entry.

```
const lv_cache_t *lv_cache_entry_get_cache(const lv_cache_entry_t *entry)
```

Get the cache instance of a cache entry.

Parametri

entry -- The cache entry to get the cache instance of.

Ritorna

The pointer to the cache instance of the cache entry.

```
lv_cache_entry_t *lv_cache_entry_get_entry(void *data, const uint32_t node_size)
```

Get the cache entry of a data. The data should be allocated by the cache instance.

Parametri

- **data** -- The data to get the cache entry of.
- **node_size** -- The size of the node in the cache.

Ritorna

The pointer to the cache entry of the data.

```
lv_cache_entry_t *lv_cache_entry_alloc(const uint32_t node_size, const lv_cache_t *cache)
```

Allocate a cache entry.

Parametri

- **node_size** -- The size of the node in the cache.
- **cache** -- The cache instance to allocate the cache entry from.

Ritorna

The pointer to the allocated cache entry.

```
void lv_cache_entry_init(lv_cache_entry_t *entry, const lv_cache_t *cache, const uint32_t node_size)
```

Initialize a cache entry.

Parametri

- **entry** -- The cache entry to initialize.
- **cache** -- The cache instance to allocate the cache entry from.
- **node_size** -- The size of the node in the cache.

```
void lv_cache_entry_delete(lv_cache_entry_t *entry)
```

Deallocate a cache entry. And the data of the cache entry will be freed.

Parametri

entry -- The cache entry to deallocate.

lv_cache_entry_private.h

Define

LV_CACHE_ENTRY_FLAG_INVALID

LV_CACHE_ENTRY_FLAG_DISABLE_DELETE

LV_CACHE_ENTRY_FLAG_CLASS_CUSTOM**Funzioni**

```
void lv_cache_entry_reset_ref(lv_cache_entry_t *entry)
void lv_cache_entry_inc_ref(lv_cache_entry_t *entry)
void lv_cache_entry_dec_ref(lv_cache_entry_t *entry)
void lv_cache_entry_set_node_size(lv_cache_entry_t *entry, uint32_t node_size)
void lv_cache_entry_set_cache(lv_cache_entry_t *entry, const lv_cache_t *cache)
void *lv_cache_entry_acquire_data(lv_cache_entry_t *entry)
void lv_cache_entry_release_data(lv_cache_entry_t *entry, void *user_data)
void lv_cache_entry_set_flag(lv_cache_entry_t *entry, uint8_t flags)
void lv_cache_entry_remove_flag(lv_cache_entry_t *entry, uint8_t flags)
bool lv_cache_entry_has_flag(lv_cache_entry_t *entry, uint8_t flags)

struct _lv_cache_entry_t
```

Membri Pubblici

```
const lv_cache_t *cache
```

```
int32_t ref_cnt
```

```
uint32_t node_size
```

```
uint8_t flags
```

lv_cache_private.h**Typedef**

```
typedef struct _lv_cache_ops_t lv_cache_ops_t
```

```
typedef struct _lv_cache_class_t lv_cache_class_t
```

```
typedef int8_t lv_cache_compare_res_t
```

```
typedef bool (*lv_cache_create_cb_t)(void *node, void *user_data)
```

```
typedef void (*lv_cache_free_cb_t)(void *node, void *user_data)
```

```
typedef lv_cache_compare_res_t (*lv_cache_compare_cb_t)(const void *a, const void *b)
```

```
typedef void *(*lv_cache_alloc_cb_t)(void)
```

The cache instance allocation function, used by the cache class to allocate memory for cache instances.

Return

It should return a pointer to the allocated instance.

```
typedef bool (*lv_cache_init_cb_t)(lv_cache_t *cache)
```

The cache instance initialization function, used by the cache class to initialize the cache instance.

Return

It should return true if the initialization is successful, false otherwise.

```
typedef void (*lv_cache_destroy_cb_t)(lv_cache_t *cache, void *user_data)
```

The cache instance destruction function, used by the cache class to destroy the cache instance.

```
typedef lv_cache_entry_t *(*lv_cache_get_cb_t)(lv_cache_t *cache, const void *key, void *user_data)
```

The cache get function, used by the cache class to get a cache entry by its key.

Return

NULL if the key is not found.

```
typedef lv_cache_entry_t *(*lv_cache_add_cb_t)(lv_cache_t *cache, const void *key, void *user_data)
```

The cache add function, used by the cache class to add a cache entry with a given key. This function only cares about how to add the entry, it doesn't check if the entry already exists and doesn't care about is it a victim or not.

Return

the added cache entry, or NULL if the entry is not added.

```
typedef void (*lv_cache_remove_cb_t)(lv_cache_t *cache, lv_cache_entry_t *entry, void *user_data)
```

The cache remove function, used by the cache class to remove a cache entry from the cache but doesn't free the memory.. This function only cares about how to remove the entry, it doesn't care about is it a victim or not.

```
typedef void (*lv_cache_drop_cb_t)(lv_cache_t *cache, const void *key, void *user_data)
```

The cache drop function, used by the cache class to remove a cache entry from the cache and free the memory.

```
typedef void (*lv_cache_drop_all_cb_t)(lv_cache_t *cache, void *user_data)
```

The cache drop all function, used by the cache class to remove all cache entries from the cache and free the memory.

```
typedef lv_cache_entry_t *(*lv_cache_get_victim_cb)(lv_cache_t *cache, void *user_data)
```

The cache get victim function, used by the cache class to get a victim entry to be evicted.

```
typedef lv_cache_reserve_cond_res_t (*lv_cache_reserve_cond_cb)(lv_cache_t *cache, const void *key, size_t size, void *user_data)
```

The cache reserve condition function, used by the cache class to check if a new entry can be added to the cache without exceeding its maximum size. See [lv_cache_reserve_cond_res_t](#) for the possible results.

```
typedef lv_iter_t *(*lv_cache_iter_create_cb)(lv_cache_t *cache)
```

The cache iterator creation function, used by the cache class to create an iterator for the cache.

Return

A pointer to the created iterator, or NULL if the iterator cannot be created.

```
typedef struct _lv_cache_slot_size_t lv_cache_slot_size_t
```

Enum

enum **lv_cache_reserve_cond_res_t**

The result of the cache reserve condition callback

Values:

enumerator **LV_CACHE_RESERVE_COND_OK**

The condition is met and no entries need to be evicted

enumerator **LV_CACHE_RESERVE_COND_TOO_LARGE**

The condition is not met and the reserve size is too large

enumerator **LV_CACHE_RESERVE_COND_NEED_VICTIM**

The condition is not met and a victim is needed to be evicted

enumerator **LV_CACHE_RESERVE_COND_ERROR**

An error occurred while checking the condition

struct **_lv_cache_ops_t**

#include <lv_cache_private.h> The cache operations struct

Membri Pubblici

lv_cache_compare_cb_t compare_cb

Compare function for keys

lv_cache_create_cb_t create_cb

Create function for nodes

lv_cache_free_cb_t free_cb

Free function for nodes

struct **_lv_cache_t**

#include <lv_cache_private.h> The cache entry struct

Membri Pubblici

const *lv_cache_class_t *clz*

Cache class. There are two built-in classes:

- *lv_cache_class_lru_rb_count* for LRU-based cache with count-based eviction policy.
- *lv_cache_class_lru_rb_size* for LRU-based cache with size-based eviction policy.

uint32_t node_size

Size of a node

uint32_t max_size

Maximum size of the cache

uint32_t size

Current size of the cache

***lv_cache_ops_t* ops**

Cache operations struct *lv_cache_ops_t*

***lv_mutex_t* lock**

Cache lock used to protect the cache in multithreading environments

const char *name

Name of the cache

struct *lv_cache_class_t*

#include <lv_cache_private.h> Cache class struct for building custom cache classes

Examples:

- lv_cache_class_lru_rb_count for LRU-based cache with count-based eviction policy.
- lv_cache_class_lru_rb_size for LRU-based cache with size-based eviction policy.

Membri Pubblici***lv_cache_alloc_cb_t* alloc_cb**

The allocation function for cache entries

***lv_cache_init_cb_t* init_cb**

The initialization function for cache entries

***lv_cache_destroy_cb_t* destroy_cb**

The destruction function for cache entries

***lv_cache_get_cb_t* get_cb**

The get function for cache entries

***lv_cache_add_cb_t* add_cb**

The add function for cache entries

***lv_cache_remove_cb_t* remove_cb**

The remove function for cache entries

***lv_cache_drop_cb_t* drop_cb**

The drop function for cache entries

***lv_cache_drop_all_cb_t* drop_all_cb**

The drop all function for cache entries

***lv_cache_get_victim_cb* get_victim_cb**

The get victim function for cache entries

***lv_cache_reserve_cond_cb* reserve_cond_cb**

The reserve condition function for cache entries

***lv_cache_iter_create_cb* iter_create_cb**

The iterator creation function for cache entries

```
struct _lv_cache_slot_size_t
#include <lv_cache_private.h> Cache entry slot struct
```

To add new fields to the cache entry, add them to a new struct and add it to the first field of the cache data struct. And this one is a size slot for the cache entry.

Membri Pubblici

size_t size

class

lv_cache_class.h

lv_cache_lru_ll.h

Variabili

const *lv_cache_class_t* **lv_cache_class_lru_ll_count**

const *lv_cache_class_t* **lv_cache_class_lru_ll_size**

lv_cache_lru_rb.h

Variabili

const *lv_cache_class_t* **lv_cache_class_lru_rb_count**

const *lv_cache_class_t* **lv_cache_class_lru_rb_size**

lv_cache_sc_da.h

Variabili

const *lv_cache_class_t* **lv_cache_class_sc_da**

instance

lv_cache_instance.h

lv_image_cache.h

Funzioni

lv_result_t lv_image_cache_init(uint32_t size)

Initialize image cache.

Parametri

- size** -- size of the cache in bytes.

Ritorna

LV_RESULT_OK: initialization succeeded, LV_RESULT_INVALID: failed.

void lv_image_cache_resize(uint32_t new_size, bool evict_now)

Resize image cache. If set to 0, the cache will be disabled.

Parametri

- **new_size** -- new size of the cache in bytes.

- **evict_now** -- true: evict the images should be removed by the eviction policy, false: wait for the next cache cleanup.

void lv_image_cache_drop(const void *src)

Invalidate image cache. Use NULL to invalidate all images.

Parametri

src -- pointer to an image source.

bool lv_image_cache_is_enabled(void)

Return true if the image cache is enabled.

Ritorna

true: enabled, false: disabled.

lv_iter_t *lv_image_cache_iter_create(void)

Create an iterator to iterate over the image cache.

Ritorna

an iterator to iterate over the image cache.

void lv_image_cache_dump(void)

Dump the content of the image cache in a human-readable format with cache order.

lv_image_header_cache.h

Funzioni

lv_result_t lv_image_header_cache_init(uint32_t count)

Initialize image header cache.

Parametri

count -- initial size of the cache in count of image headers.

Ritorna

LV_RESULT_OK: initialization succeeded, LV_RESULT_INVALID: failed.

void lv_image_header_cache_resize(uint32_t count, bool evict_now)

Resize image header cache. If set to 0, the cache is disabled.

Parametri

- **count** -- new max count of cached image headers.
- **evict_now** -- true: evict the image headers should be removed by the eviction policy, false: wait for the next cache cleanup.

void lv_image_header_cache_drop(const void *src)

Invalidate image header cache. Use NULL to invalidate all image headers. It's also automatically called when an image is invalidated.

Parametri

src -- pointer to an image source.

bool lv_image_header_cache_is_enabled(void)

Return true if the image header cache is enabled.

Ritorna

true: enabled, false: disabled.

lv_iter_t *lv_image_header_cache_iter_create(void)

Create an iterator to iterate over the image header cache.

Ritorna

an iterator to iterate over the image header cache.

void lv_image_header_cache_dump(void)

Dump the content of the image header cache in a human-readable format with cache order.

14.22 osal

14.22.1 lv_linux.h

14.22.2 lv_os.h

Funzioni

void **lv_lock**(void)

Lock LVGL's general mutex. LVGL is not thread safe, so a mutex is used to avoid executing multiple LVGL functions at the same time from different threads. It shall be called when calling LVGL functions from threads different than `lv_timer_handler`'s thread. It doesn't need to be called in LVGL events because they are called from `lv_timer_handler()`. It is called internally in `lv_timer_handler()`.

lv_result_t lv_lock_isr(void)

Same as `lv_lock()` but can be called from an interrupt.

Ritorna

LV_RESULT_OK: success; LV_RESULT_INVALID: failure

void **lv_unlock**(void)

The pair of `lv_lock()` and `lv_lock_isr()`. It unlocks LVGL general mutex. It is called internally in `lv_timer_handler()`.

void **lv_sleep_ms**(uint32_t ms)

Sleeps the current thread by an amount of milliseconds.

Parametri

`ms` -- amount of milliseconds to sleep the current thread.

14.22.3 lv_os_none.h

Typedef

typedef int **lv_mutex_t**

typedef int **lv_thread_t**

typedef int **lv_thread_sync_t**

14.22.4 lv_os_private.h

Enum

enum **lv_thread_prio_t**

Values:

enumerator **LV_THREAD_PRIO_LOWEST**

enumerator **LV_THREAD_PRIO_LOW**

enumerator **LV_THREAD_PRIO_MID**

enumerator **LV_THREAD_PRIO_HIGH**

enumerator **LV_THREAD_PRIO_HIGHEST**

Funzioni

```
void lv_os_init(void)
    Initialize the OS layer

uint32_t lv_os_get_idle_percent(void)
    Set it for LV_SYSMON_GET_IDLE to show the CPU usage
    Ritorna
        the idle percentage since the last call

static inline lv_result_t lv_thread_init(lv_thread_t *thread, const char *const name, lv_thread_prio_t prio, void
                                         (*callback)(void*), size_t stack_size, void *user_data)

static inline lv_result_t lv_thread_delete(lv_thread_t *thread)

static inline lv_result_t lv_mutex_init(lv_mutex_t *mutex)

static inline lv_result_t lv_mutex_lock(lv_mutex_t *mutex)

static inline lv_result_t lv_mutex_lock_isr(lv_mutex_t *mutex)

static inline lv_result_t lv_mutex_unlock(lv_mutex_t *mutex)

static inline lv_result_t lv_mutex_delete(lv_mutex_t *mutex)

static inline lv_result_t lv_thread_sync_init(lv_thread_sync_t *sync)

static inline lv_result_t lv_thread_sync_wait(lv_thread_sync_t *sync)

static inline lv_result_t lv_thread_sync_signal(lv_thread_sync_t *sync)

static inline lv_result_t lv_thread_sync_signal_isr(lv_thread_sync_t *sync)

static inline lv_result_t lv_thread_sync_delete(lv_thread_sync_t *sync)
```

14.23 others

14.23.1 file_explorer

lv_file_explorer.h

Enum

enum **lv_file_explorer_sort_t**

Values:

enumerator **LV_EXPLORER_SORT_NONE**

enumerator **LV_EXPLORER_SORT_KIND**

enum **lv_file_explorer_dir_t**

Values:

enumerator **LV_EXPLORER_HOME_DIR**

enumerator **LV_EXPLORER_MUSIC_DIR**

enumerator **LV_EXPLORER_PICTURES_DIR**

enumerator **LV_EXPLORER_VIDEO_DIR**

enumerator **LV_EXPLORER_DOCS_DIR**

enumerator **LV_EXPLORER_FS_DIR**

Funzioni

`lv_obj_t *lv_file_explorer_create(lv_obj_t *parent)`

void **lv_file_explorer_set_quick_access_path**(`lv_obj_t *obj, lv_file_explorer_dir_t dir, const char *path`)

Set file_explorer

Parametri

- **obj** -- pointer to a label object
- **dir** -- the dir from 'lv_file_explorer_dir_t' enum.
- **path** -- path

`void lv_file_explorer_set_sort(lv_obj_t *obj, lv_file_explorer_sort_t sort)`

Set file_explorer sort

Parametri

- **obj** -- pointer to a file explorer object
- **sort** -- the sort from 'lv_file_explorer_sort_t' enum.

`const char *lv_file_explorer_get_selected_file_name(const lv_obj_t *obj)`

Get file explorer Selected file

Parametri

obj -- pointer to a file explorer object

Ritorna

pointer to the file explorer selected file name

`const char *lv_file_explorer_get_current_path(const lv_obj_t *obj)`

Get file explorer cur path

Parametri

obj -- pointer to a file explorer object

Ritorna

pointer to the file explorer cur path

`lv_obj_t *lv_file_explorer_get_file_table(lv_obj_t *obj)`

Get file explorer file list obj(lv_table)

Parametri

obj -- pointer to a file explorer object

Ritorna

pointer to the file explorer file table obj(lv_table)

`lv_obj_t *lv_file_explorer_get_header(lv_obj_t *obj)`

Get file explorer head area obj

Parametri

obj -- pointer to a file explorer object

Ritorna

pointer to the file explorer head area obj(lv_obj)

`lv_obj_t *lv_file_explorer_get_path_label(lv_obj_t *obj)`

Get file explorer path obj(label)

Parametri

obj -- pointer to a file explorer object

Ritorna

pointer to the file explorer path obj(lv_label)

`lv_obj_t *lv_file_explorer_get_quick_access_area(lv_obj_t *obj)`

Get file explorer head area obj

Parametri

obj -- pointer to a file explorer object

Ritorna

pointer to the file explorer quick access area obj(lv_obj)

`lv_obj_t *lv_file_explorer_get_places_list(lv_obj_t *obj)`

Get file explorer places list obj(lv_list)

Parametri

obj -- pointer to a file explorer object

Ritorna

pointer to the file explorer places list obj(lv_list)

`lv_obj_t *lv_file_explorer_get_device_list(lv_obj_t *obj)`

Get file explorer device list obj(lv_list)

Parametri

obj -- pointer to a file explorer object

Ritorna

pointer to the file explorer device list obj(lv_list)

`lv_file_explorer_sort_t lv_file_explorer_get_sort(const lv_obj_t *obj)`

Set file_explorer sort

Parametri

obj -- pointer to a file explorer object

Ritorna

the current mode from 'lv_file_explorer_sort_t'

`void lv_file_explorer_open_dir(lv_obj_t *obj, const char *dir)`

Open a specified path

Parametri

- **obj** -- pointer to a file explorer object
- **dir** -- pointer to the path

Variabili

`const lv_obj_class_t lv_file_explorer_class`

lv_file_explorer_private.h**Enum**

`enum lv_file_explorer_file_kind_t`

Values:

enumerator `LV_FILE_EXPLORER_FILE_KIND_DIR`

enumerator `LV_FILE_EXPLORER_FILE_KIND_IMAGE`

enumerator `LV_FILE_EXPLORER_FILE_KIND_AUDIO`

enumerator `LV_FILE_EXPLORER_FILE_KIND_VIDEO`

enumerator **LV_FILE_EXPLORER_FILE_KIND_FILE**

struct **_lv_file_explorer_t**

Membri Pubblici

lv_obj_t **obj**

lv_obj_t ***cont**

lv_obj_t ***head_area**

lv_obj_t ***browser_area**

lv_obj_t ***file_table**

lv_obj_t ***path_label**

lv_obj_t ***quick_access_area**

lv_obj_t ***list_device**

lv_obj_t ***list_places**

char ***home_dir**

char ***music_dir**

char ***pictures_dir**

char ***video_dir**

char ***docs_dir**

char ***fs_dir**

const char ***sel_fn**

char **current_path**[(128)]

lv_file_explorer_sort_t **sort**

struct **lv_file_explorer_file_table_entry_data_t**

Membri Pubblici

`lv_file_explorer_file_kind_t file_kind`

14.23.2 fragment**lv_fragment.h**

Public header for Fragment

Typedef

`typedef struct _lv_fragment_manager_t lv_fragment_manager_t`

Funzioni

`lv_fragment_manager_t *lv_fragment_manager_create(lv_fragment_t *parent)`

Create fragment manager instance

Parametri

`parent` -- Parent fragment if this manager is placed inside another fragment, can be null.

Ritorna

Fragment manager instance

`void lv_fragment_manager_delete(lv_fragment_manager_t *manager)`

Destroy fragment manager instance

Parametri

`manager` -- Fragment manager instance

`void lv_fragment_manager_create_obj(lv_fragment_manager_t *manager)`

Create object of all fragments managed by this manager.

Parametri

`manager` -- Fragment manager instance

`void lv_fragment_manager_delete_obj(lv_fragment_manager_t *manager)`

Delete object created by all fragments managed by this manager. Instance of fragments will not be deleted.

Parametri

`manager` -- Fragment manager instance

`void lv_fragment_manager_add(lv_fragment_manager_t *manager, lv_fragment_t *fragment, lv_obj_t *const *container)`

Attach fragment to manager, and add to container.

Parametri

- `manager` -- Fragment manager instance
- `fragment` -- Fragment instance
- `container` -- Pointer to container object for manager to add objects to

`void lv_fragment_manager_remove(lv_fragment_manager_t *manager, lv_fragment_t *fragment)`

Detach and destroy fragment. If fragment is in navigation stack, remove from it.

Parametri

- `manager` -- Fragment manager instance
- `fragment` -- Fragment instance

`void lv_fragment_manager_push(lv_fragment_manager_t *manager, lv_fragment_t *fragment, lv_obj_t *const *container)`

Attach fragment to manager and add to navigation stack.

Parametri

- `manager` -- Fragment manager instance
- `fragment` -- Fragment instance
- `container` -- Pointer to container object for manager to add objects to

`bool lv_fragment_manager_pop(lv_fragment_manager_t *manager)`

Remove the top-most fragment for stack

Parametri

- **manager** -- Fragment manager instance

Ritorna

true if there is fragment to pop

`void lv_fragment_manager_replace(lv_fragment_manager_t *manager, lv_fragment_t *fragment, lv_obj_t *const *container)`

Replace fragment. Old item in the stack will be removed.

Parametri

- **manager** -- Fragment manager instance
- **fragment** -- Fragment instance
- **container** -- Pointer to container object for manager to add objects to

`bool lv_fragment_manager_send_event(lv_fragment_manager_t *manager, int code, void *userdata)`

Send event to top-most fragment

Parametri

- **manager** -- Fragment manager instance
- **code** -- User-defined ID of event
- **userdata** -- User-defined data

Ritorna

true if fragment returned true

`size_t lv_fragment_manager_get_stack_size(lv_fragment_manager_t *manager)`

Get stack size of this fragment manager

Parametri

- **manager** -- Fragment manager instance

Ritorna

Stack size of this fragment manager

`lv_fragment_t *lv_fragment_manager_get_top(lv_fragment_manager_t *manager)`

Get top most fragment instance

Parametri

- **manager** -- Fragment manager instance

Ritorna

Top most fragment instance

`lv_fragment_t *lv_fragment_manager_find_by_container(lv_fragment_manager_t *manager, const lv_obj_t *container)`

Find first fragment instance in the container

Parametri

- **manager** -- Fragment manager instance
- **container** -- Container which target fragment added to

Ritorna

First fragment instance in the container

`lv_fragment_t *lv_fragment_manager_get_parent_fragment(lv_fragment_manager_t *manager)`

Get parent fragment

Parametri

- **manager** -- Fragment manager instance

Ritorna

Parent fragment instance

`lv_fragment_t *lv_fragment_create(const lv_fragment_class_t *cls, void *args)`

Create a fragment instance.

Parametri

- **cls** -- Fragment class. This fragment must return non null object.
- **args** -- Arguments assigned by fragment manager

Ritorna
 Fragment instance

```
void lv_fragment_delete(lv_fragment_t *fragment)
```

Destroy a fragment.

Parametri

- **fragment** -- Fragment instance.

```
lv_fragment_manager_t *lv_fragment_get_manager(lv_fragment_t *fragment)
```

Get associated manager of this fragment

Parametri

- **fragment** -- Fragment instance

Ritorna
 Fragment manager instance

```
lv_obj_t *const *lv_fragment_get_container(lv_fragment_t *fragment)
```

Get container object of this fragment

Parametri

- **fragment** -- Fragment instance

Ritorna
 Reference to container object

```
lv_fragment_t *lv_fragment_get_parent(lv_fragment_t *fragment)
```

Get parent fragment of this fragment

Parametri

- **fragment** -- Fragment instance

Ritorna
 Parent fragment

```
lv_obj_t *lv_fragment_create_obj(lv_fragment_t *fragment, lv_obj_t *container)
```

Create object by fragment.

Parametri

- **fragment** -- Fragment instance.
- **container** -- Container of the objects should be created upon.

Ritorna
 Created object

```
void lv_fragment_delete_obj(lv_fragment_t *fragment)
```

Delete created object of a fragment

Parametri

- **fragment** -- Fragment instance.

```
void lv_fragment_recreate_obj(lv_fragment_t *fragment)
```

Destroy obj in fragment, and recreate them.

Parametri

- **fragment** -- Fragment instance

```
struct _lv_fragment_t
```

Membri Pubblici

```
const lv_fragment_class_t *cls
```

Class of this fragment

```
lv_fragment_managed_states_t *managed
```

Managed fragment states. If not null, then this fragment is managed.

 **Avvertimento**

Don't modify values inside this struct!

`lv_fragment_manager_t *child_manager`

Child fragment manager

`lv_obj_t *obj`

lv_obj returned by create_obj_cb

`struct _lv_fragment_class_t`

Membri Pubblici

`void (*constructor_cb)(lv_fragment_t *self, void *args)`

Constructor function for fragment class

Param self

Fragment instance

Param args

Arguments assigned by fragment manager

`void (*destructor_cb)(lv_fragment_t *self)`

Destructor function for fragment class

Param self

Fragment instance, will be freed after this call

`void (*attached_cb)(lv_fragment_t *self)`

Fragment attached to manager

Param self

Fragment instance

`void (*detached_cb)(lv_fragment_t *self)`

Fragment detached from manager

Param self

Fragment instance

`lv_obj_t *(*create_obj_cb)(lv_fragment_t *self, lv_obj_t *container)`

Create objects

Param self

Fragment instance

Param container

Container of the objects should be created upon

Return

Created object, NULL if multiple objects has been created

`void (*obj_created_cb)(lv_fragment_t *self, lv_obj_t *obj)`

Param self

Fragment instance

Param obj

lv_obj returned by create_obj_cb

```
void (*obj_will_delete_cb)(lv_fragment_t *self, lv_obj_t *obj)
```

Called before objects in the fragment will be deleted.

Param self

Fragment instance

Param obj

object with this fragment

```
void (*obj_deleted_cb)(lv_fragment_t *self, lv_obj_t *obj)
```

Called when the object created by fragment received LV_EVENT_DELETE event

Param self

Fragment instance

Param obj

object with this fragment

```
bool (*event_cb)(lv_fragment_t *self, int code, void *userdata)
```

Handle event

Param self

Fragment instance

Param which

User-defined ID of event

Param data1

User-defined data

Param data2

User-defined data

size_t instance_size

REQUIRED: Allocation size of fragment

lv_fragment_private.h

```
struct _lv_fragment_managed_states_t
```

```
#include <lv_fragment_private.h> Fragment states
```

Membri Pubblici

const lv_fragment_class_t *cls

Class of the fragment

lv_fragment_manager_t *manager

Manager the fragment attached to

lv_obj_t *const *container

Container object the fragment adding view to

lv_fragment_t *instance

Fragment instance

bool obj_created

true between create_obj_cb and obj_deleted_cb

bool `destroying_obj`

true before `lv_fragment_delete_obj` is called. Don't touch any object if this is true

bool `in_stack`

true if this fragment is in navigation stack that can be popped

14.23.3 translation

lv_translation.h

Funzioni

void `lv_translation_init(void)`

Initialize the translation module

void `lv_translation_deinit(void)`

De-initialize the translation module and free all allocated translations

`lv_translation_pack_t *lv_translation_add_static(const char *const languages[], const char *const tags[], const char *const translations[])`

Register a translation pack from static arrays. All the pointers need to be static, that is to live while they are used

Parametri

- **languages** -- List of languages. E.g. {"en", "de", NULL}
- **tags** -- Tags that are using in the UI. E.g. {"dog", "cat", NULL}
- **translations** -- List of translations. E.g. {"Dog", "Cat", "Hund", "Katze"}

Ritorna

The created pack

`lv_translation_pack_t *lv_translation_add_dynamic(void)`

Add a pack to which translations can be added dynamically. `pack->languages` needs to be a malloc-ed array where each language is also malloc-ed as an element. `pack->translation_array` stores the translation having `lv_translation_tag_dsc_t` items In each array element `tag` is a malloced string, `translations` is a malloc-ed array with malloc-ed array for each element.

Ritorna

the created pack to which data can be added manually.

void `lv_translation_set_language(const char *lang)`

Select the current language The `LV_EVENT_TRANSLATION_LANGUAGE_CHANGED` event will be sent to every widget

Parametri

`lang` -- a string from the defined languages. E.g. "en" or "de"

`const char *lv_translation_get_language(void)`

Get the current selected language

Ritorna

the current selected language

`const char *lv_translation_get(const char *tag)`

Get the translated version of a tag on the selected language

Nota

fallback rules:

- if the tag is found on the selected language return it
- if the tag is not found on the selected language, use the first language
- if the tag is not found on the first language, return the tag

Parametri

- tag** -- the tag to translate

Ritorna

the translation

```
static inline const char *lv_tr(const char *tag)
```

Shorthand of lv_translation_set_language

Parametri

- tag** -- the tag to translate

Ritorna

the translation

```
lv_result_t lv_translation_add_language(lv_translation_pack_t *pack, const char *lang)
```

Add a new language to a dynamic language pack. All languages should be added before adding tags

Parametri

- **pack** -- pointer to a dynamic translation pack
- **lang** -- language to add, e.g. "en", or "de"

Ritorna

LV_RESULT_OK: success, LV_RESULT_INVALID: failed

```
int32_t lv_translation_get_language_index(lv_translation_pack_t *pack, const char *lang_name)
```

Get the index of a language in a pack.

Parametri

- **pack** -- pointer to a static or dynamic language pack
- **lang_name** -- name of the language to find

Ritorna

index of the language or -1 if not found.

```
lv_translation_tag_dsc_t *lv_translation_add_tag(lv_translation_pack_t *pack, const char *tag_name)
```

Add a new tag to a dynamic language pack. Once the tag is added the translations for each language can be added too by using lv_translation_set_tag_translation

Parametri

- **pack** -- pointer to a dynamic translation pack
- **tag_name** -- name of the tag, e.g. "dog", or "house"

Ritorna

pointer to the allocated tag descriptor

```
lv_result_t lv_translation_set_tag_translation(lv_translation_pack_t *pack, lv_translation_tag_dsc_t *tag,
                                              uint32_t lang_idx, const char *trans)
```

Add a translation to a tag in a dynamic translation pack

Parametri

- **pack** -- pointer to a dynamic translation pack
- **tag** -- return value of lv_translation_add_tag
- **lang_idx** -- index of the language for which translation should be set
- **trans** -- the translation on the given language

Ritorna

LV_RESULT_OK: success, LV_RESULT_INVALID: failed

lv_translation_private.h

```
struct _lv_translation_tag_dsc_t
```

Membri Pubblici

const char *tag

const char **translations

Translations for each language

```
struct _lv_translation_pack_t
```

Membri Pubblici

```
const char **languages
uint32_t language_cnt
uint32_t is_static
const char **tag_p
const char **translation_p
lv_array_t translation_array
```

14.24 stdlib

14.24.1 lv_mem.h

Typedef

```
typedef void *lv_mem_pool_t
```

Funzioni

```
void lv_mem_init(void)
```

Initialize to use malloc/free/realloc etc

```
void lv_mem_deinit(void)
```

Drop all dynamically allocated memory and reset the memory pools' state

```
lv_mem_pool_t lv_mem_add_pool(void *mem, size_t bytes)
```

```
void lv_mem_remove_pool(lv_mem_pool_t pool)
```

```
void *lv_malloc(size_t size)
```

Allocate memory dynamically

Parametri

- **size** -- requested size in bytes

Ritorna

pointer to allocated uninitialized memory, or NULL on failure

```
void *lv_calloc(size_t num, size_t size)
```

Allocate a block of zeroed memory dynamically

Parametri

- **num** -- requested number of element to be allocated.
- **size** -- requested size of each element in bytes.

Ritorna

pointer to allocated zeroed memory, or NULL on failure

```
void *lv_zalloc(size_t size)
Allocate zeroed memory dynamically
Parametri
    size -- requested size in bytes
Ritorna
    pointer to allocated zeroed memory, or NULL on failure

void *lv_malloc_zeroed(size_t size)
Allocate zeroed memory dynamically
Parametri
    size -- requested size in bytes
Ritorna
    pointer to allocated zeroed memory, or NULL on failure

void lv_free(void *data)
Free an allocated data
Parametri
    data -- pointer to an allocated memory

void *lv_realloc(void *data_p, size_t new_size)
Reallocate a memory with a new size. The old content will be kept.
Parametri
    • data_p -- pointer to an allocated memory. Its content will be copied to the new
      memory block and freed
    • new_size -- the desired new size in byte
Ritorna
    pointer to the new memory, NULL on failure

void *lv_reallocf(void *data_p, size_t new_size)
Reallocate a memory with a new size. The old content will be kept. In case of failure, the old pointer is free'd.
Parametri
    • data_p -- pointer to an allocated memory. Its content will be copied to the new
      memory block and freed
    • new_size -- the desired new size in byte
Ritorna
    pointer to the new memory, NULL on failure

void *lv_malloc_core(size_t size)
Used internally to execute a plain malloc operation
Parametri
    size -- size in bytes to malloc

void lv_free_core(void *p)
Used internally to execute a plain free operation
Parametri
    p -- memory address to free

void *lv_realloc_core(void *p, size_t new_size)
Used internally to execute a plain realloc operation
Parametri
    • p -- memory address to realloc
    • new_size -- size in bytes to realloc

void lv_mem_monitor_core(lv_mem_monitor_t *mon_p)
Used internally by lv_mem_monitor() to gather LVGL heap state information.
Parametri
    mon_p -- pointer to lv_mem_monitor_t object to be populated.

lv_result_t lv_mem_test_core(void)
```

```
lv_result_t lv_mem_test(void)
```

Tests the memory allocation system by allocating and freeing a block of memory.

Ritorna

LV_RESULT_OK if the memory allocation system is working properly, or LV_RESULT_INVALID if there is an error.

```
void lv_mem_monitor(lv_mem_monitor_t *mon_p)
```

Give information about the work memory of dynamic allocation

Parametri

mon_p -- pointer to a *lv_mem_monitor_t* variable, the result of the analysis will be stored here

```
struct lv_mem_monitor_t
```

#include <lv_mem.h> Heap information structure.

Membri Pubblici

size_t total_size

Total heap size

size_t free_cnt

size_t free_size

Size of available memory

size_t free_biggest_size

size_t used_cnt

size_t max_used

Max size of Heap memory used

uint8_t used_pct

Percentage used

uint8_t frag_pct

Amount of fragmentation

14.24.2 lv_mem_private.h

14.24.3 lv_sprintf.h

Define

LV_PRIId32

LV_PRIu32

LV_PRIx32

LV_PRIX32

LV_PRId64**LV_PRIu64****LV_PRIx64**

Funzioni

`int lv_snprintf(char *buffer, size_t count, const char *format, ...)``int lv_vsnprintf(char *buffer, size_t count, const char *format, va_list va)`

14.24.4 lv_string.h

Funzioni

`void *lv_memcpy(void *dst, const void *src, size_t len)`

Copies a block of memory from a source address to a destination address.

Nota

The function does not check for any overlapping of the source and destination memory blocks.

Parametri

- **dst** -- Pointer to the destination array where the content is to be copied.
- **src** -- Pointer to the source of data to be copied.
- **len** -- Number of bytes to copy.

Ritorna

Pointer to the destination array.

`void lv_memset(void *dst, uint8_t v, size_t len)`

Fills a block of memory with a specified value.

Parametri

- **dst** -- Pointer to the destination array to fill with the specified value.
- **v** -- Value to be set. The value is passed as an int, but the function fills the block of memory using the unsigned char conversion of this value.
- **len** -- Number of bytes to be set to the value.

`void *lv_memmove(void *dst, const void *src, size_t len)`

Move a block of memory from source to destination.

Parametri

- **dst** -- Pointer to the destination array where the content is to be copied.
- **src** -- Pointer to the source of data to be copied.
- **len** -- Number of bytes to copy

Ritorna

Pointer to the destination array.

`int lv_memcmp(const void *p1, const void *p2, size_t len)`

This function will compare two memory blocks.

Parametri

- **p1** -- Pointer to the first memory block
- **p2** -- Pointer to the second memory block
- **len** -- Number of bytes to compare

Ritorna

The difference between the value of the first unmatched byte.

`static inline void lv_memzero(void *dst, size_t len)`

Same as `memset(dst, 0x00, len)`.

Parametri

- **dst** -- pointer to the destination buffer
- **len** -- number of bytes to set

`size_t lv_strlen(const char *str)`

Computes the length of the string str up to (but not including) the terminating null character.

Parametri

str -- Pointer to the null-terminated byte string to be examined.

Ritorna

The length of the string in bytes.

`size_t lv_strnlen(const char *str, size_t max_len)`

Computes the length of the string str up to (but not including) the terminating null character, or the given maximum length.

Parametri

- **str** -- Pointer to byte string that is null-terminated or at least `max_len` bytes long.
- **max_len** -- Maximum number of characters to examine.

Ritorna

The length of the string in bytes.

`size_t lv_strlcpy(char *dst, const char *src, size_t dst_size)`

Copies up to `dst_size`-1 (non-null) characters from `src` to `dst`. A null terminator is always added.

Parametri

- **dst** -- Pointer to the destination array where the content is to be copied.
- **src** -- Pointer to the source of data to be copied.
- **dst_size** -- Maximum number of characters to be copied to `dst`, including the null character.

Ritorna

The length of `src`. The return value is equivalent to the value returned by `lv_strlen(src)`

`char *lv_strncpy(char *dst, const char *src, size_t dest_size)`

Copies up to `dest_size` characters from the string pointed to by `src` to the character array pointed to by `dst` and fills the remaining length with null bytes.

 **Nota**

`dst` will not be null terminated if `dest_size` bytes were copied from `src` before the end of `src` was reached.

Parametri

- **dst** -- Pointer to the destination array where the content is to be copied.
- **src** -- Pointer to the source of data to be copied.
- **dest_size** -- Maximum number of characters to be copied to `dst`.

Ritorna

A pointer to the destination array, which is `dst`.

`char *lv_strcpy(char *dst, const char *src)`

Copies the string pointed to by `src`, including the terminating null character, to the character array pointed to by `dst`.

Parametri

- **dst** -- Pointer to the destination array where the content is to be copied.
- **src** -- Pointer to the source of data to be copied.

Ritorna

A pointer to the destination array, which is `dst`.

`int lv_strcmp(const char *s1, const char *s2)`

This function will compare two strings without specified length.

Parametri

- **s1** -- pointer to the first string
- **s2** -- pointer to the second string

Ritorna

the difference between the value of the first unmatching character.

`int lv_strncmp(const char *s1, const char *s2, size_t len)`

This function will compare two strings up to the given length.

Parametri

- **s1** -- pointer to the first string
- **s2** -- pointer to the second string
- **len** -- the maximum amount of characters to compare

Ritorna

the difference between the value of the first unmatching character.

`static inline bool lv_streq(const char *s1, const char *s2)`

Returns true if the two strings are equal. Just a wrapper around strcmp for convenience.

Parametri

- **s1** -- pointer to the first string
- **s2** -- pointer to the second string

Ritorna

true: the strings are equal; false: otherwise

`char *lv_strdup(const char *src)`

Duplicate a string by allocating a new one and copying the content.

Parametri

- **src** -- Pointer to the source of data to be copied.

Ritorna

A pointer to the new allocated string. NULL if failed.

`char *lv_strndup(const char *src, size_t max_len)`

Duplicate a string by allocating a new one and copying the content up to the end or the specified maximum length, whichever comes first.

Parametri

- **src** -- Pointer to the source of data to be copied.
- **max_len** -- Maximum number of characters to be copied.

Ritorna

Pointer to a newly allocated null-terminated string. NULL if failed.

`char *lv_strcat(char *dst, const char *src)`

Copies the string pointed to by src, including the terminating null character, to the end of the string pointed to by dst.

Parametri

- **dst** -- Pointer to the destination string where the content is to be appended.
- **src** -- Pointer to the source of data to be copied.

Ritorna

A pointer to the destination string, which is dst.

`char *lv_strncat(char *dst, const char *src, size_t src_len)`

Copies up to **src_len** characters from the string pointed to by src to the end of the string pointed to by dst. A terminating null character is appended to dst even if no null character was encountered in src after **src_len** characters were copied.

Parametri

- **dst** -- Pointer to the destination string where the content is to be appended.
- **src** -- Pointer to the source of data to be copied.
- **src_len** -- Maximum number of characters from src to be copied to the end of dst.

Ritorna

A pointer to the destination string, which is dst.

`char *lv_strchr(const char *str, int c)`

Searches for the first occurrence of character c in the string str.

Parametri

- **str** -- Pointer to the null-terminated byte string to be searched.
- **c** -- The character to be searched for.

Ritorna

A pointer to the first occurrence of character c in the string str, or a null pointer if c is not found.

14.24.5 builtin

lv_tlsf.h

Typedef

`typedef void *lv_tlsf_t`

`typedef void *lv_pool_t`

`typedef void (*lv_tlsf_walker)(void *ptr, size_t size, int used, void *user)`

Funzioni

`lv_tlsf_t lv_tlsf_create(void *mem)`

`lv_tlsf_t lv_tlsf_create_with_pool(void *mem, size_t bytes)`

`void lv_tlsf_destroy(lv_tlsf_t tlsf)`

`lv_pool_t lv_tlsf_get_pool(lv_tlsf_t tlsf)`

`lv_pool_t lv_tlsf_add_pool(lv_tlsf_t tlsf, void *mem, size_t bytes)`

`void lv_tlsf_remove_pool(lv_tlsf_t tlsf, lv_pool_t pool)`

`void *lv_tlsf_malloc(lv_tlsf_t tlsf, size_t bytes)`

`void *lv_tlsf_memalign(lv_tlsf_t tlsf, size_t align, size_t bytes)`

`void *lv_tlsf_realloc(lv_tlsf_t tlsf, void *ptr, size_t size)`

`size_t lv_tlsf_free(lv_tlsf_t tlsf, const void *ptr)`

`size_t lv_tlsf_block_size(void *ptr)`

`size_t lv_tlsf_size(void)`

`size_t lv_tlsf_align_size(void)`

`size_t lv_tlsf_block_size_min(void)`

`size_t lv_tlsf_block_size_max(void)`

`size_t lv_tlsf_pool_overhead(void)`

`size_t lv_tlsf_alloc_overhead(void)`

`void lv_tlsf_walk_pool(lv_pool_t pool, lv_tlsf_walker walker, void *user)`

`int lv_tlsf_check(lv_tlsf_t tlsf)`

`int lv_tlsf_check_pool(lv_pool_t pool)`

lv_tlsf_private.h

```
struct lv_tlsf_state_t
```

Membri Pubblici

lv_tlsf_t **tlsf**

size_t **cur_used**

size_t **max_used**

lv_ll_t **pool_ll**

14.25 themes

14.25.1 lv_theme.h

Typedef

```
typedef void (*lv_theme_apply_cb_t)(lv_theme_t*, lv_obj_t*)
```

Funzioni

lv_theme_t *lv_theme_get_from_obj(lv_obj_t *obj)

Get the theme assigned to the display of the object

Parametri

obj -- pointer to a theme object

Ritorna

the theme of the object's display (can be NULL)

void lv_theme_apply(lv_obj_t *obj)

Apply the active theme on an object

Parametri

obj -- pointer to an object

void lv_theme_set_parent(lv_theme_t *new_theme, lv_theme_t *parent)

Set a base theme for a theme. The styles from the base them will be added before the styles of the current theme. Arbitrary long chain of themes can be created by setting base themes.

Parametri

- **new_theme** -- pointer to theme which base should be set
- **parent** -- pointer to the base theme

void lv_theme_set_apply_cb(lv_theme_t *theme, lv_theme_apply_cb_t apply_cb)

Set an apply callback for a theme. The apply callback is used to add styles to different objects

Parametri

- **theme** -- pointer to theme which callback should be set
- **apply_cb** -- pointer to the callback

const lv_font_t *lv_theme_get_font_small(lv_obj_t *obj)

Get the small font of the theme

Parametri

obj -- pointer to an object

Ritorna

pointer to the font

`const lv_font_t *lv_theme_get_font_normal(lv_obj_t *obj)`

Get the normal font of the theme

Parametri

`obj` -- pointer to an object

Ritorna

pointer to the font

`const lv_font_t *lv_theme_get_font_large(lv_obj_t *obj)`

Get the subtitle font of the theme

Parametri

`obj` -- pointer to an object

Ritorna

pointer to the font

`lv_color_t lv_theme_get_color_primary(lv_obj_t *obj)`

Get the primary color of the theme

Parametri

`obj` -- pointer to an object

Ritorna

the color

`lv_color_t lv_theme_get_color_secondary(lv_obj_t *obj)`

Get the secondary color of the theme

Parametri

`obj` -- pointer to an object

Ritorna

the color

14.25.2 lv_theme_private.h

`struct _lv_theme_t`

Membri Pubblici

`lv_theme_apply_cb_t apply_cb`

`lv_theme_t *parent`

Apply the current theme's style on top of this theme.

`void *user_data`

`lv_display_t *disp`

`lv_color_t color_primary`

`lv_color_t color_secondary`

`const lv_font_t *font_small`

`const lv_font_t *font_normal`

`const lv_font_t *font_large`

`uint32_t flags`

Any custom flag used by the theme

14.25.3 default

lv_theme_default.h

Funzioni

`lv_theme_t *lv_theme_default_init(lv_display_t *disp, lv_color_t color_primary, lv_color_t color_secondary, bool dark, const lv_font_t *font)`

Initialize the theme

Parametri

- `disp` -- pointer to display
- `color_primary` -- the primary color of the theme
- `color_secondary` -- the secondary color for the theme
- `dark` --
- `font` -- pointer to a font to use.

Ritorna

a pointer to reference this theme later

`bool lv_theme_default_is_inited(void)`

Check if default theme is initialized

Ritorna

true if default theme is initialized, false otherwise

`lv_theme_t *lv_theme_default_get(void)`

Get default theme

Ritorna

a pointer to default theme, or NULL if this is not initialized

`void lv_theme_default_deinit(void)`

Deinitialize the default theme

14.25.4 mono

lv_theme_mono.h

Funzioni

`lv_theme_t *lv_theme_mono_init(lv_display_t *disp, bool dark_bg, const lv_font_t *font)`

Initialize the theme

Parametri

- `disp` -- pointer to display
- `dark_bg` --
- `font` -- pointer to a font to use.

Ritorna

a pointer to reference this theme later

`bool lv_theme_mono_is_inited(void)`

Check if the theme is initialized

Ritorna

true if default theme is initialized, false otherwise

`lv_theme_t *lv_theme_mono_get(void)`

Get mono theme

Ritorna

a pointer to mono theme, or NULL if this is not initialized

```
void lv_theme_mono_deinit(void)
```

Deinitialize the mono theme

14.25.5 simple

lv_theme_simple.h

Funzioni

```
lv_theme_t *lv_theme_simple_init(lv_display_t *disp)
```

Initialize the theme

Parametri

disp -- pointer to display

Ritorna

a pointer to reference this theme later

```
bool lv_theme_simple_is_inited(void)
```

Check if the theme is initialized

Ritorna

true if default theme is initialized, false otherwise

```
lv_theme_t *lv_theme_simple_get(void)
```

Get simple theme

Ritorna

a pointer to simple theme, or NULL if this is not initialized

```
void lv_theme_simple_deinit(void)
```

Deinitialize the simple theme

14.26 tick

14.26.1 lv_tick.h

Provide access to the system tick with 1 millisecond resolution

Typedef

```
typedef uint32_t (*lv_tick_get_cb_t)(void)
```

```
typedef void (*lv_delay_cb_t)(uint32_t ms)
```

Funzioni

```
void lv_tick_inc(uint32_t tick_period)
```

You have to call this function periodically. It is typically safe to call from an interrupt handler or a different thread.

Parametri

tick_period -- the call period of this function in milliseconds

```
uint32_t lv_tick_get(void)
```

Get the elapsed milliseconds since start up

Ritorna

the elapsed milliseconds

```
uint32_t lv_tick_elaps(uint32_t prev_tick)
```

Get the elapsed milliseconds since a previous time stamp

Parametri

prev_tick -- a previous time stamp (return value of [lv_tick_get\(\)](#))

Ritorna

the elapsed milliseconds since 'prev_tick'

`uint32_t lv_tick_diff(uint32_t tick, uint32_t prev_tick)`

Get the elapsed milliseconds between two time stamps

Parametri

- `tick` -- a time stamp
- `prev_tick` -- a time stamp before `tick`

Ritorna

the elapsed milliseconds between `prev_tick` and `tick`

`void lv_delay_ms(uint32_t ms)`

Delay for the given milliseconds. By default it's a blocking delay, but with `lv_delay_set_cb()` a custom delay function can be set too

Parametri

`ms` -- the number of milliseconds to delay

`void lv_delay_set_cb(lv_delay_cb_t cb)`

Set a callback for a blocking delay

Parametri

`cb` -- pointer to a callback

`void lv_tick_set_cb(lv_tick_get_cb_t cb)`

Set the custom callback for 'lv_tick_get'

Parametri

`cb` -- call this callback on 'lv_tick_get'

`lv_tick_get_cb_t lv_tick_get_cb(void)`

Get the custom callback for 'lv_tick_get'

Ritorna

call this callback on 'lv_tick_get'

14.26.2 lv_tick_private.h

`struct lv_tick_state_t`

Membri Pubblici

`volatile uint32_t sys_time`

`volatile uint8_t sys_irq_flag`

`lv_tick_get_cb_t tick_get_cb`

`lv_delay_cb_t delay_cb`

14.27 widgets

14.27.1 3dtexture

`lv_3dtexture.h`

Funzioni

```
lv_obj_t *lv_3dtexture_create(lv_obj_t *parent)
```

Create a 3dtexture object

Parametri

- parent** -- pointer to an object, it will be the parent of the new 3dtexture

Ritorna

pointer to the created 3dtexture

```
void lv_3dtexture_set_src(lv_obj_t *obj, lv_3dtexture_id_t id)
```

Set the source texture of the widget. The object size should be manually set to match.

Parametri

- obj** -- the 3dtexture widget
- id** -- the texture handle from the 3D graphics backend. I.e., an unsigned int texture for OpenGL.

```
void lv_3dtexture_set_flip(lv_obj_t *obj, bool h_flip, bool v_flip)
```

Set the flipping behavior of the widget.

Parametri

- obj** -- the 3dtexture widget
- h_flip** -- true to flip horizontally.
- v_flip** -- true to flip vertically.

Variabili

```
const lv_obj_class_t lv_3dtexture_class
```

lv_3dtexture_private.h

```
struct _lv_3dtexture_t
```

Membri Pubblici

```
lv_obj_t obj
```

```
lv_3dtexture_id_t id
```

```
bool h_flip
```

```
bool v_flip
```

14.27.2 animimage

lv_animimage.h

Enum

```
enum _lv_property_animimage_id_t
```

Modified by NXP in 2025

Values:

enumerator **LV_PROPERTY_ANIMIMAGE_SRC**

enumerator **LV_PROPERTY_ANIMIMAGE_DURATION**

enumerator **LV_PROPERTY_ANIMIMAGE_REPEAT_COUNT**

enumerator **LV_PROPERTY_ANIMIMAGE_SRC_COUNT**

enumerator **LV_PROPERTY_ANIMIMAGE_END**

enum **lv_animimg_part_t**

Image parts

Values:

enumerator **LV_ANIM_IMAGE_PART_MAIN**

Funzioni

lv_obj_t *lv_animimg_create(lv_obj_t *parent)

Create an animation image objects

Parametri

- **parent** -- pointer to an object, it will be the parent of the new button

Ritorna

pointer to the created animation image object

void lv_animimg_set_src(lv_obj_t *obj, const void *dsc[], size_t num)

Set the image animation images source.

Parametri

- **obj** -- pointer to an animation image object
- **dsc** -- pointer to a series images
- **num** -- images' number

void lv_animimg_set_src_reverse(lv_obj_t *obj, const void *dsc[], size_t num)

Set the images source for flip playback of animation image.

Parametri

- **obj** -- pointer to an animation image object
- **dsc** -- pointer to a series images
- **num** -- images' number

void lv_animimg_start(lv_obj_t *obj)

Startup the image animation.

Parametri

obj -- pointer to an animation image object

bool lv_animimg_delete(lv_obj_t *obj)

Delete the image animation.

Parametri

obj -- pointer to an animation image object

void lv_animimg_set_duration(lv_obj_t *obj, uint32_t duration)

Set the image animation duration time. unit:ms

Parametri

- **obj** -- pointer to an animation image object
- **duration** -- the duration in milliseconds

void lv_animimg_set_repeat_count(lv_obj_t *obj, uint32_t count)

Set the image animation repeatedly play times.

Parametri

- **obj** -- pointer to an animation image object
- **count** -- the number of times to repeat the animation

void lv_animimg_set_reverse_duration(lv_obj_t *obj, uint32_t duration)

Make the image animation to play back to when the forward direction is ready.

Parametri

- **obj** -- pointer to an animation image object
- **duration** -- the duration of the playback image animation in milliseconds. 0: disable playback

void lv_animimg_set_reverse_delay(lv_obj_t *obj, uint32_t duration)

Make the image animation to play back to when the forward direction is ready.

Parametri

- **obj** -- pointer to an animation image object
- **duration** -- delay in milliseconds before starting the playback image animation.

void lv_animimg_set_start_cb(lv_obj_t *obj, lv_anim_start_cb_t start_cb)

Set a function call when the animation image really starts (considering delay)

Parametri

- **obj** -- pointer to an animation image object
- **start_cb** -- a function call when the animation is start

void lv_animimg_set_completed_cb(lv_obj_t *obj, lv_anim_completed_cb_t completed_cb)

Set a function call when the animation is completed

Parametri

- **obj** -- pointer to an animation image object
- **completed_cb** -- a function call when the animation is completed

const void **lv_animimg_get_src(lv_obj_t *obj)

Get the image animation images source.

Parametri

- **obj** -- pointer to an animation image object

Ritorna

a pointer that will point to a series images

uint8_t lv_animimg_get_src_count(lv_obj_t *obj)

Get the image animation images source.

Parametri

- **obj** -- pointer to an animation image object

Ritorna

the number of source images

uint32_t lv_animimg_get_duration(lv_obj_t *obj)

Get the image animation duration time. unit:ms

Parametri

- **obj** -- pointer to an animation image object

Ritorna

the animation duration time

uint32_t lv_animimg_get_repeat_count(lv_obj_t *obj)

Get the image animation repeat play times.

Parametri

- **obj** -- pointer to an animation image object

Ritorna

the repeat count

lv_anim_t *lv_animimg_get_anim(lv_obj_t *obj)

Get the image animation underlying animation.

Parametri

- **obj** -- pointer to an animation image object

Ritorna

the animation reference

Variabili

```
const lv_obj_class_t lv_animimg_class
```

lv_animimage_private.h

```
struct _lv_animimg_t
#include <lv_animimage_private.h> Data of the animimage
```

Membri Pubblici

```
lv_image_t img
```

```
lv_anim_t anim
```

```
const void **dsc
```

```
int8_t pic_count
```

14.27.3 arc

lv_arc.h

Enum

```
enum lv_arc_mode_t
```

In which direction the indicator should grow.

Values:

enumerator **LV_ARC_MODE_NORMAL**

Clock-wise

enumerator **LV_ARC_MODE_SYMMETRICAL**

Left/right from the midpoint

enumerator **LV_ARC_MODE_REVERSE**

Counterclock-wise

Funzioni

```
lv_obj_t *lv_arc_create(lv_obj_t *parent)
```

Create an arc object

Parametri

- **parent** -- pointer to an object, it will be the parent of the new arc

Ritorna

pointer to the created arc

```
void lv_arc_set_start_angle(lv_obj_t *obj, lv_value_precise_t start)
```

Set the start angle of an arc. 0 deg: right, 90 bottom, etc.

Parametri

- **obj** -- pointer to an arc object
- **start** -- the start angle. (if **LV_USE_FLOAT** is enabled it can be fractional too.)

`void lv_arc_set_end_angle(lv_obj_t *obj, lv_value_precise_t end)`

Set the end angle of an arc. 0 deg: right, 90 bottom, etc.

Parametri

- **obj** -- pointer to an arc object
- **end** -- the end angle (if LV_USE_FLOAT is enabled it can be fractional too.)

`void lv_arc_set_angles(lv_obj_t *obj, lv_value_precise_t start, lv_value_precise_t end)`

Set the start and end angles

Parametri

- **obj** -- pointer to an arc object
- **start** -- the start angle (if LV_USE_FLOAT is enabled it can be fractional too.)
- **end** -- the end angle (if LV_USE_FLOAT is enabled it can be fractional too.)

`void lv_arc_set_bg_start_angle(lv_obj_t *obj, lv_value_precise_t start)`

Set the start angle of an arc background. 0 deg: right, 90 bottom, etc.

Parametri

- **obj** -- pointer to an arc object
- **start** -- the start angle (if LV_USE_FLOAT is enabled it can be fractional too.)

`void lv_arc_set_bg_end_angle(lv_obj_t *obj, lv_value_precise_t end)`

Set the start angle of an arc background. 0 deg: right, 90 bottom etc.

Parametri

- **obj** -- pointer to an arc object
- **end** -- the end angle (if LV_USE_FLOAT is enabled it can be fractional too.)

`void lv_arc_set_bg_angles(lv_obj_t *obj, lv_value_precise_t start, lv_value_precise_t end)`

Set the start and end angles of the arc background

Parametri

- **obj** -- pointer to an arc object
- **start** -- the start angle (if LV_USE_FLOAT is enabled it can be fractional too.)
- **end** -- the end angle (if LV_USE_FLOAT is enabled it can be fractional too.)

`void lv_arc_set_rotation(lv_obj_t *obj, int32_t rotation)`

Set the rotation for the whole arc

Parametri

- **obj** -- pointer to an arc object
- **rotation** -- rotation angle

`void lv_arc_set_mode(lv_obj_t *obj, lv_arc_mode_t type)`

Set in which direction the indicator should grow.

Parametri

- **obj** -- pointer to arc object
- **type** -- arc's mode

`void lv_arc_set_value(lv_obj_t *obj, int32_t value)`

Set a new value on the arc

Parametri

- **obj** -- pointer to an arc object
- **value** -- new value

`void lv_arc_set_range(lv_obj_t *obj, int32_t min, int32_t max)`

Set minimum and the maximum values of an arc

Parametri

- **obj** -- pointer to the arc object
- **min** -- minimum value
- **max** -- maximum value

`void lv_arc_set_min_value(lv_obj_t *obj, int32_t min)`

Set the minimum values of an arc

Parametri

- **obj** -- pointer to the arc object
- **min** -- minimum value

void lv_arc_set_max_value(lv_obj_t *obj, int32_t max)

Set the maximum values of an arc

Parametri

- **obj** -- pointer to the arc object
- **max** -- maximum value

void lv_arc_set_change_rate(lv_obj_t *obj, uint32_t rate)

Set a change rate to limit the speed how fast the arc should reach the pressed point.

Parametri

- **obj** -- pointer to an arc object
- **rate** -- the change rate

void lv_arc_set_knob_offset(lv_obj_t *obj, int32_t offset)

Set an offset angle for the knob

Parametri

- **obj** -- pointer to an arc object
- **offset** -- knob offset from main arc in degrees

lv_value_precise_t lv_arc_get_angle_start(lv_obj_t *obj)

Get the start angle of an arc.

Parametri

obj -- pointer to an arc object

Ritorna

the start angle [0..360] (if LV_USE_FLOAT is enabled it can be fractional too.)

lv_value_precise_t lv_arc_get_angle_end(lv_obj_t *obj)

Get the end angle of an arc.

Parametri

obj -- pointer to an arc object

Ritorna

the end angle [0..360] (if LV_USE_FLOAT is enabled it can be fractional too.)

lv_value_precise_t lv_arc_get_bg_angle_start(lv_obj_t *obj)

Get the start angle of an arc background.

Parametri

obj -- pointer to an arc object

Ritorna

the start angle [0..360] (if LV_USE_FLOAT is enabled it can be fractional too.)

lv_value_precise_t lv_arc_get_bg_angle_end(lv_obj_t *obj)

Get the end angle of an arc background.

Parametri

obj -- pointer to an arc object

Ritorna

the end angle [0..360] (if LV_USE_FLOAT is enabled it can be fractional too.)

int32_t lv_arc_get_value(const lv_obj_t *obj)

Get the value of an arc

Parametri

obj -- pointer to an arc object

Ritorna

the value of the arc

int32_t lv_arc_get_min_value(const lv_obj_t *obj)

Get the minimum value of an arc

Parametri

obj -- pointer to an arc object

Ritorna

the minimum value of the arc

```
int32_t lv_arc_get_max_value(const lv_obj_t *obj)
```

Get the maximum value of an arc

Parametri

- obj** -- pointer to an arc object

Ritorna

the maximum value of the arc

```
lv_arc_mode_t lv_arc_get_mode(const lv_obj_t *obj)
```

Get whether the arc is type or not.

Parametri

- obj** -- pointer to an arc object

Ritorna

arc's mode

```
int32_t lv_arc_get_rotation(const lv_obj_t *obj)
```

Get the rotation for the whole arc

Parametri

- obj** -- pointer to an arc object

Ritorna

arc's current rotation

```
int32_t lv_arc_get_knob_offset(const lv_obj_t *obj)
```

Get the current knob angle offset

Parametri

- obj** -- pointer to an arc object

Ritorna

arc's current knob offset

```
lv_observer_t *lv_arc_bind_value(lv_obj_t *obj, lv_subject_t *subject)
```

Bind an integer subject to an Arc's value.

Parametri

- **obj** -- pointer to Arc
- **subject** -- pointer to Subject

Ritorna

pointer to newly-created Observer

```
void lv_arc_align_obj_to_angle(const lv_obj_t *obj, lv_obj_t *obj_to_align, int32_t r_offset)
```

Align an object to the current position of the arc (knob)

Parametri

- **obj** -- pointer to an arc object
- **obj_to_align** -- pointer to an object to align
- **r_offset** -- consider the radius larger with this value (< 0: for smaller radius)

```
void lv_arc_rotate_obj_to_angle(const lv_obj_t *obj, lv_obj_t *obj_to_rotate, int32_t r_offset)
```

Rotate an object to the current position of the arc (knob)

Parametri

- **obj** -- pointer to an arc object
- **obj_to_rotate** -- pointer to an object to rotate
- **r_offset** -- consider the radius larger with this value (< 0: for smaller radius)

Variabili

```
const lv_obj_class_t lv_arc_class
```

lv_arc_private.h

```
struct _lv_arc_t
```

Membri Pubblici

lv_obj_t obj

int32_t rotation

lv_value_precise_t indic_angle_start

lv_value_precise_t indic_angle_end

lv_value_precise_t bg_angle_start

lv_value_precise_t bg_angle_end

int32_t value

Current value of the arc

int32_t min_value

Minimum value of the arc

int32_t max_value

Maximum value of the arc

uint32_t dragging

uint32_t type

uint32_t min_close

1: the last pressed angle was closer to minimum end

uint32_t in_out

1: The click was within the background arc angles. 0: Click outside

uint32_t chg_rate

Drag angle rate of change of the arc (degrees/sec)

uint32_t last_tick

Last dragging event timestamp of the arc

lv_value_precise_t last_angle

Last dragging angle of the arc

int16_t knob_offset

knob offset from the main arc

14.27.4 arclabel

lv_arclabel.h

Define

LV_ARCLABEL_DOT_NUM

LV_ARCLABEL_DEFAULT_TEXT

Enum

enum **lv_arclabel_dir_t**

Values:

enumerator **LV_ARCLABEL_DIR_CLOCKWISE**

enumerator **LV_ARCLABEL_DIR_COUNTER_CLOCKWISE**

enum **lv_arclabel_text_align_t**

Values:

enumerator **LV_ARCLABEL_TEXT_ALIGN_DEFAULT**

enumerator **LV_ARCLABEL_TEXT_ALIGN.LEADING**

enumerator **LV_ARCLABEL_TEXT_ALIGN.CENTER**

enumerator **LV_ARCLABEL_TEXT_ALIGN.TRAILING**

Funzioni

lv_obj_t *lv_arclabel_create(lv_obj_t *parent)

Create an arc label object

Parametri

parent -- pointer to an object, it will be the parent of the new arc label

Ritorna

pointer to the created arc label

void lv_arclabel_set_text(lv_obj_t *obj, const char *text)

Set the text of the arc label.

This function sets the text displayed by an arc label object.

Parametri

- **obj** -- Pointer to the arc label object.

- **text** -- Pointer to a null-terminated string containing the new text for the label.

void lv_arclabel_set_text_fmt(lv_obj_t *obj, const char *fmt, ...)

Set the formatted text of an arc label object.

This function sets the text of an arc label object with support for variable arguments formatting, similar to `printf`.

Parametri

- **obj** -- The arc label object to set the text for.

- **fmt** -- A format string that specifies how subsequent arguments are converted to text.

- ... -- Arguments following the format string that are used to replace format specifiers in the format string.

void lv_arclabel_set_text_static(lv_obj_t *obj, const char *text)

Sets a new static text for the arc label or refreshes it with the current text. The 'text' must remain valid in memory; the arc label does not manage its lifecycle.

Parametri

- **obj** -- Pointer to the arc label object.
- **text** -- Pointer to the new text. If NULL, the label is refreshed with its current text.

void lv_arclabel_set_angle_start(lv_obj_t *obj, lv_value_precise_t start)

Set the start angle of an arc. 0 deg: right, 90 bottom, etc.

Parametri

- **obj** -- pointer to an arc label object
- **start** -- the start angle. (if LV_USE_FLOAT is enabled it can be fractional too.)

void lv_arclabel_set_angle_size(lv_obj_t *obj, lv_value_precise_t size)

Set the end angle of an arc. 0 deg: right, 90 bottom, etc.

Parametri

- **obj** -- pointer to an arc label object
- **size** -- the angle size (if LV_USE_FLOAT is enabled it can be fractional too.)

void lv_arclabel_set_offset(lv_obj_t *obj, int32_t offset)

Set the rotation for the whole arc

Parametri

- **obj** -- pointer to an arc label object
- **offset** -- rotation angle

void lv_arclabel_set_dir(lv_obj_t *obj, *lv_arclabel_dir_t* dir)

Set the type of arc.

Parametri

- **obj** -- pointer to an arc label object
- **dir** -- arc label's direction

void lv_arclabel_set_recolor(lv_obj_t *obj, bool en)

Enable the recoloring by in-line commands

Parametri

- **obj** -- pointer to an arc label object
- **en** -- true: enable recoloring, false: disable Example: "This is a #ff0000 red# word"

void lv_arclabel_set_radius(lv_obj_t *obj, uint32_t radius)

Set the radius for an arc label object.

Parametri

- **obj** -- pointer to the arc label object.
- **radius** -- The radius value to set for the label's curvature, in pixels.

void lv_arclabel_set_center_offset_x(lv_obj_t *obj, uint32_t x)

Set the center offset x for an arc label object.

Parametri

- **obj** -- pointer to an arc label object
- **x** -- the x offset

void lv_arclabel_set_center_offset_y(lv_obj_t *obj, uint32_t y)

Set the center offset y for an arc label object.

Parametri

- **obj** -- pointer to an arc label object
- **y** -- the y offset

void lv_arclabel_set_text_vertical_align(lv_obj_t *obj, *lv_arclabel_text_align_t* align)

Set the text vertical alignment for an arc label object.

Parametri

- **obj** -- pointer to an arc label object
- **align** -- the vertical alignment

`void lv_arclabel_set_text_horizontal_align(lv_obj_t *obj, lv_arclabel_text_align_t align)`

Set the text horizontal alignment for an arc label object.

Parametri

- **obj** -- pointer to an arc label object
- **align** -- the horizontal alignment

`lv_value_precise_t lv_arclabel_get_angle_start(lv_obj_t *obj)`

Get the start angle of an arc label.

Parametri

obj -- pointer to an arc label object

Ritorna

the start angle [0..360] (if LV_USE_FLOAT is enabled it can be fractional too.)

`lv_value_precise_t lv_arclabel_get_angle_size(lv_obj_t *obj)`

Get the angle size of an arc label.

Parametri

obj -- pointer to an arc label object

Ritorna

the end angle [0..360] (if LV_USE_FLOAT is enabled it can be fractional too.)

`lv_arclabel_dir_t lv_arclabel_get_dir(const lv_obj_t *obj)`

Get whether the arc label is type or not.

Parametri

obj -- pointer to an arc label object

Ritorna

arc label's direction

`bool lv_arclabel_get_recolor(lv_obj_t *obj)`

Enable the recoloring by in-line commands

 **Vedi anche**

[lv_arclabel_set_recolor](#)

Parametri

obj -- pointer to a label object

Ritorna

true: enable recoloring, false: disable

`uint32_t lv_arclabel_get_radius(lv_obj_t *obj)`

Get the text of the arc label.

Parametri

obj -- pointer to an arc label object

Ritorna

the radius of the arc label

`uint32_t lv_arclabel_get_center_offset_x(lv_obj_t *obj)`

Get the center offset x for an arc label object.

Parametri

obj -- pointer to an arc label object

Ritorna

the x offset

`uint32_t lv_arclabel_get_center_offset_y(lv_obj_t *obj)`

Get the center offset y for an arc label object.

Parametri

obj -- pointer to an arc label object

Ritorna

the y offset

```
lv_arclabel_text_align_t lv_arclabel_get_text_vertical_align(lv_obj_t *obj)
```

Get the text vertical alignment for an arc label object.

Parametri

obj -- pointer to an arc label object

Ritorna

the vertical alignment

```
lv_arclabel_text_align_t lv_arclabel_get_text_horizontal_align(lv_obj_t *obj)
```

Get the text horizontal alignment for an arc label object.

Parametri

obj -- pointer to an arc label object

Ritorna

the horizontal alignment

Variabili

```
const lv_obj_class_t lv_arclabel_class
```

lv_arclabel_private.h

```
struct _lv_arclabel_t
```

Membri Pubblici

lv_obj_t obj

char *text

char dot[3 + 1]

Bytes that have been replaced with dots

uint32_t dot_begin

Offset where bytes have been replaced with dots

lv_value_precise_t angle_start

The starting angle of the arc in degrees.

This variable represents the beginning of the arc's angular range in terms of degrees. It is used in conjunction with **angle_size** to define the section of the circle that the arc covers. Values are normalized to the range [0, 360), ensuring compatibility with circular representations.

Modifying this value affects the visual rendering of the arc and its associated indicators or labels, necessitating a subsequent invalidation or update of the object's display to reflect the changes.

 **Nota**

When setting this value programmatically, ensure it does not exceed 360 degrees, as it will be automatically adjusted to fit within the valid range.

lv_value_precise_t angle_size

```

uint32_t offset

uint32_t radius

lv_point_t center_offset

lv_arclabel_dir_t dir

lv_arclabel_text_align_t text_align_v
    Vertical text alignment

lv_arclabel_text_align_t text_align_h
    Horizontal text alignment

uint8_t static_txt
    Flag to indicate the text is static

uint8_t recolor
    Enable in-line letter re-coloring

```

14.27.5 bar

lv_bar.h

Enum

enum **lv_bar_mode_t**

Values:

enumerator **LV_BAR_MODE_NORMAL**

enumerator **LV_BAR_MODE_SYMMETRICAL**

enumerator **LV_BAR_MODE_RANGE**

enum **lv_bar_orientation_t**

Values:

enumerator **LV_BAR_ORIENTATION_AUTO**

enumerator **LV_BAR_ORIENTATION_HORIZONTAL**

enumerator **LV_BAR_ORIENTATION_VERTICAL**

Funzioni

lv_obj_t *lv_bar_create(lv_obj_t *parent)

Create a bar object

Parametri

parent -- pointer to an object, it will be the parent of the new bar

Ritorna

pointer to the created bar

```
void lv_bar_set_value(lv_obj_t *obj, int32_t value, lv_anim_enable_t anim)
```

Set a new value on the bar

Parametri

- **obj** -- pointer to a bar object
- **value** -- new value
- **anim** -- LV_ANIM_ON: set the value with an animation; LV_ANIM_OFF: change the value immediately

```
void lv_bar_set_start_value(lv_obj_t *obj, int32_t start_value, lv_anim_enable_t anim)
```

Set a new start value on the bar

Parametri

- **obj** -- pointer to a bar object
- **start_value** -- new start value
- **anim** -- LV_ANIM_ON: set the value with an animation; LV_ANIM_OFF: change the value immediately

```
void lv_bar_set_range(lv_obj_t *obj, int32_t min, int32_t max)
```

Set minimum and the maximum values of a bar

 **Nota**

If min is greater than max, the drawing direction becomes to the opposite direction.

Parametri

- **obj** -- pointer to the bar object
- **min** -- minimum value
- **max** -- maximum value

```
void lv_bar_set_min_value(lv_obj_t *obj, int32_t min)
```

Set minimum value of a bar

Parametri

- **obj** -- pointer to the bar object
- **min** -- minimum value

```
void lv_bar_set_max_value(lv_obj_t *obj, int32_t max)
```

Set maximum value of a bar

Parametri

- **obj** -- pointer to the bar object
- **max** -- maximum value

```
void lv_bar_set_mode(lv_obj_t *obj, lv_bar_mode_t mode)
```

Set the type of bar.

Parametri

- **obj** -- pointer to bar object
- **mode** -- bar type from *lv_bar_mode_t*

```
void lv_bar_set_orientation(lv_obj_t *obj, lv_bar_orientation_t orientation)
```

Set the orientation of bar.

Parametri

- **obj** -- pointer to bar object
- **orientation** -- bar orientation from *lv_bar_orientation_t*

```
int32_t lv_bar_get_value(const lv_obj_t *obj)
```

Get the value of a bar

Parametri

- **obj** -- pointer to a bar object

Ritorna

the value of the bar

```
int32_t lv_bar_get_start_value(const lv_obj_t *obj)
```

Get the start value of a bar

Parametri

obj -- pointer to a bar object

Ritorna

the start value of the bar

```
int32_t lv_bar_get_min_value(const lv_obj_t *obj)
```

Get the minimum value of a bar

Parametri

obj -- pointer to a bar object

Ritorna

the minimum value of the bar

```
int32_t lv_bar_get_max_value(const lv_obj_t *obj)
```

Get the maximum value of a bar

Parametri

obj -- pointer to a bar object

Ritorna

the maximum value of the bar

```
lv_bar_mode_t lv_bar_get_mode(lv_obj_t *obj)
```

Get the type of bar.

Parametri

obj -- pointer to bar object

Ritorna

bar type from `lv_bar_mode_t`

```
lv_bar_orientation_t lv_bar_get_orientation(lv_obj_t *obj)
```

Get the orientation of bar.

Parametri

obj -- pointer to bar object

Ritorna

bar orientation from `lv_bar_orientation_t`

```
bool lv_bar_is_symmetrical(lv_obj_t *obj)
```

Give the bar is in symmetrical mode or not

Parametri

obj -- pointer to bar object

Ritorna

true: in symmetrical mode false : not in

```
lv_observer_t *lv_bar_bind_value(lv_obj_t *obj, lv_subject_t *subject)
```

Bind an integer or float Subject to a Bar's value.

Parametri

- **obj** -- pointer to Bar
- **subject** -- pointer to Subject

Ritorna

pointer to newly-created Observer

Variabili

```
const lv_obj_class_t lv_bar_class
```

lv_bar_private.h

```
struct _lv_bar_anim_t
```

Membri Pubblici

`lv_obj_t *bar`

`int32_t anim_start`

`int32_t anim_end`

`int32_t anim_state`

```
struct _lv_bar_t
```

Membri Pubblici

`lv_obj_t obj`

`int32_t cur_value`

Current value of the bar

`int32_t min_value`

Minimum value of the bar

`int32_t max_value`

Maximum value of the bar

`int32_t start_value`

Start value of the bar

`lv_area_t indic_area`

Save the indicator area. Might be used by derived types

`bool val_reversed`

Whether value been reversed

`lv_bar_anim_t cur_value_anim`

`lv_bar_anim_t start_value_anim`

`lv_bar_mode_t mode`

Type of bar

`lv_bar_orientation_t orientation`

Orientation of bar

14.27.6 button

lv_button.h

Funzioni

`lv_obj_t *lv_button_create(lv_obj_t *parent)`

Create a button object

Parametri

`parent` -- pointer to an object, it will be the parent of the new button

Ritorna

pointer to the created button

Variabili

`const lv_obj_class_t lv_button_class`

lv_button_private.h

`struct _lv_button_t`

Membri Pubblici

`lv_obj_t obj`

14.27.7 buttonmatrix

lv_buttonmatrix.h

Define

`LV_BUTTONMATRIX_BUTTON_NONE`

Typedef

`typedef bool (*lv_buttonmatrix_button_draw_cb_t)(lv_obj_t *btnm, uint32_t btn_id, const lv_area_t *draw_area, const lv_area_t *clip_area)`

Enum

`enum lv_buttonmatrix_ctrl_t`

Type to store button control flags (disabled, hidden etc.) The least-significant 4 bits are used to store button-width proportions in range [1..15].

Values:

enumerator `LV_BUTTONMATRIX_CTRL_NONE`

No extra control, use the default settings

enumerator `LV_BUTTONMATRIX_CTRL_WIDTH_1`

Set the width to 1 relative to the other buttons in the same row

enumerator `LV_BUTTONMATRIX_CTRL_WIDTH_2`

Set the width to 2 relative to the other buttons in the same row

enumerator **LV_BUTTONMATRIX_CTRL_WIDTH_3**

Set the width to 3 relative to the other buttons in the same row

enumerator **LV_BUTTONMATRIX_CTRL_WIDTH_4**

Set the width to 4 relative to the other buttons in the same row

enumerator **LV_BUTTONMATRIX_CTRL_WIDTH_5**

Set the width to 5 relative to the other buttons in the same row

enumerator **LV_BUTTONMATRIX_CTRL_WIDTH_6**

Set the width to 6 relative to the other buttons in the same row

enumerator **LV_BUTTONMATRIX_CTRL_WIDTH_7**

Set the width to 7 relative to the other buttons in the same row

enumerator **LV_BUTTONMATRIX_CTRL_WIDTH_8**

Set the width to 8 relative to the other buttons in the same row

enumerator **LV_BUTTONMATRIX_CTRL_WIDTH_9**

Set the width to 9 relative to the other buttons in the same row

enumerator **LV_BUTTONMATRIX_CTRL_WIDTH_10**

Set the width to 10 relative to the other buttons in the same row

enumerator **LV_BUTTONMATRIX_CTRL_WIDTH_11**

Set the width to 11 relative to the other buttons in the same row

enumerator **LV_BUTTONMATRIX_CTRL_WIDTH_12**

Set the width to 12 relative to the other buttons in the same row

enumerator **LV_BUTTONMATRIX_CTRL_WIDTH_13**

Set the width to 13 relative to the other buttons in the same row

enumerator **LV_BUTTONMATRIX_CTRL_WIDTH_14**

Set the width to 14 relative to the other buttons in the same row

enumerator **LV_BUTTONMATRIX_CTRL_WIDTH_15**

Set the width to 15 relative to the other buttons in the same row

enumerator **LV_BUTTONMATRIX_CTRL_HIDDEN**

Hides button; it continues to hold its space in layout.

enumerator **LV_BUTTONMATRIX_CTRL_NO_REPEAT**

Do not emit LV_EVENT_LONG_PRESSED_REPEAT events while button is long-pressed.

enumerator **LV_BUTTONMATRIX_CTRL_DISABLED**

Disables button like LV_STATE_DISABLED on normal Widgets.

enumerator LV_BUTTONMATRIX_CTRL_CHECKABLE

Enable toggling of LV_STATE_CHECKED when clicked.

enumerator LV_BUTTONMATRIX_CTRL_CHECKED

Make the button checked. It will use the :cpp:enumerator:LV_STATE_CHECKED styles.

enumerator LV_BUTTONMATRIX_CTRL_CLICK_TRIG

1: Enables sending LV_EVENT_VALUE_CHANGE on CLICK, 0: sends LV_EVENT_VALUE_CHANGE on PRESS.

enumerator LV_BUTTONMATRIX_CTRL_POPOVER

Show button text in a pop-over while being pressed.

enumerator LV_BUTTONMATRIX_CTRL_RECOLOR

Enable text recoloring with #color

enumerator LV_BUTTONMATRIX_CTRL_RESERVED_1

Reserved for later use

enumerator LV_BUTTONMATRIX_CTRL_RESERVED_2

Reserved for later use

enumerator LV_BUTTONMATRIX_CTRL_CUSTOM_1

Custom free-to-use flag

enumerator LV_BUTTONMATRIX_CTRL_CUSTOM_2

Custom free-to-use flag

Funzioni

`lv_obj_t *lv_buttonmatrix_create(lv_obj_t *parent)`

Create a button matrix object

Parametri

- **parent** -- pointer to an object, it will be the parent of the new button matrix

Ritorna

pointer to the created button matrix

`void lv_buttonmatrix_set_map(lv_obj_t *obj, const char *const map[])`

Set a new map. Buttons will be created/deleted according to the map. The button matrix keeps a reference to the map and so the string array must not be deallocated during the life of the matrix.

Parametri

- **obj** -- pointer to a button matrix object
- **map** -- pointer a string array. The last string has to be: "". Use "\n" to make a line break.

`void lv_buttonmatrix_set_ctrl_map(lv_obj_t *obj, const lv_buttonmatrix_ctrl_t ctrl_map[])`

Set the button control map (hidden, disabled etc.) for a button matrix. The control map array will be copied and so may be deallocated after this function returns.

Parametri

- **obj** -- pointer to a button matrix object
- **ctrl_map** -- pointer to an array of `lv_button_ctrl_t` control bytes. The length of the array and position of the elements must match the number and order of the individual buttons (i.e. excludes newline entries). An element of the map should

look like e.g.: `ctrl_map[0] = width | LV_BUTTONMATRIX_CTRL_NO_REPEAT | LV_BUTTONMATRIX_CTRL_TGL_ENABLE`

void lv_buttonmatrix_set_selected_button(lv_obj_t *obj, uint32_t btn_id)

Set the selected buttons

Parametri

- **obj** -- pointer to button matrix object
- **btn_id** -- 0 based index of the button to modify. (Not counting new lines)

void lv_buttonmatrix_set_button_ctrl(lv_obj_t *obj, uint32_t btn_id, lv_buttonmatrix_ctrl_t ctrl)

Set the attributes of a button of the button matrix

Parametri

- **obj** -- pointer to button matrix object
- **btn_id** -- 0 based index of the button to modify. (Not counting new lines)
- **ctrl** -- OR-ed attributes. E.g. `LV_BUTTONMATRIX_CTRL_NO_REPEAT | LV_BUTTONMATRIX_CTRL_CHECKABLE`

void lv_buttonmatrix_clear_button_ctrl(lv_obj_t *obj, uint32_t btn_id, lv_buttonmatrix_ctrl_t ctrl)

Clear the attributes of a button of the button matrix

Parametri

- **obj** -- pointer to button matrix object
- **btn_id** -- 0 based index of the button to modify. (Not counting new lines)
- **ctrl** -- OR-ed attributes. E.g. `LV_BUTTONMATRIX_CTRL_NO_REPEAT | LV_BUTTONMATRIX_CTRL_CHECKABLE`

void lv_buttonmatrix_set_button_ctrl_all(lv_obj_t *obj, lv_buttonmatrix_ctrl_t ctrl)

Set attributes of all buttons of a button matrix

Parametri

- **obj** -- pointer to a button matrix object
- **ctrl** -- attribute(s) to set from `lv_buttonmatrix_ctrl_t`. Values can be ORed.

void lv_buttonmatrix_clear_button_ctrl_all(lv_obj_t *obj, lv_buttonmatrix_ctrl_t ctrl)

Clear the attributes of all buttons of a button matrix

Parametri

- **obj** -- pointer to a button matrix object
- **ctrl** -- attribute(s) to set from `lv_buttonmatrix_ctrl_t`. Values can be ORed.

void lv_buttonmatrix_set_button_width(lv_obj_t *obj, uint32_t btn_id, uint32_t width)

Set a single button's relative width. This method will cause the matrix be regenerated and is a relatively expensive operation. It is recommended that initial width be specified using `lv_buttonmatrix_set_ctrl_map` and this method only be used for dynamic changes.

Parametri

- **obj** -- pointer to button matrix object
- **btn_id** -- 0 based index of the button to modify.
- **width** -- relative width compared to the buttons in the same row. [1..15]

void lv_buttonmatrix_set_one_checked(lv_obj_t *obj, bool en)

Make the button matrix like a selector widget (only one button may be checked at a time). `LV_BUTTONMATRIX_CTRL_CHECKABLE` must be enabled on the buttons to be selected using `lv_buttonmatrix_set_ctrl()` or `lv_buttonmatrix_set_button_ctrl_all()`.

Parametri

- **obj** -- pointer to a button matrix object
- **en** -- whether "one check" mode is enabled

const char *const *lv_buttonmatrix_get_map(const lv_obj_t *obj)

Get the current map of a button matrix

Parametri

obj -- pointer to a button matrix object

Ritorna

the current map

`uint32_t lv_buttonmatrix_get_selected_button(const lv_obj_t *obj)`
Get the index of the lastly "activated" button by the user (pressed, released, focused etc) Useful in the event_cb to get the text of the button, check if hidden etc.

Parametri

- **obj** -- pointer to button matrix object

Ritorna

index of the last released button (LV_BUTTONMATRIX_BUTTON_NONE: if unset)

`const char *lv_buttonmatrix_get_button_text(const lv_obj_t *obj, uint32_t btn_id)`

Get the button's text

Parametri

- **obj** -- pointer to button matrix object
- **btn_id** -- the index a button not counting new line characters.

Ritorna

text of btn_index` button

`bool lv_buttonmatrix_has_button_ctrl(lv_obj_t *obj, uint32_t btn_id, lv_buttonmatrix_ctrl_t ctrl)`

Get the whether a control value is enabled or disabled for button of a button matrix

Parametri

- **obj** -- pointer to a button matrix object
- **btn_id** -- the index of a button not counting new line characters.
- **ctrl** -- control values to check (ORed value can be used)

Ritorna

true: the control attribute is enabled false: disabled

`bool lv_buttonmatrix_get_one_checked(const lv_obj_t *obj)`

Tell whether "one check" mode is enabled or not.

Parametri

- **obj** -- Button matrix object

Ritorna

true: "one check" mode is enabled; false: disabled

Variabili

`const lv_obj_class_t lv_buttonmatrix_class`

lv_buttonmatrix_private.h

```
struct _lv_buttonmatrix_t
#include <lv_buttonmatrix_private.h> Data of button matrix
```

Membri Pubblici

`lv_obj_t obj`

`const char *const *map_p`

Pointer to the current map

`lv_area_t *button_areas`

Array of areas of buttons

`lv_buttonmatrix_ctrl_t *ctrl_bits`

Array of control bytes

uint32_t btn_cnt

Number of button in 'map_p'(Handled by the library)

uint32_t row_cnt

Number of rows in 'map_p'(Handled by the library)

uint32_t btn_id_sel

Index of the active button (being pressed/released etc) or LV_BUTTONMATRIX_BUTTON_NONE

uint32_t one_check

1: Single button toggled at once

uint32_t auto_free_map

1: Automatically free the map when the widget is deleted

14.27.8 calendar

lv_calendar.h

Funzioni

lv_obj_t *lv_calendar_create(lv_obj_t *parent)

Create a calendar widget

Parametri

- **parent** -- pointer to an object, it will be the parent of the new calendar

Ritorna

pointer the created calendar

void lv_calendar_set_today_date(lv_obj_t *obj, uint32_t year, uint32_t month, uint32_t day)

Set the today's year, month and day at once

Parametri

- **obj** -- pointer to a calendar object
- **year** -- today's year
- **month** -- today's month [1..12]
- **day** -- today's day [1..31]

void lv_calendar_set_today_year(lv_obj_t *obj, uint32_t year)

Set the today's year

Parametri

- **obj** -- pointer to a calendar object
- **year** -- today's year

void lv_calendar_set_today_month(lv_obj_t *obj, uint32_t month)

Set the today's year

Parametri

- **obj** -- pointer to a calendar object
- **month** -- today's month [1..12]

void lv_calendar_set_today_day(lv_obj_t *obj, uint32_t day)

Set the today's year

Parametri

- **obj** -- pointer to a calendar object
- **day** -- today's day [1..31]

void lv_calendar_set_month_shown(lv_obj_t *obj, uint32_t year, uint32_t month)

Set the currently shown year and month at once

Parametri

- **obj** -- pointer to a calendar object
- **year** -- shown year
- **month** -- shown month [1..12]

`void lv_calendar_set_shown_year(lv_obj_t *obj, uint32_t year)`

Set the currently shown year

Parametri

- **obj** -- pointer to a calendar object
- **year** -- shown year

`void lv_calendar_set_shown_month(lv_obj_t *obj, uint32_t month)`

Set the currently shown month

Parametri

- **obj** -- pointer to a calendar object
- **month** -- shown month [1..12]

`void lv_calendar_set_highlighted_dates(lv_obj_t *obj, lv_calendar_date_t highlighted[], size_t date_num)`

Set the highlighted dates

Parametri

- **obj** -- pointer to a calendar object
- **highlighted** -- pointer to an `lv_calendar_date_t` array containing the dates. Only the pointer will be saved so this variable can't be local which will be destroyed later.
- **date_num** -- number of dates in the array

`void lv_calendar_set_day_names(lv_obj_t *obj, const char **day_names)`

Set the name of the days

Parametri

- **obj** -- pointer to a calendar object
- **day_names** -- pointer to an array with the names. E.g. `const char * days[7] = {"Sun", "Mon", ...}` Only the pointer will be saved so this variable can't be local which will be destroyed later.

`lv_obj_t *lv_calendar_get_btnmatrix(const lv_obj_t *obj)`

Get the button matrix object of the calendar. It shows the dates and day names.

Parametri

`obj` -- pointer to a calendar object

Ritorna

pointer to a the button matrix

`const lv_calendar_date_t *lv_calendar_get_today_date(const lv_obj_t *calendar)`

Get the today's date

Parametri

`calendar` -- pointer to a calendar object

Ritorna

return pointer to an `lv_calendar_date_t` variable containing the date of today.

`const lv_calendar_date_t *lv_calendar_get_showed_date(const lv_obj_t *calendar)`

Get the currently showed

Parametri

`calendar` -- pointer to a calendar object

Ritorna

pointer to an `lv_calendar_date_t` variable containing the date is being shown.

`lv_calendar_date_t *lv_calendar_get_highlighted_dates(const lv_obj_t *calendar)`

Get the highlighted dates

Parametri

`calendar` -- pointer to a calendar object

Ritorna

pointer to an `lv_calendar_date_t` array containing the dates.

`size_t lv_calendar_get_highlighted_dates_num(const lv_obj_t *calendar)`

Get the number of the highlighted dates

Parametri

- **calendar** -- pointer to a calendar object

Ritorna

number of highlighted days

`lv_result_t lv_calendar_get_pressed_date(const lv_obj_t *calendar, lv_calendar_date_t *date)`

Get the currently pressed day

Parametri

- **calendar** -- pointer to a calendar object
- **date** -- store the pressed date here

Ritorna

LV_RESULT_OK: there is a valid pressed date
LV_RESULT_INVALID: there is no pressed data

Variabili

`const lv_obj_class_t lv_calendar_class`

`struct lv_calendar_date_t`

#include <lv_calendar.h> Represents a date on the calendar object (platform-agnostic).

Membri Pubblici

`uint16_t year`

`uint8_t month`

1..12

`uint8_t day`

1..31

lv_calendar_chinese.h

Funzioni

`void lv_calendar_set_chinese_mode(lv_obj_t *obj, bool en)`

Enable the chinese calendar.

Parametri

- **obj** -- pointer to a calendar object.
- **en** -- true: enable chinese calendar; false: disable

`const char *lv_calendar_get_day_name(lv_calendar_date_t *gregorian)`

Get the name of the day

Parametri

gregorian -- to obtain the gregorian time for the name

Ritorna

return the name of the day

`void lv_calendar_gregorian_to_chinese(lv_calendar_date_t *gregorian_time, lv_calendar_chinese_t *chinese_time)`

Get the chinese time of the gregorian time (reference: <https://www.cnblogs.com/liyang31tg/p/4123171.html>)

Parametri

- **gregorian_time** -- need to convert to chinese time in gregorian time
- **chinese_time** -- the chinese time convert from gregorian time

```
struct lv_calendar_chinese_t
```

Membri Pubblici

lv_calendar_date_t **today**

bool **leap_month**

lv_calendar_header_arrow.h

Funzioni

lv_obj_t ***lv_calendar_add_header_arrow**(*lv_obj_t* *parent)

Create a calendar header with drop-downs to select the year and month

Parametri

parent -- pointer to a calendar object.

Ritorna

the created header

Variabili

const *lv_obj_class_t* **lv_calendar_header_arrow_class**

lv_calendar_header_dropdown.h

Funzioni

lv_obj_t ***lv_calendar_add_header_dropdown**(*lv_obj_t* *parent)

Create a calendar header with drop-downs to select the year and month

Parametri

parent -- pointer to a calendar object.

Ritorna

the created header

void **lv_calendar_header_dropdown_set_year_list**(*lv_obj_t* *parent, const char *years_list)

Sets a custom calendar year list

Parametri

- **parent** -- pointer to a calendar object
- **years_list** -- pointer to an const char array with the years list, see *lv_dropdown_set_options* for more information. E.g. `const char * years = "2023\n2022\n2021\n2020\n2019" Only the pointer will be saved so this variable can't be local which will be destroyed later.

Variabili

const *lv_obj_class_t* **lv_calendar_header_dropdown_class**

lv_calendar_private.h

struct **_lv_calendar_t**

#include <lv_calendar_private.h> Data of calendar

Membri Pubblici

`lv_obj_t obj`

`lv_obj_t *btnm`

`lv_calendar_date_t today`

Date of today

`lv_calendar_date_t showed_date`

Currently visible month (day is ignored)

`lv_calendar_date_t *highlighted_dates`

Apply different style on these days (pointer to user-defined array)

`size_t highlighted_dates_num`

Number of elements in highlighted_days

`const char *map[8 * 7]`

`bool use_chinese_calendar`

`char nums[7 * 6][20]`

7 * 6: A week has 7 days, and the calendar displays 6 weeks in total. 20: Including the number of dates, line breaks, names for each day, and reserving several spaces for addresses.

14.27.9 canvas

lv_canvas.h

Define

`LV_CANVAS_BUF_SIZE(w, h, bpp, stride)`

Funzioni

`lv_obj_t *lv_canvas_create(lv_obj_t *parent)`

Create a canvas object

Parametri

`parent` -- pointer to an object, it will be the parent of the new canvas

Ritorna

pointer to the created canvas

`void lv_canvas_set_buffer(lv_obj_t *obj, void *buf, int32_t w, int32_t h, lv_color_format_t cf)`

Set a buffer for the canvas.

Use `lv_canvas_set_draw_buf()` instead if you need to set a buffer with alignment requirement.

Parametri

- `obj` -- pointer to a canvas object
- `buf` -- buffer where content of canvas will be. The required size is (`lv_image_color_format_get_px_size(cf) * w / 8 * h`) It can be allocated with `lv_malloc()` or it can be statically allocated array (e.g. static `lv_color_t buf[100*50]`) or it can be an address in RAM or external SRAM
- `w` -- width of canvas
- `h` -- height of canvas

- **cf** -- color format. LV_COLOR_FORMAT...

`void lv_canvas_set_draw_buf(lv_obj_t *obj, lv_draw_buf_t *draw_buf)`

Set a draw buffer for the canvas. A draw buffer either can be allocated by `lv_draw_buf_create()` or defined statically by LV_DRAW_BUF_DEFINE_STATIC. When buffer start address and stride has alignment requirement, it's recommended to use `lv_draw_buf_create`.

Parametri

- **obj** -- pointer to a canvas object
- **draw_buf** -- pointer to a draw buffer

`void lv_canvas_set_px(lv_obj_t *obj, int32_t x, int32_t y, lv_color_t color, lv_opa_t opa)`

Set a pixel's color and opacity

Nota

The following color formats are supported LV_COLOR_FORMAT_I1/2/4/8, LV_COLOR_FORMAT_A8, LV_COLOR_FORMAT_RGB565, LV_COLOR_FORMAT_RGB888, LV_COLOR_FORMAT_XRGB8888, LV_COLOR_FORMAT_ARGB8888

Parametri

- **obj** -- pointer to a canvas
- **x** -- X coordinate of the pixel
- **y** -- Y coordinate of the pixel
- **color** -- the color
- **opa** -- the opacity

`void lv_canvas_set_palette(lv_obj_t *obj, uint8_t index, lv_color32_t color)`

Set the palette color of a canvas for index format. Valid only for LV_COLOR_FORMAT_I1/2/4/8

Parametri

- **obj** -- pointer to canvas object
- **index** -- the palette color to set:
 - for LV_COLOR_FORMAT_I1: 0..1
 - for LV_COLOR_FORMAT_I2: 0..3
 - for LV_COLOR_FORMAT_I4: 0..15
 - for LV_COLOR_FORMAT_I8: 0..255
- **color** -- the color to set

`lv_draw_buf_t *lv_canvas_get_draw_buf(lv_obj_t *obj)`

`lv_color32_t lv_canvas_get_px(lv_obj_t *obj, int32_t x, int32_t y)`

Get a pixel's color and opacity

Parametri

- **obj** -- pointer to a canvas
- **x** -- X coordinate of the pixel
- **y** -- Y coordinate of the pixel

Ritorna

ARGB8888 color of the pixel

`lv_image_dsc_t *lv_canvas_get_image(lv_obj_t *canvas)`

Get the image of the canvas as a pointer to an `lv_image_dsc_t` variable.

Parametri

- **canvas** -- pointer to a canvas object

Ritorna

pointer to the image descriptor.

`const void *lv_canvas_get_buf(lv_obj_t *canvas)`

Return the pointer for the buffer. It's recommended to use this function instead of the buffer form the return value of `lv_canvas_get_image()` as is can be aligned

Parametri

- canvas** -- pointer to a canvas object

Ritorna

- pointer to the buffer

```
void lv_canvas_copy_buf(lv_obj_t *obj, const lv_area_t *canvas_area, lv_draw_buf_t *dest_buf, const lv_area_t *dest_area)
```

Copy a buffer to the canvas

Parametri

- obj** -- pointer to a canvas object
- canvas_area** -- the area of the canvas to copy
- dest_buf** -- pointer to a buffer to store the copied data
- dest_area** -- the area of the destination buffer to copy to. If omitted NULL, copy to the whole dest_buf

```
void lv_canvas_fill_bg(lv_obj_t *obj, lv_color_t color, lv_opa_t opa)
```

Fill the canvas with color

Parametri

- obj** -- pointer to a canvas
- color** -- the background color
- opa** -- the desired opacity

```
void lv_canvas_init_layer(lv_obj_t *canvas, lv_layer_t *layer)
```

Initialize a layer to use LVGL's generic draw functions (lv_draw_rect/label/...) on the canvas. Needs to be used in pair with `lv_canvas_finish_layer`.

Parametri

- canvas** -- pointer to a canvas
- layer** -- pointer to a layer variable to initialize

```
void lv_canvas_finish_layer(lv_obj_t *canvas, lv_layer_t *layer)
```

Wait until all the drawings are finished on layer. Needs to be used in pair with `lv_canvas_init_layer`.

Parametri

- canvas** -- pointer to a canvas
- layer** -- pointer to a layer to finalize

```
uint32_t lv_canvas_buf_size(int32_t w, int32_t h, uint8_t bpp, uint8_t stride)
```

Just a wrapper to `LV_CANVAS_BUF_SIZE` for bindings.

Variabili

```
const lv_obj_class_t lv_canvas_class
```

lv_canvas_private.h

```
struct _lv_canvas_t
```

```
#include <lv_canvas_private.h> Canvas data
```

Membri Pubblici

```
lv_image_t img
```

```
lv_draw_buf_t *draw_buf
```

```
lv_draw_buf_t static_buf
```

14.27.10 chart

[lv_chart.h](#)

Define

LV_CHART_POINT_NONE

Default value of points. Can be used to not draw a point

Enum

enum **lv_chart_type_t**

Chart types

Values:

enumerator **LV_CHART_TYPE_NONE**

Don't draw the series

enumerator **LV_CHART_TYPE_LINE**

Connect the points with lines

enumerator **LV_CHART_TYPE_BAR**

Draw bars for each series

enumerator **LV_CHART_TYPE_STACKED**

Draw a single stacked bar for each data point. Supports only positive values

enumerator **LV_CHART_TYPE_SCATTER**

Draw points and lines in 2D (x,y coordinates)

enum **lv_chart_update_mode_t**

Chart update mode for `lv_chart_set_next`

Values:

enumerator **LV_CHART_UPDATE_MODE_SHIFT**

Shift old data to the left and add the new one the right

enumerator **LV_CHART_UPDATE_MODE_CIRCULAR**

Add the new data in a circular way

enum **lv_chart_axis_t**

Enumeration of the axis'

Values:

enumerator **LV_CHART_AXIS_PRIMARY_Y**

enumerator **LV_CHART_AXIS_SECONDARY_Y**

enumerator **LV_CHART_AXIS_PRIMARY_X**

enumerator `LV_CHART_AXIS_SECONDARY_X`

enumerator `LV_CHART_AXIS_LAST`

Funzioni

`lv_obj_t *lv_chart_create(lv_obj_t *parent)`

Create a chart object

Parametri

- `parent` -- pointer to an object, it will be the parent of the new chart

Ritorna

pointer to the created chart

`void lv_chart_set_type(lv_obj_t *obj, lv_chart_type_t type)`

Set a new type for a chart

Parametri

- `obj` -- pointer to a chart object
- `type` -- new type of the chart (from '`lv_chart_type_t`' enum)

`void lv_chart_set_point_count(lv_obj_t *obj, uint32_t cnt)`

Set the number of points on a data line on a chart

Parametri

- `obj` -- pointer to a chart object
- `cnt` -- new number of points on the data lines

`void lv_chart_set_axis_range(lv_obj_t *obj, lv_chart_axis_t axis, int32_t min, int32_t max)`

Set the minimal and maximal y values on an axis

Parametri

- `obj` -- pointer to a chart object
- `axis` -- `LV_CHART_AXIS_PRIMARY_Y` or `LV_CHART_AXIS_SECONDARY_Y`
- `min` -- minimum value of the y axis
- `max` -- maximum value of the y axis

`void lv_chart_set_axis_min_value(lv_obj_t *obj, lv_chart_axis_t axis, int32_t min)`

Set the minimal values on an axis

Parametri

- `obj` -- pointer to a chart object
- `axis` -- `LV_CHART_AXIS_PRIMARY_Y` or `LV_CHART_AXIS_SECONDARY_Y`
- `min` -- minimal value of the y axis

`void lv_chart_set_axis_max_value(lv_obj_t *obj, lv_chart_axis_t axis, int32_t max)`

Set the maximal y values on an axis

Parametri

- `obj` -- pointer to a chart object
- `axis` -- `LV_CHART_AXIS_PRIMARY_Y` or `LV_CHART_AXIS_SECONDARY_Y`
- `max` -- maximum value of the y axis

`void lv_chart_set_update_mode(lv_obj_t *obj, lv_chart_update_mode_t update_mode)`

Set update mode of the chart object. Affects

Parametri

- `obj` -- pointer to a chart object
- `update_mode` -- the update mode

`void lv_chart_set_div_line_count(lv_obj_t *obj, uint32_t hdiv, uint32_t vdiv)`

Set the number of horizontal and vertical division lines

Parametri

- `obj` -- pointer to a chart object
- `hdiv` -- number of horizontal division lines

- **vdiv** -- number of vertical division lines

void lv_chart_set_hor_div_line_count(lv_obj_t *obj, uint32_t cnt)

Set the number of horizontal division lines

Parametri

- **obj** -- pointer to a chart object
- **cnt** -- number of horizontal division lines

void lv_chart_set_ver_div_line_count(lv_obj_t *obj, uint32_t cnt)

Set the number of vertical division lines

Parametri

- **obj** -- pointer to a chart object
- **cnt** -- number of vertical division lines

lv_chart_type_t **lv_chart_get_type(const lv_obj_t *obj)**

Get the type of a chart

Parametri

- **obj** -- pointer to chart object

Ritorna

type of the chart (from 'lv_chart_t' enum)

uint32_t lv_chart_get_point_count(const lv_obj_t *obj)

Get the data point number per data line on chart

Parametri

- **obj** -- pointer to chart object

Ritorna

point number on each data line

uint32_t lv_chart_get_x_start_point(const lv_obj_t *obj, lv_chart_series_t *ser)

Get the current index of the x-axis start point in the data array

Parametri

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'

Ritorna

the index of the current x start point in the data array

void lv_chart_get_point_pos_by_id(lv_obj_t *obj, lv_chart_series_t *ser, uint32_t id, *lv_point_t* *p_out)

Get the position of a point to the chart.

Parametri

- **obj** -- pointer to a chart object
- **ser** -- pointer to series
- **id** -- the index.
- **p_out** -- store the result position here

void lv_chart_refresh(lv_obj_t *obj)

Refresh a chart if its data line has changed

Parametri

- **obj** -- pointer to chart object

lv_chart_series_t *lv_chart_add_series(lv_obj_t *obj, *lv_color_t* color, *lv_chart_axis_t* axis)

Allocate and add a data series to the chart

Parametri

- **obj** -- pointer to a chart object
- **color** -- color of the data series
- **axis** -- the y axis to which the series should be attached (:LV_CHART_AXIS_PRIMARY_Y or ::LV_CHART_AXIS_SECONDARY_Y)

Ritorna

pointer to the allocated data series or NULL on failure

void lv_chart_remove_series(lv_obj_t *obj, lv_chart_series_t *series)

Deallocate and remove a data series from a chart

Parametri

- **obj** -- pointer to a chart object
- **series** -- pointer to a data series on 'chart'

void lv_chart_hide_series(lv_obj_t *chart, lv_chart_series_t *series, bool hide)

Hide/Unhide a single series of a chart.

Parametri

- **chart** -- pointer to a chart object.
- **series** -- pointer to a series object
- **hide** -- true: hide the series

void lv_chart_set_series_color(lv_obj_t *chart, lv_chart_series_t *series, lv_color_t color)

Change the color of a series

Parametri

- **chart** -- pointer to a chart object.
- **series** -- pointer to a series object
- **color** -- the new color of the series

lv_color_t lv_chart_get_series_color(lv_obj_t *chart, const lv_chart_series_t *series)

Get the color of a series

Parametri

- **chart** -- pointer to a chart object.
- **series** -- pointer to a series object

Ritorna

the color of the series

void lv_chart_set_x_start_point(lv_obj_t *obj, lv_chart_series_t *ser, uint32_t id)

Set the index of the x-axis start point in the data array. This point will be considered the first (left) point and the other points will be drawn after it.

Parametri

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'
- **id** -- the index of the x point in the data array

lv_chart_series_t *lv_chart_get_series_next(const lv_obj_t *chart, const lv_chart_series_t *ser)

Get the next series.

Parametri

- **chart** -- pointer to a chart
- **ser** -- the previous series or NULL to get the first

Ritorna

the next series or NULL if there is no more.

lv_chart_cursor_t *lv_chart_add_cursor(lv_obj_t *obj, lv_color_t color, lv_dir_t dir)

Add a cursor with a given color

Parametri

- **obj** -- pointer to chart object
- **color** -- color of the cursor
- **dir** -- direction of the cursor. LV_DIR_RIGHT/LEFT/TOP/DOWN/HOR/VER/ALL.
OR-ed values are possible

Ritorna

pointer to the created cursor

void lv_chart_remove_cursor(lv_obj_t *obj, lv_chart_cursor_t *cursor)

Remove a cursor

Parametri

- **obj** -- pointer to chart object
- **cursor** -- pointer to the cursor

void lv_chart_set_cursor_pos(lv_obj_t *chart, lv_chart_cursor_t *cursor, lv_point_t *pos)

Set the coordinate of the cursor with respect to the paddings

Parametri

- **chart** -- pointer to a chart object
- **cursor** -- pointer to the cursor
- **pos** -- the new coordinate of cursor relative to the chart

void lv_chart_set_cursor_pos_x(lv_obj_t *chart, lv_chart_cursor_t *cursor, int32_t x)

Set the X coordinate of the cursor with respect to the paddings

Parametri

- **chart** -- pointer to a chart object
- **cursor** -- pointer to the cursor
- **x** -- the new X coordinate of cursor relative to the chart

void lv_chart_set_cursor_pos_y(lv_obj_t *chart, lv_chart_cursor_t *cursor, int32_t y)

Set the coordinate of the cursor with respect to the paddings

Parametri

- **chart** -- pointer to a chart object
- **cursor** -- pointer to the cursor
- **y** -- the new Y coordinate of cursor relative to the chart

void lv_chart_set_cursor_point(lv_obj_t *chart, lv_chart_cursor_t *cursor, lv_chart_series_t *ser, uint32_t point_id)

Stick the cursor to a point

Parametri

- **chart** -- pointer to a chart object
- **cursor** -- pointer to the cursor
- **ser** -- pointer to a series
- **point_id** -- the point's index or LV_CHART_POINT_NONE to not assign to any points.

lv_point_t lv_chart_get_cursor_point(lv_obj_t *chart, lv_chart_cursor_t *cursor)

Get the coordinate of the cursor with respect to the paddings

Parametri

- **chart** -- pointer to a chart object
- **cursor** -- pointer to cursor

Ritorna

coordinate of the cursor as *lv_point_t*

void lv_chart_set_all_values(lv_obj_t *obj, lv_chart_series_t *ser, int32_t value)

Initialize all data points of a series with a value

Parametri

- **obj** -- pointer to chart object
- **ser** -- pointer to a data series on 'chart'
- **value** -- the new value for all points. LV_CHART_POINT_NONE can be used to hide the points.

void lv_chart_set_next_value(lv_obj_t *obj, lv_chart_series_t *ser, int32_t value)

Set the next point's Y value according to the update mode policy.

Parametri

- **obj** -- pointer to chart object
- **ser** -- pointer to a data series on 'chart'
- **value** -- the new value of the next data

void lv_chart_set_next_value2(lv_obj_t *obj, lv_chart_series_t *ser, int32_t x_value, int32_t y_value)

Set the next point's X and Y value according to the update mode policy.

Parametri

- **obj** -- pointer to chart object
- **ser** -- pointer to a data series on 'chart'
- **x_value** -- the new X value of the next data
- **y_value** -- the new Y value of the next data

void lv_chart_set_series_values(lv_obj_t *obj, lv_chart_series_t *ser, const int32_t values[], size_t values_cnt)

Same as `lv_chart_set_next_value` but set the values from an array

Parametri

- **obj** -- pointer to chart object
- **ser** -- pointer to a data series on 'chart'
- **values** -- the new values to set
- **values_cnt** -- number of items in **values**

```
void lv_chart_set_series_values2(lv_obj_t *obj, lv_chart_series_t *ser, const int32_t x_values[], const int32_t y_values[], size_t values_cnt)
```

Same as `lv_chart_set_next_value2` but set the values from an array

Parametri

- **obj** -- pointer to chart object
- **ser** -- pointer to a data series on 'chart'
- **x_values** -- the new values to set on the X axis
- **y_values** -- the new values to set on the Y axis
- **values_cnt** -- number of items in **x_values** and **y_values**

```
void lv_chart_set_series_value_by_id(lv_obj_t *obj, lv_chart_series_t *ser, uint32_t id, int32_t value)
```

Set an individual point's y value of a chart's series directly based on its index

Parametri

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'
- **id** -- the index of the x point in the array
- **value** -- value to assign to array point

```
void lv_chart_set_series_value_by_id2(lv_obj_t *obj, lv_chart_series_t *ser, uint32_t id, int32_t x_value, int32_t y_value)
```

Set an individual point's x and y value of a chart's series directly based on its index Can be used only with `LV_CHART_TYPE_SCATTER`.

Parametri

- **obj** -- pointer to chart object
- **ser** -- pointer to a data series on 'chart'
- **id** -- the index of the x point in the array
- **x_value** -- the new X value of the next data
- **y_value** -- the new Y value of the next data

```
void lv_chart_set_series_ext_y_array(lv_obj_t *obj, lv_chart_series_t *ser, int32_t array[])
```

Set an external array for the y data points to use for the chart NOTE: It is the users responsibility to make sure the `point_cnt` matches the external array size.

Parametri

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'
- **array** -- external array of points for chart

```
void lv_chart_set_series_ext_x_array(lv_obj_t *obj, lv_chart_series_t *ser, int32_t array[])
```

Set an external array for the x data points to use for the chart NOTE: It is the users responsibility to make sure the `point_cnt` matches the external array size.

Parametri

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'
- **array** -- external array of points for chart

```
int32_t *lv_chart_get_series_y_array(const lv_obj_t *obj, lv_chart_series_t *ser)
```

Get the array of y values of a series

Parametri

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'

Ritorna

the array of values with '`point_count`' elements

```
int32_t *lv_chart_get_series_x_array(const lv_obj_t *obj, lv_chart_series_t *ser)
```

Get the array of x values of a series

Parametri

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'

Ritorna

the array of values with 'point_count' elements

```
uint32_t lv_chart_get_pressed_point(const lv_obj_t *obj)
```

Get the index of the currently pressed point. It's the same for every series.

Parametri

- **obj** -- pointer to a chart object

Ritorna

the index of the point [0 .. point count] or LV_CHART_POINT_ID_NONE if no point is being pressed

```
int32_t lv_chart_get_first_point_center_offset(lv_obj_t *obj)
```

Get the overall offset from the chart's side to the center of the first point. In case of a bar chart it will be the center of the first column group

Parametri

- **obj** -- pointer to a chart object

Ritorna

the offset of the center

Variabili

```
const lv_obj_class_t lv_chart_class
```

lv_chart_private.h

```
struct _lv_chart_series_t
#include <lv_chart_private.h> Descriptor a chart series
```

Membri Pubblici

```
int32_t *x_points
```

```
int32_t *y_points
```

```
lv_color_t color
```

```
uint32_t start_point
```

```
uint32_t hidden
```

```
uint32_t x_ext_buf_assigned
```

```
uint32_t y_ext_buf_assigned
```

```
uint32_t x_axis_sec
```

```
uint32_t y_axis_sec

struct _lv_chart_cursor_t
```

Membri Pubblici*lv_point_t* pos

int32_t point_id

lv_color_t color

lv_chart_series_t *ser

lv_dir_t dir

uint32_t pos_set

1: pos is set; 0: point_id is set

struct _lv_chart_t

Membri Pubblici

lv_obj_t obj

lv_ll_t series_ll

Linked list for series (stores lv_chart_series_t)

lv_ll_t cursor_ll

Linked list for cursors (stores lv_chart_cursor_t)

int32_t ymin[2]

int32_t ymax[2]

int32_t xmin[2]

int32_t xmax[2]

int32_t pressed_point_id

uint32_t hdiv_cnt

Number of horizontal division lines

uint32_t vdiv_cnt

Number of vertical division lines

```
uint32_t point_cnt
```

Number of points in all series

```
lv_chart_type_t type
```

Chart type

```
lv_chart_update_mode_t update_mode
```

14.27.11 checkbox

lv_checkbox.h

Funzioni

```
lv_obj_t *lv_checkbox_create(lv_obj_t *parent)
```

Create a check box object

Parametri

- parent** -- pointer to an object, it will be the parent of the new button

Ritorna

pointer to the created check box

```
void lv_checkbox_set_text(lv_obj_t *obj, const char *txt)
```

Set the text of a check box. txt will be copied and may be deallocated after this function returns.

Parametri

- obj** -- pointer to a check box
- txt** -- the text of the check box. NULL to refresh with the current text.

```
void lv_checkbox_set_text_static(lv_obj_t *obj, const char *txt)
```

Set the text of a check box. txt must not be deallocated during the life of this checkbox.

Parametri

- obj** -- pointer to a check box
- txt** -- the text of the check box.

```
const char *lv_checkbox_get_text(const lv_obj_t *obj)
```

Get the text of a check box

Parametri

- obj** -- pointer to check box object

Ritorna

pointer to the text of the check box

Variabili

```
const lv_obj_class_t lv_checkbox_class
```

lv_checkbox_private.h

```
struct _lv_checkbox_t
```

Membri Pubblici

```
lv_obj_t obj
```

```
char *txt
```

```
uint32_t static_txt
```

14.27.12 dropdown

lv_dropdown.h

Define

LV_DROPDOWN_POS_LAST

Enum

enum **_lv_property_dropdown_id_t**

Values:

enumerator **LV_PROPERTY_DROPDOWN_TEXT**

enumerator **LV_PROPERTY_DROPDOWN_OPTIONS**

enumerator **LV_PROPERTY_DROPDOWN_OPTION_COUNT**

enumerator **LV_PROPERTY_DROPDOWN_SELECTED**

enumerator **LV_PROPERTY_DROPDOWN_DIR**

enumerator **LV_PROPERTY_DROPDOWN_SYMBOL**

enumerator **LV_PROPERTY_DROPDOWN_SELECTED_HIGHLIGHT**

enumerator **LV_PROPERTY_DROPDOWN_LIST**

enumerator **LV_PROPERTY_DROPDOWN_IS_OPEN**

enumerator **LV_PROPERTY_DROPDOWN_END**

Funzioni

lv_obj_t *lv_dropdown_create(lv_obj_t *parent)

Create a drop-down list object

Parametri

parent -- pointer to an object, it will be the parent of the new drop-down list

Ritorna

pointer to the created drop-down list

void lv_dropdown_set_text(lv_obj_t *obj, const char *txt)

Set text of the drop-down list's button. If set to NULL the selected option's text will be displayed on the button. If set to a specific text then that text will be shown regardless of the selected option.

Parametri

- **obj** -- pointer to a drop-down list object
- **txt** -- the text as a string (Only its pointer is saved)

void lv_dropdown_set_options(lv_obj_t *obj, const char *options)

Set the options in a drop-down list from a string. The options will be copied and saved in the object so the options can be destroyed after calling this function

Parametri

- **obj** -- pointer to drop-down list object
- **options** -- a string with '
' separated options. E.g. "One\nTwo\nThree"

void lv_dropdown_set_options_static(lv_obj_t *obj, const char *options)

Set the options in a drop-down list from a static string (global, static or dynamically allocated). Only the pointer of the option string will be saved.

Parametri

- **obj** -- pointer to drop-down list object
- **options** -- a static string with '
' separated options. E.g. "One\nTwo\nThree"

void lv_dropdown_add_option(lv_obj_t *obj, const char *option, uint32_t pos)

Add an options to a drop-down list from a string. Only works for non-static options.

Parametri

- **obj** -- pointer to drop-down list object
- **option** -- a string without '
' E.g. "Four"
- **pos** -- the insert position, indexed from 0, LV_DROPDOWN_POS_LAST = end of string

void lv_dropdown_clear_options(lv_obj_t *obj)

Clear all options in a drop-down list. Works with both static and dynamic options.

Parametri

- **obj** -- pointer to drop-down list object

void lv_dropdown_set_selected(lv_obj_t *obj, uint32_t sel_opt)

Set the selected option

Parametri

- **obj** -- pointer to drop-down list object
- **sel_opt** -- id of the selected option (0 ... number of option - 1);

void lv_dropdown_set_dir(lv_obj_t *obj, lv_dir_t dir)

Set the direction of the a drop-down list

Parametri

- **obj** -- pointer to a drop-down list object
- **dir** -- LV_DIR_LEFT/RIGHT/TOP/BOTTOM

void lv_dropdown_set_symbol(lv_obj_t *obj, const void *symbol)

Set an arrow or other symbol to display when on drop-down list's button. Typically a down caret or arrow.

Nota

angle and zoom transformation can be applied if the symbol is an image. E.g. when drop down is checked (opened) rotate the symbol by 180 degree

Parametri

- **obj** -- pointer to drop-down list object
- **symbol** -- a text like LV_SYMBOL_DOWN, an image (pointer or path) or NULL to not draw symbol icon

void lv_dropdown_set_selected_highlight(lv_obj_t *obj, bool en)

Set whether the selected option in the list should be highlighted or not

Parametri

- **obj** -- pointer to drop-down list object
- **en** -- true: highlight enabled; false: disabled

`lv_obj_t *lv_dropdown_get_list(lv_obj_t *obj)`

Get the list of a drop-down to allow styling or other modifications

Parametri

`obj` -- pointer to a drop-down list object

Ritorna

pointer to the list of the drop-down

`const char *lv_dropdown_get_text(lv_obj_t *obj)`

Get text of the drop-down list's button.

Parametri

`obj` -- pointer to a drop-down list object

Ritorna

the text as string, NULL if no text

`const char *lv_dropdown_get_options(const lv_obj_t *obj)`

Get the options of a drop-down list

Parametri

`obj` -- pointer to drop-down list object

Ritorna

the options separated by '

'-s (E.g. "Option1\nOption2\nOption3")

`uint32_t lv_dropdown_get_selected(const lv_obj_t *obj)`

Get the index of the selected option

Parametri

`obj` -- pointer to drop-down list object

Ritorna

index of the selected option (0 ... number of option - 1);

`uint32_t lv_dropdown_get_option_count(const lv_obj_t *obj)`

Get the total number of options

Parametri

`obj` -- pointer to drop-down list object

Ritorna

the total number of options in the list

`void lv_dropdown_get_selected_str(const lv_obj_t *obj, char *buf, uint32_t buf_size)`

Get the current selected option as a string

Parametri

- `obj` -- pointer to drop-down object
- `buf` -- pointer to an array to store the string
- `buf_size` -- size of buf in bytes. 0: to ignore it.

`int32_t lv_dropdown_get_option_index(lv_obj_t *obj, const char *option)`

Get the index of an option.

Parametri

- `obj` -- pointer to drop-down object
- `option` -- an option as string

Ritorna

index of option in the list of all options. -1 if not found.

`const char *lv_dropdown_get_symbol(lv_obj_t *obj)`

Get the symbol on the drop-down list. Typically a down caret or arrow.

Parametri

`obj` -- pointer to drop-down list object

Ritorna

the symbol or NULL if not enabled

`bool lv_dropdown_get_selected_highlight(lv_obj_t *obj)`

Get whether the selected option in the list should be highlighted or not

Parametri

`obj` -- pointer to drop-down list object

Ritorna

true: highlight enabled; false: disabled

`lv_dir_t lv_dropdown_get_dir(const lv_obj_t *obj)`

Get the direction of the drop-down list

Parametri

`obj` -- pointer to a drop-down list object

Ritorna

LV_DIR_LEFT/RIGHT/TOP/BOTTOM

`void lv_dropdown_open(lv_obj_t *dropdown_obj)`

Open the drop-down list

Parametri

`dropdown_obj` -- pointer to drop-down list object

`void lv_dropdown_close(lv_obj_t *obj)`

Close (Collapse) the drop-down list

Parametri

`obj` -- pointer to drop-down list object

`bool lv_dropdown_is_open(lv_obj_t *obj)`

Tells whether the list is opened or not

Parametri

`obj` -- pointer to a drop-down list object

Ritorna

true if the list is opened

`lv_observer_t *lv_dropdown_bind_value(lv_obj_t *obj, lv_subject_t *subject)`

Bind an integer Subject to a Dropdown's value.

Parametri

- `obj` -- pointer to Dropdown
- `subject` -- pointer to Subject

Ritorna

pointer to newly-created Observer

Variabili

`const lv_obj_class_t lv_dropdown_class`

`const lv_obj_class_t lv_dropdownlist_class`

lv_dropdown_private.h

`struct _lv_dropdown_t`

Membri Pubblici

`lv_obj_t obj`

`lv_obj_t *list`

The dropped down list

```

const char *text
    Text to display on the dropdown's button

const void *symbol
    Arrow or other icon when the drop-down list is closed

char *options
    Options in a '
    ' separated list

uint32_t option_cnt
    Number of options

uint32_t sel_opt_id
    Index of the currently selected option

uint32_t sel_opt_id_orig
    Store the original index on focus

uint32_t pr_opt_id
    Index of the currently pressed option

uint8_t dir
    Direction in which the list should open

uint8_t static_txt
    1: Only a pointer is saved in options

uint8_t selected_highlight
    1: Make the selected option highlighted in the list

```

struct _lv_dropdown_list_t

Membri Pubblici

lv_obj_t obj

lv_obj_t *dropdown

14.27.13 image

lv_image.h

Define

LV_IMAGE_DECLARE(var_name)

Use this macro to declare an image in a C file

Enum

enum **lv_image_align_t**

Image size mode, when image size and object size is different

Values:

enumerator **LV_IMAGE_ALIGN_DEFAULT**

enumerator **LV_IMAGE_ALIGN_TOP_LEFT**

enumerator **LV_IMAGE_ALIGN_TOP_MID**

enumerator **LV_IMAGE_ALIGN_TOP_RIGHT**

enumerator **LV_IMAGE_ALIGN_BOTTOM_LEFT**

enumerator **LV_IMAGE_ALIGN_BOTTOM_MID**

enumerator **LV_IMAGE_ALIGN_BOTTOM_RIGHT**

enumerator **LV_IMAGE_ALIGN_LEFT_MID**

enumerator **LV_IMAGE_ALIGN_RIGHT_MID**

enumerator **LV_IMAGE_ALIGN_CENTER**

enumerator **_LV_IMAGE_ALIGN_AUTO_TRANSFORM**

Marks the start of modes that transform the image

enumerator **LV_IMAGE_ALIGN_STRETCH**

Set X and Y scale to fill the Widget's area.

enumerator **LV_IMAGE_ALIGN_TILE**

Tile image to fill Widget's area. Offset is applied to shift the tiling.

enumerator **LV_IMAGE_ALIGN_CONTAIN**

The image keeps its aspect ratio, but is resized to the maximum size that fits within the Widget's area.

enumerator **LV_IMAGE_ALIGN_COVER**

The image keeps its aspect ratio and fills the Widget's area.

enum **_lv_property_image_id_t**

Values:

enumerator **LV_PROPERTY_IMAGE_SRC**

enumerator **LV_PROPERTY_IMAGE_OFFSET_X**

enumerator **LV_PROPERTY_IMAGE_OFFSET_Y**

enumerator **LV_PROPERTY_IMAGE_ROTATION**

enumerator **LV_PROPERTY_IMAGE_PIVOT**

enumerator **LV_PROPERTY_IMAGE_SCALE**

enumerator **LV_PROPERTY_IMAGE_SCALE_X**

enumerator **LV_PROPERTY_IMAGE_SCALE_Y**

enumerator **LV_PROPERTY_IMAGE_BLEND_MODE**

enumerator **LV_PROPERTY_IMAGE_ANTIALIAS**

enumerator **LV_PROPERTY_IMAGE_INNER_ALIGN**

enumerator **LV_PROPERTY_IMAGE_END**

Funzioni

`lv_obj_t *lv_image_create(lv_obj_t *parent)`

Create an image object

Parametri

parent -- pointer to an object, it will be the parent of the new image

Ritorna

pointer to the created image

`void lv_image_set_src(lv_obj_t *obj, const void *src)`

Set the image data to display on the object

Parametri

- **obj** -- pointer to an image object
- **src** -- 1) pointer to an `lv_image_dsc_t` descriptor (converted by LVGL's image converter) (e.g. &my_img) or 2) path to an image file (e.g. "S:/dir/img.bin") or 3) a SYMBOL (e.g. LV_SYMBOL_OK)

`void lv_image_set_offset_x(lv_obj_t *obj, int32_t x)`

Set an offset for the source of an image so the image will be displayed from the new origin.

Parametri

- **obj** -- pointer to an image
- **x** -- the new offset along x axis.

`void lv_image_set_offset_y(lv_obj_t *obj, int32_t y)`

Set an offset for the source of an image. so the image will be displayed from the new origin.

Parametri

- **obj** -- pointer to an image
- **y** -- the new offset along y axis.

`void lv_image_set_rotation(lv_obj_t *obj, int32_t angle)`

Set the rotation angle of the image. The image will be rotated around the set pivot set by `lv_image_set_pivot()`. Note that indexed and alpha only images can't be transformed.

Nota

if `image_align` is `LV_IMAGE_ALIGN_STRETCH` or `LV_IMAGE_ALIGN_FIT` rotation will be set to 0 automatically.

Parametri

- **obj** -- pointer to an image object
- **angle** -- rotation in degree with 0.1 degree resolution (0..3600: clock wise)

`void lv_image_set_pivot(lv_obj_t *obj, int32_t x, int32_t y)`

Set the rotation center of the image. The image will be rotated around this point. x, y can be set with value of `LV_PCT`, `lv_image_get_pivot` will return the true pixel coordinate of pivot in this case.

Parametri

- **obj** -- pointer to an image object
- **x** -- rotation center x of the image
- **y** -- rotation center y of the image

`void lv_image_set_pivot_x(lv_obj_t *obj, int32_t x)`

Set the rotation horizontal center of the image.

Parametri

- **obj** -- pointer to an image object
- **x** -- rotation center x of the image, or `lv_pct()`

`void lv_image_set_pivot_y(lv_obj_t *obj, int32_t y)`

Set the rotation vertical center of the image.

Parametri

- **obj** -- pointer to an image object
- **y** -- rotation center y of the image, or `lv_pct()`

`void lv_image_set_scale(lv_obj_t *obj, uint32_t zoom)`

Set the zoom factor of the image. Note that indexed and alpha only images can't be transformed.

Parametri

- **obj** -- pointer to an image object
- **zoom** -- the zoom factor. Example values:
 - 256 or `LV_SCALE_NONE`: no zoom
 - <256: scale down
 - >256: scale up
 - 128: half size
 - 512: double size

`void lv_image_set_scale_x(lv_obj_t *obj, uint32_t zoom)`

Set the horizontal zoom factor of the image. Note that indexed and alpha only images can't be transformed.

Parametri

- **obj** -- pointer to an image object
- **zoom** -- the zoom factor. Example values:
 - 256 or `LV_SCALE_NONE`: no zoom
 - <256: scale down
 - >256: scale up
 - 128: half size
 - 512: double size

`void lv_image_set_scale_y(lv_obj_t *obj, uint32_t zoom)`

Set the vertical zoom factor of the image. Note that indexed and alpha only images can't be transformed.

Parametri

- **obj** -- pointer to an image object
- **zoom** -- the zoom factor. Example values:
 - 256 or LV_SCALE_NONE: no zoom
 - <256: scale down
 - >256: scale up
 - 128: half size
 - 512: double size

`void lv_image_set_blend_mode(lv_obj_t *obj, lv_blend_mode_t blend_mode)`

Set the blend mode of an image.

Parametri

- **obj** -- pointer to an image object
- **blend_mode** -- the new blend mode

`void lv_image_set_antialias(lv_obj_t *obj, bool antialias)`

Enable/disable anti-aliasing for the transformations (rotate, zoom) or not. The quality is better with anti-aliasing looks better but slower.

Parametri

- **obj** -- pointer to an image object
- **antialias** -- true: anti-aliased; false: not anti-aliased

`void lv_image_set_inner_align(lv_obj_t *obj, lv_image_align_t align)`

Set the image object size mode.

Nota

if `image_align` is `LV_IMAGE_ALIGN_STRETCH` or `LV_IMAGE_ALIGN_FIT` rotation, scale and pivot will be overwritten and controlled internally.

Parametri

- **obj** -- pointer to an image object
- **align** -- the new align mode.

`void lv_image_set_bitmap_map_src(lv_obj_t *obj, const lv_image_dsc_t *src)`

Set an A8 bitmap mask for the image.

Parametri

- **obj** -- pointer to an image object
- **src** -- an `lv_image_dsc_t` bitmap mask source.

`const void *lv_image_get_src(lv_obj_t *obj)`

Get the source of the image

Parametri

`obj` -- pointer to an image object

Ritorna

the image source (symbol, file name or `::lv-img_dsc_t` for C arrays)

`int32_t lv_image_get_offset_x(lv_obj_t *obj)`

Get the offset's x attribute of the image object.

Parametri

`obj` -- pointer to an image

Ritorna

offset X value.

`int32_t lv_image_get_offset_y(lv_obj_t *obj)`

Get the offset's y attribute of the image object.

Parametri

`obj` -- pointer to an image

Ritorna

offset Y value.

```
int32_t lv_image_get_rotation(lv_obj_t *obj)
```

Get the rotation of the image.

 **Nota**

if `image_align` is `LV_IMAGE_ALIGN_STRETCH` or `LV_IMAGE_ALIGN_FIT` rotation will be set to 0 automatically.

Parametri

- `obj` -- pointer to an image object

Ritorna

rotation in 0.1 degrees (0..3600)

```
void lv_image_get_pivot(lv_obj_t *obj, lv_point_t *pivot)
```

Get the pivot (rotation center) of the image. If pivot is set with `LV_PCT`, convert it to px before return.

Parametri

- `obj` -- pointer to an image object
- `pivot` -- store the rotation center here

```
int32_t lv_image_get_scale(lv_obj_t *obj)
```

Get the zoom factor of the image.

Parametri

- `obj` -- pointer to an image object

Ritorna

zoom factor (256: no zoom)

```
int32_t lv_image_get_scale_x(lv_obj_t *obj)
```

Get the horizontal zoom factor of the image.

Parametri

- `obj` -- pointer to an image object

Ritorna

zoom factor (256: no zoom)

```
int32_t lv_image_get_scale_y(lv_obj_t *obj)
```

Get the vertical zoom factor of the image.

Parametri

- `obj` -- pointer to an image object

Ritorna

zoom factor (256: no zoom)

```
int32_t lv_image_get_src_width(lv_obj_t *obj)
```

Get the width of an image before any transformations.

Parametri

- `obj` -- Pointer to an image object.

Ritorna

The width of the image.

```
int32_t lv_image_get_src_height(lv_obj_t *obj)
```

Get the height of an image before any transformations.

Parametri

- `obj` -- Pointer to an image object.

Ritorna

The height of the image.

```
int32_t lv_image_get_transformed_width(lv_obj_t *obj)
```

Get the transformed width of an image object.

Parametri

obj -- Pointer to an image object.

Ritorna

The transformed width of the image.

```
int32_t lv_image_get_transformed_width(lv_obj_t *obj)
```

Get the transformed width of an image object.

Parametri

obj -- Pointer to an image object.

Ritorna

The transformed width of the image.

```
lv_blend_mode_t lv_image_get_blend_mode(lv_obj_t *obj)
```

Get the current blend mode of the image

Parametri

obj -- pointer to an image object

Ritorna

the current blend mode

```
bool lv_image_get_antialias(lv_obj_t *obj)
```

Get whether the transformations (rotate, zoom) are anti-aliased or not

Parametri

obj -- pointer to an image object

Ritorna

true: anti-aliased; false: not anti-aliased

```
lv_align_t lv_image_get_inner_align(lv_obj_t *obj)
```

Get the size mode of the image

Parametri

obj -- pointer to an image object

Ritorna

element of *lv_align_t*

```
const lv_image_dsc_t *lv_image_get_bitmap_map_src(lv_obj_t *obj)
```

Get the bitmap mask source.

Parametri

obj -- pointer to an image object

Ritorna

an *lv_image_dsc_t* bitmap mask source.

```
lv_observer_t *lv_image_bind_src(lv_obj_t *obj, lv_subject_t *subject)
```

Bind a pointer Subject to an Image's source.

Parametri

- **obj** -- pointer to Image
- **subject** -- pointer to Subject

Ritorna

pointer to newly-created Observer

Variabili

```
const lv_obj_class_t lv_image_class
```

lv_image_private.h

```
struct _lv_image_t
```

```
#include <lv_image_private.h> Data of image
```

Membri Pubblici**lv_obj_t obj****const void *src**

Image source: Pointer to an array or a file or a symbol

const *lv_image_dsc_t* *bitmap_mask_src

Pointer to an A8 bitmap mask

lv_point_t* offset*int32_t w**

Width of the image (Handled by the library)

int32_t h

Height of the image (Handled by the library)

uint32_t rotation

Rotation angle of the image

uint32_t scale_x

256 means no zoom, 512 double size, 128 half size

uint32_t scale_y

256 means no zoom, 512 double size, 128 half size

***lv_point_t* pivot**

Rotation center of the image

uint32_t src_typeSee: *lv_image_src_t***uint32_t cf**Color format from *lv_color_format_t***uint32_t antialias**

Apply anti-aliasing in transformations (rotate, zoom)

uint32_t alignImage size mode when image size and object size is different. See *lv_image_align_t***uint32_t blend_mode**Element of *lv_blend_mode_t*

14.27.14 imagebutton

lv_imagebutton.h**Enum**

enum **lv_imagebutton_state_t**

Values:

enumerator **LV_IMAGEBUTTON_STATE_RELEASED**

enumerator **LV_IMAGEBUTTON_STATE_PRESSED**

enumerator **LV_IMAGEBUTTON_STATE_DISABLED**

enumerator **LV_IMAGEBUTTON_STATE_CHECKED_RELEASED**

enumerator **LV_IMAGEBUTTON_STATE_CHECKED_PRESSED**

enumerator **LV_IMAGEBUTTON_STATE_CHECKED_DISABLED**

enumerator **LV_IMAGEBUTTON_STATE_NUM**

Funzioni

lv_obj_t *lv_imagebutton_create(lv_obj_t *parent)

Create an image button object

Parametri

- **parent** -- pointer to an object, it will be the parent of the new image button

Ritorna

pointer to the created image button

void lv_imagebutton_set_src(lv_obj_t *imagebutton, lv_imagebutton_state_t state, const void *src_left, const void *src_mid, const void *src_right)

Set images for a state of the image button

Parametri

- **imagebutton** -- pointer to an image button object
- **state** -- for which state set the new image
- **src_left** -- pointer to an image source for the left side of the button (a C array or path to a file)
- **src_mid** -- pointer to an image source for the middle of the button (ideally 1px wide) (a C array or path to a file)
- **src_right** -- pointer to an image source for the right side of the button (a C array or path to a file)

void lv_imagebutton_set_state(lv_obj_t *imagebutton, lv_imagebutton_state_t state)

Use this function instead of `lv_obj_add/remove_state` to set a state manually

Parametri

- **imagebutton** -- pointer to an image button object
- **state** -- the new state

const void *lv_imagebutton_get_src_left(lv_obj_t *imagebutton, lv_imagebutton_state_t state)

Get the left image in a given state

Parametri

- **imagebutton** -- pointer to an image button object
- **state** -- the state where to get the image (from `lv_button_state_t`)

Ritorna

pointer to the left image source (a C array or path to a file)

```
const void *lv_imagebutton_get_src_middle(lv_obj_t *imagebutton, lv_imagebutton_state_t state)
```

Get the middle image in a given state

Parametri

- **imagebutton** -- pointer to an image button object
- **state** -- the state where to get the image (from `lv_button_state_t`)`

Ritorna

pointer to the middle image source (a C array or path to a file)

```
const void *lv_imagebutton_get_src_right(lv_obj_t *imagebutton, lv_imagebutton_state_t state)
```

Get the right image in a given state

Parametri

- **imagebutton** -- pointer to an image button object
- **state** -- the state where to get the image (from `lv_button_state_t`)`

Ritorna

pointer to the left image source (a C array or path to a file)

Variabili

```
const lv_obj_class_t lv_imagebutton_class
```

lv_imagebutton_private.h

```
struct _lv_imagebutton_src_info_t
```

Membri Pubblici

```
const void *img_src
```

```
lv_image_header_t header
```

```
struct _lv_imagebutton_t
```

```
#include <lv_imagebutton_private.h> Data of image button
```

Membri Pubblici

```
lv_obj_t obj
```

```
lv_imagebutton_src_info_t src_mid[LV_IMAGEBUTTON_STATE_NUM]
```

Store center images to each state

```
lv_imagebutton_src_info_t src_left[LV_IMAGEBUTTON_STATE_NUM]
```

Store left side images to each state

```
lv_imagebutton_src_info_t src_right[LV_IMAGEBUTTON_STATE_NUM]
```

Store right side images to each state

14.27.15 ime

lv_ime_pinyin.h

Define

```
LVIME_PINYIN_K9_MAX_INPUT
```

Enum

enum **lv_ime_pinyin_mode_t**

Values:

enumerator **LVIME_PINYIN_MODE_K26**

enumerator **LVIME_PINYIN_MODE_K9**

enumerator **LVIME_PINYIN_MODE_K9_NUMBER**

Funzioni

lv_obj_t ***lv_ime_pinyin_create**(lv_obj_t *parent)

void **lv_ime_pinyin_set_keyboard**(lv_obj_t *obj, lv_obj_t *kb)

Set the keyboard of Pinyin input method.

Parametri

- **obj** -- pointer to a Pinyin input method object
- **kb** -- pointer to a Pinyin input method keyboard

void **lv_ime_pinyin_set_dict**(lv_obj_t *obj, *lv_pinyin_dict_t* *dict)

Set the dictionary of Pinyin input method.

Parametri

- **obj** -- pointer to a Pinyin input method object
- **dict** -- pointer to a Pinyin input method dictionary

void **lv_ime_pinyin_set_mode**(lv_obj_t *obj, *lv_ime_pinyin_mode_t* mode)

Set mode, 26-key *input(k26)* or 9-key *input(k9)*.

Parametri

- **obj** -- pointer to a Pinyin input method object
- **mode** -- the mode from 'lv_ime_pinyin_mode_t'

lv_obj_t ***lv_ime_pinyin_get_kb**(lv_obj_t *obj)

Set the dictionary of Pinyin input method.

Parametri

obj -- pointer to a Pinyin IME object

Ritorna

 pointer to the Pinyin IME keyboard

lv_obj_t ***lv_ime_pinyin_get_cand_panel**(lv_obj_t *obj)

Set the dictionary of Pinyin input method.

Parametri

obj -- pointer to a Pinyin input method object

Ritorna

 pointer to the Pinyin input method candidate panel

const *lv_pinyin_dict_t* ***lv_ime_pinyin_get_dict**(lv_obj_t *obj)

Set the dictionary of Pinyin input method.

Parametri

obj -- pointer to a Pinyin input method object

Ritorna

 pointer to the Pinyin input method dictionary

Variabili

```
const lv_obj_class_t lv_ime_pinyin_class
```

```
struct lv_pinyin_dict_t
```

Membri Pubblici

```
const char *const py
```

```
const char *const py_mb
```

```
struct ime_pinyin_k9_py_str_t
```

Membri Pubblici

```
char py_str[7]
```

lv_ime_pinyin_private.h

```
struct _lv_ime_pinyin_t
```

Membri Pubblici

```
lv_obj_t obj
```

```
lv_obj_t *kb
```

```
lv_obj_t *cand_panel
```

```
const lv_pinyin_dict_t *dict
```

```
lv_ll_t k9_legal_py_ll
```

```
char *cand_str
```

```
char input_char[16]
```

```
char k9_input_str[7 + 1]
```

```
uint16_t k9_py_ll_pos
```

```
uint16_t k9_legal_py_count
```

```
uint16_t k9_input_str_len
```

```
uint16_t ta_count
```

```

uint16_t cand_num

uint16_t py_page

uint16_t py_num[26]

uint16_t py_pos[26]

lv_ime_pinyin_mode_t mode

```

14.27.16 keyboard

lv_keyboard.h

Define

LV_KEYBOARD_CTRL_BUTTON_FLAGS

Enum

enum **lv_keyboard_mode_t**

Current keyboard mode.

Values:

enumerator **LV_KEYBOARD_MODE_TEXT_LOWER**

enumerator **LV_KEYBOARD_MODE_TEXT_UPPER**

enumerator **LV_KEYBOARD_MODE_SPECIAL**

enumerator **LV_KEYBOARD_MODE_NUMBER**

enumerator **LV_KEYBOARD_MODE_USER_1**

enumerator **LV_KEYBOARD_MODE_USER_2**

enumerator **LV_KEYBOARD_MODE_USER_3**

enumerator **LV_KEYBOARD_MODE_USER_4**

enumerator **LV_KEYBOARD_MODE_TEXT_ARABIC**

enum **_lv_property_keyboard_id_t**

Values:

enumerator **LV_PROPERTY_KEYBOARD_TEXTAREA**

enumerator **LV_PROPERTY_KEYBOARD_MODE**

enumerator **LV_PROPERTY_KEYBOARD_POPOVERS**

enumerator **LV_PROPERTY_KEYBOARD_SELECTED_BUTTON**

enumerator **LV_PROPERTY_KEYBOARD_END**

Funzioni

lv_obj_t *lv_keyboard_create(lv_obj_t *parent)

Create a Keyboard object

Parametri

- **parent** -- pointer to an object, it will be the parent of the new keyboard

Ritorna

- pointer to the created keyboard object

void lv_keyboard_set_textarea(lv_obj_t *kb, lv_obj_t *ta)

Assign a text area to the keyboard. Pressed characters will be inserted there.

Parametri

- **kb** -- pointer to a keyboard object
- **ta** -- pointer to a text area object to write into

void lv_keyboard_set_mode(lv_obj_t *kb, lv_keyboard_mode_t mode)

Set a new mode (e.g., text, number, special characters).

Parametri

- **kb** -- pointer to a keyboard object
- **mode** -- the desired mode (see '*lv_keyboard_mode_t*')

void lv_keyboard_set_popovers(lv_obj_t *kb, bool en)

Enable or disable popovers showing button titles on press.

Parametri

- **kb** -- pointer to a keyboard object
- **en** -- true to enable popovers; false to disable

void lv_keyboard_set_map(lv_obj_t *kb, lv_keyboard_mode_t mode, const char *const map[], const lv_buttonmatrix_ctrl_t ctrl_map[])

Set a custom button map for the keyboard.

Parametri

- **kb** -- pointer to a keyboard object
- **mode** -- the mode to assign the new map to (see '*lv_keyboard_mode_t*')
- **map** -- pointer to a string array describing the button map see '*lv_buttonmatrix_set_map()*' for more details
- **ctrl_map** -- pointer to the control map. See '*lv_buttonmatrix_set_ctrl_map()*'

lv_obj_t *lv_keyboard_get_textarea(const lv_obj_t *kb)

Get the text area currently assigned to the keyboard.

Parametri

- **kb** -- pointer to a keyboard object

Ritorna

- pointer to the assigned text area object

lv_keyboard_mode_t lv_keyboard_get_mode(const lv_obj_t *kb)

Get the current mode of the keyboard.

Parametri

- **kb** -- pointer to a keyboard object

Ritorna
the current mode (see '*lv_keyboard_mode_t*')

```
bool lv_keyboard_get_popovers(const lv_obj_t *obj)
```

Check whether popovers are enabled on the keyboard.

Parametri

- obj** -- pointer to a keyboard object

Ritorna
true if popovers are enabled; false otherwise

```
const char *const *lv_keyboard_get_map_array(const lv_obj_t *kb)
```

Get the current button map of the keyboard.

Parametri

- kb** -- pointer to a keyboard object

Ritorna
pointer to the map array

```
uint32_t lv_keyboard_get_selected_button(const lv_obj_t *obj)
```

Get the index of the last selected button (pressed, released, focused, etc.). Useful in the event_cb to retrieve button text or properties.

Parametri

- obj** -- pointer to a keyboard object

Ritorna
index of the last interacted button returns LV_BUTTONMATRIX_BUTTON_NONE if not set

```
const char *lv_keyboard_get_button_text(const lv_obj_t *obj, uint32_t btn_id)
```

Get the text of a button by index.

Parametri

- **obj** -- pointer to a keyboard object
- **btn_id** -- index of the button (excluding newline characters)

Ritorna
pointer to the text of the button

```
void lv_keyboard_def_event_cb(lv_event_t *e)
```

Default keyboard event callback to handle button presses. Adds characters to the text area and switches map if needed. If a custom event_cb is used, this function can be called within it.

Parametri

- e** -- the triggering event

Variabili

```
const lv_obj_class_t lv_keyboard_class
```

lv_keyboard_private.h

```
struct _lv_keyboard_t
#include <lv_keyboard_private.h> Data of keyboard
```

Membri Pubblici

lv_buttonmatrix_t btnm

lv_obj_t *ta

Pointer to the assigned text area

lv_keyboard_mode_t mode

Key map type

uint8_t popovers

Show button titles in popovers on press

14.27.17 label

lv_label.h**Define****LV_LABEL_DOT_NUM****LV_LABEL_POS_LAST****LV_LABEL_TEXT_SELECTION_OFF****LV_LABEL_DEFAULT_TEXT****Enum****enum lv_label_long_mode_t**

Long mode behaviors. Used in 'lv_label_ext_t'

Values:

enumerator LV_LABEL_LONG_MODE_WRAP

Keep the object width, wrap lines longer than object width and expand the object height

enumerator LV_LABEL_LONG_MODE_DOTS

Keep the size and write dots at the end if the text is too long

enumerator LV_LABEL_LONG_MODE_SCROLL

Keep the size and roll the text back and forth

enumerator LV_LABEL_LONG_MODE_SCROLL_CIRCULAR

Keep the size and roll the text circularly

enumerator LV_LABEL_LONG_MODE_CLIP

Keep the size and clip the text out of it

enum _lv_property_label_id_t

Values:

enumerator LV_PROPERTY_LABEL_TEXT**enumerator LV_PROPERTY_LABEL_LONG_MODE**

enumerator **LV_PROPERTY_LABEL_TEXT_SELECTION_START**

enumerator **LV_PROPERTY_LABEL_TEXT_SELECTION_END**

enumerator **LV_PROPERTY_LABEL_END**

Funzioni

`lv_obj_t *lv_label_create(lv_obj_t *parent)`

Create a label object

Parametri

- **parent** -- pointer to an object, it will be the parent of the new label.

Ritorna

- pointer to the created button

`void lv_label_set_text(lv_obj_t *obj, const char *text)`

Set a new text for a label. Memory will be allocated to store the text by the label.

Nota

If `LV_USE_ARABIC_PERSIAN_CHARS` is enabled the text will be modified to have the correct Arabic characters in it.

Parametri

- **obj** -- pointer to a label object
- **text** -- '\0' terminated character string. NULL to refresh with the current text.

`void lv_label_set_text_fmt(lv_obj_t *obj, const char *fmt, ...)`

Set a new formatted text for a label. Memory will be allocated to store the text by the label.

Nota

If `LV_USE_ARABIC_PERSIAN_CHARS` is enabled the text will be modified to have the correct Arabic characters in it.

Parametri

- **obj** -- pointer to a label object
- **fmt** -- printf-like format string Example:

```
lv_label_set_text_fmt(label1, "%d user", user_num);
```

`void lv_label_set_text_vfmt(lv_obj_t *obj, const char *fmt, va_list args)`

Set a new formatted text for a label. Memory will be allocated to store the text by the label.

Example:

```
va_list args;
va_start(args, fmt);
lv_label_set_text_vfmt(label1, fmt, args);
va_end(args);
```

Nota

It ignores `LV_USE_ARABIC_PERSIAN_CHARS`

Parametri

- **obj** -- pointer to a label object

- **fmt** -- printf-like format string
- **args** -- variadic arguments list

`void lv_label_set_text_static(lv_obj_t *obj, const char *text)`

Set a static text. It will not be saved by the label so the 'text' variable has to be 'alive' while the label exists.

Nota

It ignores LV_USE_ARABIC_PERSIAN_CHARS

Parametri

- **obj** -- pointer to a label object
- **text** -- pointer to a text. NULL to refresh with the current text.

`void lv_label_set_long_mode(lv_obj_t *obj, lv_label_long_mode_t long_mode)`

Set the behavior of the label with text longer than the object size

Parametri

- **obj** -- pointer to a label object
- **long_mode** -- the new mode from 'lv_label_long_mode' enum. In LV_LONG_WRAP/DOT/SCROLL/SCROLL_CIRC the size of the label should be set AFTER this function

`void lv_label_set_text_selection_start(lv_obj_t *obj, uint32_t index)`

Set where text selection should start

Parametri

- **obj** -- pointer to a label object
- **index** -- character index from where selection should start. LV_LABEL_TEXT_SELECTION_OFF for no selection

`void lv_label_set_text_selection_end(lv_obj_t *obj, uint32_t index)`

Set where text selection should end

Parametri

- **obj** -- pointer to a label object
- **index** -- character index where selection should end. LV_LABEL_TEXT_SELECTION_OFF for no selection

`void lv_label_set_recolor(lv_obj_t *obj, bool en)`

Enable the recoloring by in-line commands

Parametri

- **obj** -- pointer to a label object
- **en** -- true: enable recoloring, false: disable Example: "This is a #ff0000 red# word"

`void lv_label_set_translation_tag(lv_obj_t *obj, const char *tag)`

Assign a translation tag for this label. Memory will be allocated to store the tag by the label. The label text will automatically update when the language is changed via `lv_translation_set_language`.

Parametri

- **obj** -- pointer to a label object
- **tag** -- '\0' terminated character string.

`char *lv_label_get_text(const lv_obj_t *obj)`

Get the text of a label

Parametri

obj -- pointer to a label object

Ritorna

the text of the label

`lv_label_long_mode_t lv_label_get_long_mode(const lv_obj_t *obj)`

Get the long mode of a label

Parametri

obj -- pointer to a label object

Ritorna

the current long mode

```
void lv_label_get_letter_pos(const lv_obj_t *obj, uint32_t char_id, lv_point_t *pos)
```

Get the relative x and y coordinates of a letter

Parametri

- **obj** -- pointer to a label object
- **char_id** -- index of the character [0 ... text length - 1]. Expressed in character index, not byte index (different in UTF-8)
- **pos** -- store the result here (E.g. index = 0 gives 0;0 coordinates if the text is aligned to the left)

```
uint32_t lv_label_get_letter_on(const lv_obj_t *obj, lv_point_t *pos_in, bool bidi)
```

Get the index of letter on a relative point of a label.

Parametri

- **obj** -- pointer to label object
- **pos_in** -- pointer to point with coordinates on a the label
- **bidi** -- whether to use bidi processed

Ritorna

The index of the letter on the 'pos_p' point (E.g. on 0;0 is the 0. letter if aligned to the left)
Expressed in character index and not byte index (different in UTF-8)

```
bool lv_label_is_char_under_pos(const lv_obj_t *obj, lv_point_t *pos)
```

Check if a character is drawn under a point.

Parametri

- **obj** -- pointer to a label object
- **pos** -- Point to check for character under

Ritorna

whether a character is drawn under the point

```
uint32_t lv_label_get_text_selection_start(const lv_obj_t *obj)
```

Get the selection start index.

Parametri

obj -- pointer to a label object.

Ritorna

selection start index. LV_LABEL_TEXT_SELECTION_OFF if nothing is selected.

```
uint32_t lv_label_get_text_selection_end(const lv_obj_t *obj)
```

Get the selection end index.

Parametri

obj -- pointer to a label object.

Ritorna

selection end index. LV_LABEL_TXT_SEL_0FF if nothing is selected.

```
bool lv_label_get_recolor(const lv_obj_t *obj)
```

Get the recoloring attribute.

Parametri

obj -- pointer to a label object.

Ritorna

true: recoloring is enabled, false: recoloring is disabled

```
lv_observer_t *lv_label_bind_text(lv_obj_t *obj, lv_subject_t *subject, const char *fmt)
```

Bind an integer, string, or pointer Subject to a Label.

Nota

If **fmt** == NULL strings and pointers (\0 terminated string) will be shown as text as they are, integers as d, floats as %0.1f

Parametri

- **obj** -- pointer to Label
- **subject** -- pointer to Subject
- **fmt** -- optional printf-like format string with 1 format specifier (e.g. "%d °C") or NULL to bind to the value directly.

Ritorna

pointer to newly-created Observer

```
void lv_label_ins_text(lv_obj_t *obj, uint32_t pos, const char *txt)
```

Insert a text to a label. The label text cannot be static.

Parametri

- **obj** -- pointer to a label object
- **pos** -- character index to insert. Expressed in character index and not byte index. 0: before first char. LV_LABEL_POS_LAST: after last char.
- **txt** -- pointer to the text to insert

```
void lv_label_cut_text(lv_obj_t *obj, uint32_t pos, uint32_t cnt)
```

Delete characters from a label. The label text cannot be static.

Parametri

- **obj** -- pointer to a label object
- **pos** -- character index from where to cut. Expressed in character index and not byte index. 0: start in front of the first character
- **cnt** -- number of characters to cut

Variabili

```
const lv_obj_class_t lv_label_class
```

lv_label_private.h

```
struct _lv_label_t
```

Membri Pubblici

lv_obj_t obj

char *text

char *translation_tag

char dot[3 + 1]

Bytes that have been replaced with dots

uint32_t dot_begin

Offset where bytes have been replaced with dots

lv_draw_label_hint_t hint

uint32_t sel_start

uint32_t sel_end

lv_point_t **size_cache**

Text size cache

lv_point_t **offset**

Text draw position offset

lv_label_long_mode_t **long_mode**

Determine what to do with the long texts

uint8_t static_txt

Flag to indicate the text is static

uint8_t recolor

Enable in-line letter re-coloring

uint8_t expand

Ignore real width (used by the library with LV_LABEL_LONG_MODE_SCROLL)

uint8_t invalid_size_cache

1: Recalculate size and update cache

uint8_t need_refr_text

1: Refresh text after layout update completion

lv_point_t **text_size**

14.27.18 led

lv_led.h

Define

LV_LED_BRIGHT_MIN

Brightness when the LED if OFF

LV_LED_BRIGHT_MAX

Brightness when the LED if ON

Funzioni

lv_obj_t *lv_led_create(lv_obj_t *parent)

Create a led object

Parametri

- **parent** -- pointer to an object, it will be the parent of the new led

Ritorna

- pointer to the created led

void lv_led_set_color(lv_obj_t *led, *lv_color_t* color)

Set the color of the LED

Parametri

- **led** -- pointer to a LED object
- **color** -- the color of the LED

void lv_led_set_brightness(lv_obj_t *led, uint8_t bright)

Set the brightness of a LED object

Parametri

- **led** -- pointer to a LED object
- **bright** -- LV_LED_BRIGHT_MIN (max. dark) ... LV_LED_BRIGHT_MAX (max. light)

void lv_led_on(lv_obj_t *led)

Light on a LED

Parametri

led -- pointer to a LED object

void lv_led_off(lv_obj_t *led)

Light off a LED

Parametri

led -- pointer to a LED object

void lv_led_toggle(lv_obj_t *led)

Toggle the state of a LED

Parametri

led -- pointer to a LED object

uint8_t lv_led_get_brightness(const lv_obj_t *obj)

Get the brightness of a LED object

Parametri

obj -- pointer to LED object

Ritorna

bright 0 (max. dark) ... 255 (max. light)

Variabili

const lv_obj_class_t **lv_led_class**

lv_led_private.h

struct **_lv_led_t**

#include <lv_led_private.h> Data of led

Membri Pubblici

lv_obj_t **obj**

lv_color_t **color**

uint8_t **bright**

Current brightness of the LED (0..255)

14.27.19 line

lv_line.h

Funzioni

lv_obj_t ***lv_line_create(lv_obj_t *parent)**

Create a line object

Parametri

- parent** -- pointer to an object, it will be the parent of the new line

Ritorna

- pointer to the created line

void lv_line_set_points(lv_obj_t *obj, const lv_point_precise_t points[], uint32_t point_num)

Set an array of points. The line object will connect these points.

Parametri

- obj** -- pointer to a line object
- points** -- an array of points. Only the address is saved, so the array needs to be alive while the line exists
- point_num** -- number of points in 'point_a'

void lv_line_set_points Mutable(lv_obj_t *obj, lv_point_precise_t points[], uint32_t point_num)

Set a non-const array of points. Identical to `lv_line_set_points` except the array may be retrieved by `lv_line_get_points Mutable`.

Parametri

- obj** -- pointer to a line object
- points** -- a non-const array of points. Only the address is saved, so the array needs to be alive while the line exists.
- point_num** -- number of points in 'point_a'

void lv_line_set_y_invert(lv_obj_t *obj, bool en)

Enable (or disable) the y coordinate inversion. If enabled then y will be subtracted from the height of the object, therefore the y = 0 coordinate will be on the bottom.

Parametri

- obj** -- pointer to a line object
- en** -- true: enable the y inversion, false:disable the y inversion

const lv_point_precise_t *lv_line_get_points(lv_obj_t *obj)

Get the pointer to the array of points.

Parametri

- obj** -- pointer to a line object

Ritorna

- const pointer to the array of points

uint32_t lv_line_get_point_count(lv_obj_t *obj)

Get the number of points in the array of points.

Parametri

- obj** -- pointer to a line object

Ritorna

- number of points in array of points

bool lv_line_is_point_array Mutable(lv_obj_t *obj)

Check the mutability of the stored point array pointer.

Parametri

- obj** -- pointer to a line object

Ritorna

- true: the point array pointer is mutable, false: constant

lv_point_precise_t *lv_line_get_points Mutable(lv_obj_t *obj)

Get a pointer to the mutable array of points or NULL if it is not mutable

Parametri

- obj** -- pointer to a line object

Ritorna

- pointer to the array of points. NULL if not mutable.

bool lv_line_get_y_invert(const lv_obj_t *obj)

Get the y inversion attribute

Parametri

- obj** -- pointer to a line object

Ritorna

true: y inversion is enabled, false: disabled

Variabili

```
const lv_obj_class_t lv_line_class
```

lv_line_private.h

```
struct lv_line_t
#include <lv_line_private.h> Data of line
```

Membri Pubblici

lv_obj_t obj

const lv_point_precise_t *constant

lv_point_precise_t *mut

union lv_line_t point_array

Pointer to an array with the points of the line

uint32_t point_num

Number of points in 'point_array'

uint32_t y_inv

1: y == 0 will be on the bottom

uint32_t point_array_is Mutable

whether the point array is const or mutable

14.27.20 list**lv_list.h****Funzioni**

```
lv_obj_t *lv_list_create(lv_obj_t *parent)
```

Create a list object

Parametri

- **parent** -- pointer to an object, it will be the parent of the new list

Ritorna

pointer to the created list

```
lv_obj_t *lv_list_add_text(lv_obj_t *list, const char *txt)
```

Add text to a list

Parametri

- **list** -- pointer to a list, it will be the parent of the new label
- **txt** -- text of the new label

Ritorna

pointer to the created label

`lv_obj_t *lv_list_add_button(lv_obj_t *list, const void *icon, const char *txt)`

Add button to a list

Parametri

- **list** -- pointer to a list, it will be the parent of the new button
- **icon** -- icon for the button, when NULL it will have no icon
- **txt** -- text of the new button, when NULL no text will be added

Ritorna

pointer to the created button

`const char *lv_list_get_button_text(lv_obj_t *list, lv_obj_t *btn)`

Get text of a given list button

Parametri

- **list** -- pointer to a list
- **btn** -- pointer to the button

Ritorna

text of btn, if btn doesn't have text "" will be returned

`void lv_list_set_button_text(lv_obj_t *list, lv_obj_t *btn, const char *txt)`

Set text of a given list button

Parametri

- **list** -- pointer to a list
- **btn** -- pointer to the button
- **txt** -- pointer to the text

`lv_obj_t *lv_list_add_translation_tag(lv_obj_t *list, const char *tag)`

Add translation tag text to a list

Parametri

- **list** -- pointer to a list, it will be the parent of the new label
- **tag** -- translation tag of the new label

Ritorna

pointer to the created label

`lv_obj_t *lv_list_add_button_translation_tag(lv_obj_t *list, const void *icon, const char *tag)`

Add translation tag button to a list

Parametri

- **list** -- pointer to a list, it will be the parent of the new button
- **icon** -- icon for the button, when NULL it will have no icon
- **tag** -- translation tag of the new button, when NULL no translation tag will be added

Ritorna

pointer to the created button

`void lv_list_set_button_translation_tag(lv_obj_t *list, lv_obj_t *btn, const char *tag)`

Set translation tag text of a given list button

Parametri

- **list** -- pointer to a list
- **btn** -- pointer to the button
- **tag** -- pointer to the translation tag

Variabili

`const lv_obj_class_t lv_list_class`

`const lv_obj_class_t lv_list_text_class`

`const lv_obj_class_t lv_list_button_class`

14.27.21 lottie

lv_lottie.h

Funzioni

`lv_obj_t *lv_lottie_create(lv_obj_t *parent)`

Create a lottie animation

Parametri

- **parent** -- pointer to the parent widget

Ritorna

pointer to the created Lottie animation widget

`void lv_lottie_set_buffer(lv_obj_t *obj, int32_t w, int32_t h, void *buf)`

Set a buffer for the animation. It also defines the size of the animation

Parametri

- **obj** -- pointer to a lottie widget
- **w** -- width of the animation and buffer
- **h** -- height of the animation and buffer
- **buf** -- a static buffer with `width x height x 4` byte size

`void lv_lottie_set_draw_buf(lv_obj_t *obj, lv_draw_buf_t *draw_buf)`

Set a draw buffer for the animation. It also defines the size of the animation

Parametri

- **obj** -- pointer to a lottie widget
- **draw_buf** -- an initialized draw buffer with ARGB8888 color format

`void lv_lottie_set_src_data(lv_obj_t *obj, const void *src, size_t src_size)`

Set the source for the animation as an array

Parametri

- **obj** -- pointer to a lottie widget
- **src** -- the lottie animation converted to an nul terminated array
- **src_size** -- size of the source array in bytes

`void lv_lottie_set_src_file(lv_obj_t *obj, const char *src)`

Set the source for the animation as a path. Lottie doesn't use LVGL's File System API.

Parametri

- **obj** -- pointer to a lottie widget
- **src** -- path to a json file, e.g. "path/to/file.json"

`lv_anim_t *lv_lottie_get_anim(lv_obj_t *obj)`

Get the LVGL animation which controls the lottie animation

Parametri

- **obj** -- pointer to a lottie widget

Ritorna

the LVGL animation

lv_lottie_private.h

`struct lv_lottie_t`

Membri Pubblici

`lv_canvas_t canvas`

`Tvg_Paint *tvg_paint`

```
Tvg_Canvas *tvg_canvas
```

```
Tvg_Animation *tvg_anim
```

```
lv_anim_t *anim
```

```
int32_t last_rendered_time
```

14.27.22 menu

lv_menu.h

Enum

```
enum lv_menu_mode_header_t
```

Values:

```
enumerator LV_MENU_HEADER_TOP_FIXED
```

Header is positioned at the top

```
enumerator LV_MENU_HEADER_TOP_UNFIXED
```

Header is positioned at the top and can be scrolled out of view

```
enumerator LV_MENU_HEADER_BOTTOM_FIXED
```

Header is positioned at the bottom

```
enum lv_menu_mode_root_back_button_t
```

Values:

```
enumerator LV_MENU_ROOT_BACK_BUTTON_DISABLED
```

```
enumerator LV_MENU_ROOT_BACK_BUTTON_ENABLED
```

Funzioni

```
lv_obj_t *lv_menu_create(lv_obj_t *parent)
```

Create a menu object

Parametri

- **parent** -- pointer to an object, it will be the parent of the new menu

Ritorna

- pointer to the created menu

```
lv_obj_t *lv_menu_page_create(lv_obj_t *menu, char const *const title)
```

Create a menu page object.

This call inserts the new page under menu->storage as its parent, which is itself a child of the menu, so the resulting object hierarchy is: menu => storage => new_page where storage is a Base Widget.

Parametri

- **menu** -- pointer to menu object.
- **title** -- pointer to text for title in header (NULL to not display title)

Ritorna

- pointer to the created menu page

`lv_obj_t *lv_menu_cont_create(lv_obj_t *parent)`

Create a menu cont object

Parametri

- **parent** -- pointer to a menu page or menu section object, it will be the parent of the new menu cont object

Ritorna

- pointer to the created menu cont

`lv_obj_t *lv_menu_section_create(lv_obj_t *parent)`

Create a menu section object

Parametri

- **parent** -- pointer to a menu page object, it will be the parent of the new menu section object

Ritorna

- pointer to the created menu section

`lv_obj_t *lv_menu_separator_create(lv_obj_t *parent)`

Create a menu separator object

Parametri

- **parent** -- pointer to a menu page object, it will be the parent of the new menu separator object

Ritorna

- pointer to the created menu separator

`void lv_menu_set_page(lv_obj_t *obj, lv_obj_t *page)`

Set menu page to display in main

Parametri

- **obj** -- pointer to the menu
- **page** -- pointer to the menu page to set (NULL to clear main and clear menu history)

`void lv_menu_set_page_title(lv_obj_t *page, char const *const title)`

Set menu page title

Parametri

- **page** -- pointer to the menu page
- **title** -- pointer to text for title in header (NULL to not display title)

`void lv_menu_set_page_title_static(lv_obj_t *page, char const *const title)`

Set menu page title with a static text. It will not be saved by the label so the 'text' variable has to be 'alive' while the page exists.

Parametri

- **page** -- pointer to the menu page
- **title** -- pointer to text for title in header (NULL to not display title)

`void lv_menu_set_sidebar_page(lv_obj_t *obj, lv_obj_t *page)`

Set menu page to display in sidebar

Parametri

- **obj** -- pointer to the menu
- **page** -- pointer to the menu page to set (NULL to clear sidebar)

`void lv_menu_set_mode_header(lv_obj_t *obj, lv_menu_mode_header_t mode)`

Set the how the header should behave and its position

Parametri

- **obj** -- pointer to a menu
- **mode** -- LV_MENU_HEADER_TOP_FIXED/TOP_UNFIXED/BOTTOM_FIXED

`void lv_menu_set_mode_root_back_button(lv_obj_t *obj, lv_menu_mode_root_back_button_t mode)`

Set whether back button should appear at root

Parametri

- **obj** -- pointer to a menu
- **mode** -- LV_MENU_ROOT_BACK_BUTTON_DISABLED/ENABLED

`void lv_menu_set_load_page_event(lv_obj_t *menu, lv_obj_t *obj, lv_obj_t *page)`

Add menu to the menu item

Parametri

- **menu** -- pointer to the menu
- **obj** -- pointer to the obj
- **page** -- pointer to the page to load when obj is clicked

`lv_obj_t *lv_menu_get_cur_main_page(lv_obj_t *obj)`

Get a pointer to menu page that is currently displayed in main

Parametri

obj -- pointer to the menu

Ritorna

pointer to current page

`lv_obj_t *lv_menu_get_cur_sidebar_page(lv_obj_t *obj)`

Get a pointer to menu page that is currently displayed in sidebar

Parametri

obj -- pointer to the menu

Ritorna

pointer to current page

`lv_obj_t *lv_menu_get_main_header(lv_obj_t *obj)`

Get a pointer to main header obj

Parametri

obj -- pointer to the menu

Ritorna

pointer to main header obj

`lv_obj_t *lv_menu_get_main_header_back_button(lv_obj_t *obj)`

Get a pointer to main header back btn obj

Parametri

obj -- pointer to the menu

Ritorna

pointer to main header back btn obj

`lv_obj_t *lv_menu_get_sidebar_header(lv_obj_t *obj)`

Get a pointer to sidebar header obj

Parametri

obj -- pointer to the menu

Ritorna

pointer to sidebar header obj

`lv_obj_t *lv_menu_get_sidebar_header_back_button(lv_obj_t *obj)`

Get a pointer to sidebar header obj

Parametri

obj -- pointer to the menu

Ritorna

pointer to sidebar header back btn obj

`bool lv_menu_back_button_is_root(lv_obj_t *menu, lv_obj_t *obj)`

Check if an obj is a root back btn

Parametri

- **menu** -- pointer to the menu
- **obj** -- pointer to the back button

Ritorna

true if it is a root back btn

`void lv_menu_clear_history(lv_obj_t *obj)`

Clear menu history

Parametri

obj -- pointer to the menu

Variabili

```
const lv_obj_class_t lv_menu_class
```

```
const lv_obj_class_t lv_menu_page_class
```

```
const lv_obj_class_t lv_menu_cont_class
```

```
const lv_obj_class_t lv_menu_section_class
```

```
const lv_obj_class_t lv_menu_separator_class
```

```
const lv_obj_class_t lv_menu_sidebar_cont_class
```

```
const lv_obj_class_t lv_menu_main_cont_class
```

```
const lv_obj_class_t lv_menu_sidebar_header_cont_class
```

```
const lv_obj_class_t lv_menu_main_header_cont_class
```

lv_menu_private.h

```
struct _lv_menu_load_page_event_data_t
```

Membri Pubblici

```
lv_obj_t *menu
```

```
lv_obj_t *page
```

```
struct _lv_menu_history_t
```

Membri Pubblici

```
lv_obj_t *page
```

```
struct _lv_menu_t
```

Membri Pubblici

```
lv_obj_t obj
```

```
lv_obj_t *storage
```

a pointer to obj that is the parent of all pages not displayed

`lv_obj_t *main`

`lv_obj_t *main_page`

`lv_obj_t *main_header`

`lv_obj_t *main_header_back_btn`

a pointer to obj that on click triggers back btn event handler, can be same as 'main_header'

`lv_obj_t *main_header_title`

`lv_obj_t *sidebar`

`lv_obj_t *sidebar_page`

`lv_obj_t *sidebar_header`

`lv_obj_t *sidebar_header_back_btn`

a pointer to obj that on click triggers back btn event handler, can be same as 'sidebar_header'

`lv_obj_t *sidebar_header_title`

`lv_obj_t *selected_tab`

`lv_ll_t history_ll`

`uint8_t cur_depth`

`uint8_t prev_depth`

`uint8_t sidebar_generated`

`lv_menu_mode_header_t mode_header`

`lv_menu_mode_root_back_button_t mode_root_back_btn`

struct `_lv_menu_page_t`

Membri Pubblici

`lv_obj_t obj`

`char *title`

`bool static_title`

14.27.23 msgbox

lv_msgbox.h

Funzioni

`lv_obj_t *lv_msgbox_create(lv_obj_t *parent)`

Create an empty message box

Parametri

- **parent** -- the parent or NULL to create a modal msgbox

Ritorna

the created message box

`lv_obj_t *lv_msgbox_add_title(lv_obj_t *obj, const char *title)`

Add title to the message box. It also creates a header for the title.

Parametri

- **obj** -- pointer to a message box
- **title** -- the text of the tile

Ritorna

the created title label

`lv_obj_t *lv_msgbox_add_header_button(lv_obj_t *obj, const void *icon)`

Add a button to the header of to the message box. It also creates a header.

Parametri

- **obj** -- pointer to a message box
- **icon** -- the icon of the button

Ritorna

the created button

`lv_obj_t *lv_msgbox_add_text(lv_obj_t *obj, const char *text)`

Add a text to the content area of message box. Multiple texts will be created below each other.

Parametri

- **obj** -- pointer to a message box
- **text** -- text to add

Ritorna

the created button

`lv_obj_t *lv_msgbox_add_footer_button(lv_obj_t *obj, const char *text)`

Add a button to the footer of to the message box. It also creates a footer.

Parametri

- **obj** -- pointer to a message box
- **text** -- the text of the button

Ritorna

the created button

`lv_obj_t *lv_msgbox_add_close_button(lv_obj_t *obj)`

Add a close button to the message box. It also creates a header.

Parametri

- **obj** -- pointer to a message box

Ritorna

the created close button

`lv_obj_t *lv_msgbox_get_header(lv_obj_t *obj)`

Get the header widget

Parametri

- **obj** -- pointer to a message box

Ritorna

the header, or NULL if not exists

`lv_obj_t *lv_msgbox_get_footer(lv_obj_t *obj)`

Get the footer widget

Parametri

obj -- pointer to a message box

Ritorna

the footer, or NULL if not exists

lv_obj_t *lv_msgbox_get_content(lv_obj_t *obj)

Get the content widget

Parametri

obj -- pointer to a message box

Ritorna

the content

lv_obj_t *lv_msgbox_get_title(lv_obj_t *obj)

Get the title label

Parametri

obj -- pointer to a message box

Ritorna

the title, or NULL if it does not exist

void lv_msgbox_close(lv_obj_t *mbox)

Close a message box

Parametri

mbox -- pointer to a message box

void lv_msgbox_close_async(lv_obj_t *mbox)

Close a message box in the next call of the message box

Parametri

mbox -- pointer to a message box

Variabili

const lv_obj_class_t **lv_msgbox_class**

const lv_obj_class_t **lv_msgbox_header_class**

const lv_obj_class_t **lv_msgbox_content_class**

const lv_obj_class_t **lv_msgbox_footer_class**

const lv_obj_class_t **lv_msgbox_header_button_class**

const lv_obj_class_t **lv_msgbox_footer_button_class**

const lv_obj_class_t **lv_msgbox_backdrop_class**

lv_msgbox_private.h

struct **_lv_msgbox_t**

Membri Pubblici

lv_obj_t **obj**

```
lv_obj_t *header
```

```
lv_obj_t *content
```

```
lv_obj_t *footer
```

```
lv_obj_t *title
```

14.27.24 objx_tmpl

lv_objx_tmpl.h

14.27.25 property

lv_obj_property_names.h

GENERATED FILE, DO NOT EDIT IT!

Variabili

```
const lv_property_name_t lv_animimage_property_names[4]
```

```
const lv_property_name_t lv_dropdown_property_names[9]
```

```
const lv_property_name_t lv_image_property_names[11]
```

```
const lv_property_name_t lv_keyboard_property_names[4]
```

```
const lv_property_name_t lv_label_property_names[4]
```

```
const lv_property_name_t lv_obj_property_names[75]
```

```
const lv_property_name_t lv_roller_property_names[3]
```

```
const lv_property_name_t lv_slider_property_names[8]
```

```
const lv_property_name_t lv_style_property_names[121]
```

```
const lv_property_name_t lv_textarea_property_names[15]
```

lv_style_properties.h

GENERATED FILE, DO NOT EDIT IT!

Enum

```
enum _lv_property_style_id_t
```

Values:

enumerator **LV_PROPERTY_STYLE_ALIGN**

enumerator **LV_PROPERTY_STYLE_ANIM**

enumerator **LV_PROPERTY_STYLE_ANIM_DURATION**

enumerator **LV_PROPERTY_STYLE_ARC_COLOR**

enumerator **LV_PROPERTY_STYLE_ARC_IMAGE_SRC**

enumerator **LV_PROPERTY_STYLE_ARC_OPA**

enumerator **LV_PROPERTY_STYLE_ARC_ROUNDED**

enumerator **LV_PROPERTY_STYLE_ARC_WIDTH**

enumerator **LV_PROPERTY_STYLE_BASE_DIR**

enumerator **LV_PROPERTY_STYLE_BG_COLOR**

enumerator **LV_PROPERTY_STYLE_BG_GRAD**

enumerator **LV_PROPERTY_STYLE_BG_GRAD_COLOR**

enumerator **LV_PROPERTY_STYLE_BG_GRAD_DIR**

enumerator **LV_PROPERTY_STYLE_BG_GRAD_OPA**

enumerator **LV_PROPERTY_STYLE_BG_GRAD_STOP**

enumerator **LV_PROPERTY_STYLE_BG_IMAGE_OPA**

enumerator **LV_PROPERTY_STYLE_BG_IMAGE_RECOLOR**

enumerator **LV_PROPERTY_STYLE_BG_IMAGE_RECOLOR_OPA**

enumerator **LV_PROPERTY_STYLE_BG_IMAGE_SRC**

enumerator **LV_PROPERTY_STYLE_BG_IMAGE_TILED**

enumerator **LV_PROPERTY_STYLE_BG_MAIN_OPA**

enumerator **LV_PROPERTY_STYLE_BG_MAIN_STOP**

enumerator **LV_PROPERTY_STYLE_BG_OPA**

enumerator **LV_PROPERTY_STYLE_BITMAP_MASK_SRC**

enumerator **LV_PROPERTY_STYLE_BLEND_MODE**

enumerator **LV_PROPERTY_STYLE_BORDER_COLOR**

enumerator **LV_PROPERTY_STYLE_BORDER_OPA**

enumerator **LV_PROPERTY_STYLE_BORDER_POST**

enumerator **LV_PROPERTY_STYLE_BORDER_SIDE**

enumerator **LV_PROPERTY_STYLE_BORDER_WIDTH**

enumerator **LV_PROPERTY_STYLE_CLIP_CORNER**

enumerator **LV_PROPERTY_STYLE_COLOR_FILTER_DSC**

enumerator **LV_PROPERTY_STYLE_COLOR_FILTER_OPA**

enumerator **LV_PROPERTY_STYLE_FLEX_CROSS_PLACE**

enumerator **LV_PROPERTY_STYLE_FLEX_FLOW**

enumerator **LV_PROPERTY_STYLE_FLEX_GROW**

enumerator **LV_PROPERTY_STYLE_FLEX_MAIN_PLACE**

enumerator **LV_PROPERTY_STYLE_FLEX_TRACK_PLACE**

enumerator **LV_PROPERTY_STYLE_GRID_CELL_COLUMN_POS**

enumerator **LV_PROPERTY_STYLE_GRID_CELL_COLUMN_SPAN**

enumerator **LV_PROPERTY_STYLE_GRID_CELL_ROW_POS**

enumerator **LV_PROPERTY_STYLE_GRID_CELL_ROW_SPAN**

enumerator **LV_PROPERTY_STYLE_GRID_CELL_X_ALIGN**

enumerator **LV_PROPERTY_STYLE_GRID_CELL_Y_ALIGN**

enumerator **LV_PROPERTY_STYLE_GRID_COLUMN_ALIGN**

enumerator **LV_PROPERTY_STYLE_GRID_COLUMN_DSC_ARRAY**

enumerator **LV_PROPERTY_STYLE_GRID_ROW_ALIGN**

enumerator **LV_PROPERTY_STYLE_GRID_ROW_DSC_ARRAY**

enumerator **LV_PROPERTY_STYLE_HEIGHT**

enumerator **LV_PROPERTY_STYLE_IMAGE_COLORKEY**

enumerator **LV_PROPERTY_STYLE_IMAGE_OPA**

enumerator **LV_PROPERTY_STYLE_IMAGE_RECOLOR**

enumerator **LV_PROPERTY_STYLE_IMAGE_RECOLOR_OPA**

enumerator **LV_PROPERTY_STYLE_LAST_BUILT_IN_PROP**

enumerator **LV_PROPERTY_STYLE_LAYOUT**

enumerator **LV_PROPERTY_STYLE_LENGTH**

enumerator **LV_PROPERTY_STYLE_LINE_COLOR**

enumerator **LV_PROPERTY_STYLE_LINE_DASH_GAP**

enumerator **LV_PROPERTY_STYLE_LINE_DASH_WIDTH**

enumerator **LV_PROPERTY_STYLE_LINE_OPA**

enumerator **LV_PROPERTY_STYLE_LINE_ROUNDED**

enumerator **LV_PROPERTY_STYLE_LINE_WIDTH**

enumerator **LV_PROPERTY_STYLE_MARGIN_BOTTOM**

enumerator **LV_PROPERTY_STYLE_MARGIN_LEFT**

enumerator **LV_PROPERTY_STYLE_MARGIN_RIGHT**

enumerator **LV_PROPERTY_STYLE_MARGIN_TOP**

enumerator **LV_PROPERTY_STYLE_MAX_HEIGHT**

enumerator **LV_PROPERTY_STYLE_MAX_WIDTH**

enumerator **LV_PROPERTY_STYLE_MIN_HEIGHT**

enumerator **LV_PROPERTY_STYLE_MIN_WIDTH**

enumerator **LV_PROPERTY_STYLE_OPA**

enumerator **LV_PROPERTY_STYLE_OPA_LAYERED**

enumerator **LV_PROPERTY_STYLE_OUTLINE_COLOR**

enumerator **LV_PROPERTY_STYLE_OUTLINE_OPA**

enumerator **LV_PROPERTY_STYLE_OUTLINE_PAD**

enumerator **LV_PROPERTY_STYLE_OUTLINE_WIDTH**

enumerator **LV_PROPERTY_STYLE_PAD_BOTTOM**

enumerator **LV_PROPERTY_STYLE_PAD_COLUMN**

enumerator **LV_PROPERTY_STYLE_PAD_LEFT**

enumerator **LV_PROPERTY_STYLE_PAD_RADIAL**

enumerator **LV_PROPERTY_STYLE_PAD_RIGHT**

enumerator **LV_PROPERTY_STYLE_PAD_ROW**

enumerator **LV_PROPERTY_STYLE_PAD_TOP**

enumerator **LV_PROPERTY_STYLE_PROP_INV**

enumerator **LV_PROPERTY_STYLE_RADIAL_OFFSET**

enumerator **LV_PROPERTY_STYLE_RADIUS**

enumerator **LV_PROPERTY_STYLE_RECOLOR**

enumerator **LV_PROPERTY_STYLE_RECOLOR_OPA**

enumerator **LV_PROPERTY_STYLE_ROTARY_SENSITIVITY**

enumerator **LV_PROPERTY_STYLE_SHADOW_COLOR**

enumerator **LV_PROPERTY_STYLE_SHADOW_OFFSET_X**

enumerator **LV_PROPERTY_STYLE_SHADOW_OFFSET_Y**

enumerator **LV_PROPERTY_STYLE_SHADOW_OPA**

enumerator **LV_PROPERTY_STYLE_SHADOW_SPREAD**

enumerator **LV_PROPERTY_STYLE_SHADOW_WIDTH**

enumerator **LV_PROPERTY_STYLE_TEXT_ALIGN**

enumerator **LV_PROPERTY_STYLE_TEXT_COLOR**

enumerator **LV_PROPERTY_STYLE_TEXT_DECOR**

enumerator **LV_PROPERTY_STYLE_TEXT_FONT**

enumerator **LV_PROPERTY_STYLE_TEXT_LETTER_SPACE**

enumerator **LV_PROPERTY_STYLE_TEXT_LINE_SPACE**

enumerator **LV_PROPERTY_STYLE_TEXT_OPA**

enumerator **LV_PROPERTY_STYLE_TEXT_OUTLINE_STROKE_COLOR**

enumerator **LV_PROPERTY_STYLE_TEXT_OUTLINE_STROKE_OPA**

enumerator **LV_PROPERTY_STYLE_TEXT_OUTLINE_STROKE_WIDTH**

enumerator **LV_PROPERTY_STYLE_TRANSFORM_HEIGHT**

enumerator **LV_PROPERTY_STYLE_TRANSFORM_PIVOT_X**

enumerator **LV_PROPERTY_STYLE_TRANSFORM_PIVOT_Y**

enumerator **LV_PROPERTY_STYLE_TRANSFORM_ROTATION**

enumerator **LV_PROPERTY_STYLE_TRANSFORM_SCALE_X**

enumerator **LV_PROPERTY_STYLE_TRANSFORM_SCALE_Y**

enumerator **LV_PROPERTY_STYLE_TRANSFORM_SKEW_X**

enumerator **LV_PROPERTY_STYLE_TRANSFORM_SKEW_Y**

enumerator **LV_PROPERTY_STYLE_TRANSFORM_WIDTH**

enumerator **LV_PROPERTY_STYLE_TRANSITION**

enumerator **LV_PROPERTY_STYLE_TRANSLATE_RADIAL**

enumerator **LV_PROPERTY_STYLE_TRANSLATE_X**

enumerator **LV_PROPERTY_STYLE_TRANSLATE_Y**

enumerator **LV_PROPERTY_STYLE_WIDTH**

enumerator **LV_PROPERTY_STYLE_X**

enumerator **LV_PROPERTY_STYLE_Y**

14.27.26 roller

lv_roller.h

Enum

enum **lv_roller_mode_t**

Roller mode.

Values:

enumerator **LV_ROLLER_MODE_NORMAL**

Normal mode (roller ends at the end of the options).

enumerator **LV_ROLLER_MODE_INFINITE**

Infinite mode (roller can be scrolled forever).

enum **_lv_property_roller_id_t**

Values:

enumerator **LV_PROPERTY_ROLLER_OPTIONS**

enumerator **LV_PROPERTY_ROLLER_SELECTED**

enumerator **LV_PROPERTY_ROLLER_VISIBLE_ROW_COUNT**

enumerator **LV_PROPERTY_ROLLER_END**

Funzioni

lv_obj_t *lv_roller_create(lv_obj_t *parent)

Create a roller object

Parametri

parent -- pointer to an object, it will be the parent of the new roller.

Ritorna

pointer to the created roller

void lv_roller_set_options(lv_obj_t *obj, const char *options, lv_roller_mode_t mode)

Set the options on a roller

Parametri

- **obj** -- pointer to roller object
- **options** -- a string with '
' separated options. E.g. "One\nTwo\nThree"
- **mode** -- LV_ROLLER_MODE_NORMAL or LV_ROLLER_MODE_INFINITE

void lv_roller_set_selected(lv_obj_t *obj, uint32_t sel_opt, lv_anim_enable_t anim)

Set the selected option

Parametri

- **obj** -- pointer to a roller object
- **sel_opt** -- index of the selected option (0 ... number of option - 1);
- **anim** -- LV_ANIM_ON: set with animation; LV_ANIM_OFF set immediately

bool lv_roller_set_selected_str(lv_obj_t *obj, const char *sel_opt, lv_anim_enable_t anim)

Sets the given string as the selection on the roller. Does not alter the current selection on failure.

Parametri

- **obj** -- pointer to roller object
- **sel_opt** -- pointer to the string you want to set as an option
- **anim** -- LV_ANIM_ON: set with animation; LV_ANIM_OFF set immediately

Ritorna

true if set successfully and false if the given string does not exist as an option in the roller

void lv_roller_set_visible_row_count(lv_obj_t *obj, uint32_t row_cnt)

Set the height to show the given number of rows (options)

Parametri

- **obj** -- pointer to a roller object
- **row_cnt** -- number of desired visible rows

uint32_t lv_roller_get_selected(const lv_obj_t *obj)

Get the index of the selected option

Parametri

obj -- pointer to a roller object

Ritorna

index of the selected option (0 ... number of option - 1);

void lv_roller_get_selected_str(const lv_obj_t *obj, char *buf, uint32_t buf_size)

Get the current selected option as a string.

Parametri

- **obj** -- pointer to roller object
- **buf** -- pointer to an array to store the string
- **buf_size** -- size of buf in bytes. 0: to ignore it.

const char *lv_roller_get_options(const lv_obj_t *obj)

Get the options of a roller

Parametri

obj -- pointer to roller object

Ritorna

the options separated by '

'-s (E.g. "Option1\nOption2\nOption3")

uint32_t lv_roller_get_option_count(const lv_obj_t *obj)

Get the total number of options

Parametri

obj -- pointer to a roller object

Ritorna

the total number of options

`lv_result_t lv_roller_get_option_str(const lv_obj_t *obj, uint32_t option, char *buf, uint32_t buf_size)`

Get an option as a string.

Parametri

- **obj** -- pointer to roller object
- **option** -- index of chosen option
- **buf** -- pointer to an array to store the string
- **buf_size** -- size of buf in bytes. 0: to ignore it.

Ritorna

`LV_RESULT_OK` if option found

`lv_observer_t *lv_roller_bind_value(lv_obj_t *obj, lv_subject_t *subject)`

Bind an integer Subject to a Roller's value.

Parametri

- **obj** -- pointer to Roller
- **subject** -- pointer to Subject

Ritorna

pointer to newly-created Observer

Variabili

`const lv_obj_class_t lv_roller_class`

lv_roller_private.h

`struct _lv_roller_t`

Membri Pubblici

`lv_obj_t obj`

`uint32_t option_cnt`

Number of options

`uint32_t sel_opt_id`

Index of the current option

`uint32_t sel_opt_id_ori`

Store the original index on focus

`uint32_t inf_page_cnt`

Number of extra pages added to make the roller look infinite

`lv_roller_mode_t mode`

`uint32_t moved`

14.27.27 scala

lv_scale.h

Define

LV_SCALE_TOTAL_TICK_COUNT_DEFAULT

Default value of total minor ticks.

LV_SCALE_MAJOR_TICK_EVERY_DEFAULT

Default value of major tick every nth ticks.

LV_SCALE_LABEL_ENABLED_DEFAULT

Default value of scale label enabled.

LV_SCALE_LABEL_ROTATE_MATCH_TICKS**LV_SCALE_LABEL_ROTATE_KEEP_UPRIGHT****LV_SCALE_ROTATION_ANGLE_MASK****Enum**

enum **lv_scale_mode_t**

Scale mode

Values:

enumerator **LV_SCALE_MODE_HORIZONTAL_TOP**

enumerator **LV_SCALE_MODE_HORIZONTAL_BOTTOM**

enumerator **LV_SCALE_MODE_VERTICAL_LEFT**

enumerator **LV_SCALE_MODE_VERTICAL_RIGHT**

enumerator **LV_SCALE_MODE_ROUND_INNER**

enumerator **LV_SCALE_MODE_ROUND_OUTER**

enumerator **LV_SCALE_MODE_LAST**

Funzioni

lv_obj_t *lv_scale_create(lv_obj_t *parent)

Create an scale object

Parametri

parent -- pointer to an object, it will be the parent of the new scale

Ritorna

pointer to created Scale Widget

void lv_scale_set_mode(lv_obj_t *obj, [lv_scale_mode_t](#) mode)

Set scale mode. See [lv_scale_mode_t](#).

Parametri

- **obj** -- pointer to Scale Widget
- **mode** -- the new scale mode

void lv_scale_set_total_tick_count(lv_obj_t *obj, uint32_t total_tick_count)

Set scale total tick count (including minor and major ticks).

Parametri

- **obj** -- pointer to Scale Widget
- **total_tick_count** -- New total tick count

void lv_scale_set_major_tick_every(lv_obj_t *obj, uint32_t major_tick_every)

Sets how often major ticks are drawn.

Parametri

- **obj** -- pointer to Scale Widget
- **major_tick_every** -- the new count for major tick drawing

void lv_scale_set_label_show(lv_obj_t *obj, bool show_label)

Sets label visibility.

Parametri

- **obj** -- pointer to Scale Widget
- **show_label** -- true/false to enable tick label

void lv_scale_set_range(lv_obj_t *obj, int32_t min, int32_t max)

Set minimum and maximum values on Scale.

Parametri

- **obj** -- pointer to Scale Widget
- **min** -- minimum value of Scale
- **max** -- maximum value of Scale

void lv_scale_set_min_value(lv_obj_t *obj, int32_t min)

Set minimum values on Scale.

Parametri

- **obj** -- pointer to Scale Widget
- **min** -- minimum value of Scale

void lv_scale_set_max_value(lv_obj_t *obj, int32_t max)

Set maximum values on Scale.

Parametri

- **obj** -- pointer to Scale Widget
- **min** -- minimum value of Scale

void lv_scale_set_angle_range(lv_obj_t *obj, uint32_t angle_range)

Set angle between the low end and the high end of the Scale. (Applies only to round Scales.)

Parametri

- **obj** -- pointer to Scale Widget
- **max_angle** -- angle in degrees from Scale minimum where top end of Scale will be drawn

void lv_scale_set_rotation(lv_obj_t *obj, int32_t rotation)

Set angular offset from the 3-o'clock position of the low end of the Scale. (Applies only to round Scales.)

Parametri

- **obj** -- pointer to Scale Widget
- **rotation** -- clockwise angular offset (in degrees) from the 3-o'clock position of the low end of the scale; negative and >360 values are first normalized to range [0..360]. Examples:

- 0 = 3 o'clock (right side)
- 30 = 4 o'clock
- 60 = 5 o'clock
- 90 = 6 o'clock
- 135 = midway between 7 and 8 o'clock (default)
- 180 = 9 o'clock

- 270 = 12 o'clock
- 300 = 1 o'clock
- 330 = 2 o'clock
- -30 = 2 o'clock
- 390 = 4 o'clock

```
void lv_scale_set_line_needle_value(lv_obj_t *obj, lv_obj_t *needle_line, int32_t needle_length, int32_t value)
```

Point line needle to specified value.

Parametri

- **obj** -- pointer to Scale Widget
- **needle_line** -- needle_line of the Scale. The line points will be allocated and managed by the Scale unless the line point array was previously set using `lv_line_set_points Mutable`.
- **needle_length** -- length of the needle
 - needle_length>0: needle_length=needle_length;
 - needle_length<0: needle_length=radius-|needle_length|;
- **value** -- Scale value needle will point to

```
void lv_scale_set_image_needle_value(lv_obj_t *obj, lv_obj_t *needle_img, int32_t value)
```

Point image needle to specified value; image must point to the right. E.g. -O----->

Parametri

- **obj** -- pointer to Scale Widget
- **needle_img** -- pointer to needle's Image
- **value** -- Scale value needle will point to

```
void lv_scale_set_text_src(lv_obj_t *obj, const char *txt_src[])
```

Set custom text source for major ticks labels.

Parametri

- **obj** -- pointer to Scale Widget
- **txt_src** -- pointer to an array of strings which will be display at major ticks; last element must be a NULL pointer.

```
void lv_scale_set_post_draw(lv_obj_t *obj, bool en)
```

Draw Scale after all its children are drawn.

Parametri

- **obj** -- pointer to Scale Widget
- **en** -- true: enable post draw

```
void lv_scale_set_draw_ticks_on_top(lv_obj_t *obj, bool en)
```

Draw Scale ticks on top of all other parts.

Parametri

- **obj** -- pointer to Scale Widget
- **en** -- true: enable draw ticks on top of all parts

```
lv_scale_section_t *lv_scale_add_section(lv_obj_t *obj)
```

Add a Section to specified Scale. Section will not be drawn until a valid range is set for it using `lv_scale_set_section_range()`.

Parametri

obj -- pointer to Scale Widget

Ritorna

pointer to new Section

```
void lv_scale_section_set_range(lv_scale_section_t *section, int32_t min, int32_t max)
```

DEPRECATED, use `lv_scale_set_section_range` instead. Set range for specified Scale Section

Parametri

- **section** -- pointer to Section

- **range_min** -- Section new minimum value
- **range_max** -- Section new maximum value

void lv_scale_set_section_range(lv_obj_t *scale, lv_scale_section_t *section, int32_t min, int32_t max)

Set the range of a scale section

Parametri

- **scale** -- pointer to scale
- **section** -- pointer to section
- **range_min** -- the section's new minimum value
- **range_max** -- the section's new maximum value

void lv_scale_set_section_min_value(lv_obj_t *scale, lv_scale_section_t *section, int32_t min)

Set the minimum value of a scale section

Parametri

- **scale** -- pointer to scale
- **section** -- pointer to section
- **min** -- the section's new minimum value

void lv_scale_set_section_max_value(lv_obj_t *scale, lv_scale_section_t *section, int32_t max)

Set the maximum value of a scale section

Parametri

- **scale** -- pointer to scale
- **section** -- pointer to section
- **max** -- the section's new maximum value

void lv_scale_section_set_style(lv_scale_section_t *section, *lv_part_t* part, *lv_style_t* *section_part_style)

DEPRECATED, use lv_scale_set_section_style_main/indicator/items instead. Set style for specified part of Section.

Parametri

- **section** -- pointer to Section
- **part** -- the part of the Scale the style will apply to, e.g. LV_PART_INDICATOR
- **section_part_style** -- pointer to style to apply

void lv_scale_set_section_style_main(lv_obj_t *scale, lv_scale_section_t *section, const *lv_style_t* *style)

Set the style of the line on a section.

Parametri

- **scale** -- pointer to scale
- **section** -- pointer to section
- **style** -- point to a style

void lv_scale_set_section_style_indicator(lv_obj_t *scale, lv_scale_section_t *section, const *lv_style_t* *style)

Set the style of the major ticks and label on a section.

Parametri

- **scale** -- pointer to scale
- **section** -- pointer to section
- **style** -- point to a style

void lv_scale_set_section_style_items(lv_obj_t *scale, lv_scale_section_t *section, const *lv_style_t* *style)

Set the style of the minor ticks on a section.

Parametri

- **scale** -- pointer to scale
- **section** -- pointer to section
- **style** -- point to a style

***lv_scale_mode_t* lv_scale_get_mode(lv_obj_t *obj)**

Get scale mode. See *lv_scale_mode_t*

Parametri

- **obj** -- pointer to Scale Widget

Ritorna
 Scale mode

```
int32_t lv_scale_get_total_tick_count(lv_obj_t *obj)
```

Get scale total tick count (including minor and major ticks)

Parametri

- obj** -- pointer to Scale Widget

Ritorna
 Scale total tick count

```
int32_t lv_scale_get_major_tick_every(lv_obj_t *obj)
```

Get how often the major tick will be drawn

Parametri

- obj** -- pointer to Scale Widget

Ritorna
 Scale major tick every count

```
int32_t lv_scale_get_rotation(lv_obj_t *obj)
```

Get angular location of low end of Scale.

Parametri

- obj** -- pointer to Scale Widget

Ritorna
 Scale low end angular location

```
bool lv_scale_get_label_show(lv_obj_t *obj)
```

Gets label visibility

Parametri

- obj** -- pointer to Scale Widget

Ritorna
 true if tick label is enabled, false otherwise

```
uint32_t lv_scale_get_angle_range(lv_obj_t *obj)
```

Get Scale's range in degrees

Parametri

- obj** -- pointer to Scale Widget

Ritorna
 Scale's angle_range

```
int32_t lv_scale_get_range_min_value(lv_obj_t *obj)
```

Get minimum value for Scale

Parametri

- obj** -- pointer to Scale Widget

Ritorna
 Scale's minimum value

```
int32_t lv_scale_get_range_max_value(lv_obj_t *obj)
```

Get maximum value for Scale

Parametri

- obj** -- pointer to Scale Widget

Ritorna
 Scale's maximum value

```
lv_observer_t *lv_scale_bind_section_min_value(lv_obj_t *obj, lv_scale_section_t *section, lv_subject_t *subject)
```

Bind an integer subject to a scales section minimum value

Parametri

- **obj** -- pointer to a Scale
- **section** -- pointer to a Scale section
- **subject** -- pointer to a Subject

Ritorna

pointer to newly-created Observer

```
lv_observer_t *lv_scale_bind_section_max_value(lv_obj_t *obj, lv_scale_section_t *section, lv_subject_t *subject)
```

Bind an integer subject to a scales section maximum value

Parametri

- **obj** -- pointer to an Scale
- **section** -- pointer to a Scale section
- **subject** -- pointer to a Subject

Ritorna

pointer to newly-created Observer

Variabili

```
const lv_obj_class_t lv_scale_class
```

lv_scale_private.h

```
struct _lv_scale_section_t
```

Membri Pubblici

```
const lv_style_t *main_style
```

Style to use for MAIN part(s) of scale when it falls within this section's range

```
const lv_style_t *indicator_style
```

Style to use for INDICATOR part(s) of scale when it falls within this section's range

```
const lv_style_t *items_style
```

Style to use for ITEMS part(s) of scale when it falls within this section's range

```
int32_t range_min
```

Scale parts with value >= this value will be drawn using applicable style.

```
int32_t range_max
```

Scale parts with value <= this value will be drawn using applicable style.

```
uint32_t first_tick_idx_in_section
```

Internal (set during drawing): Tick index of first tick that falls within this section; LV_SCALE_TICK_IDX_DEFAULT_ID if section contains no ticks.

```
uint32_t last_tick_idx_in_section
```

Internal (set during drawing): Tick index of last tick that falls within this section; LV_SCALE_TICK_IDX_DEFAULT_ID if section contains no ticks.

```
int32_t first_tick_in_section_width
```

Internal (set during drawing)

```
int32_t last_tick_in_section_width
```

Internal (set during drawing)

lv_point_t **first_tick_in_section**

Internal (set during drawing)

lv_point_t **last_tick_in_section**

Internal (set during drawing)

uint32_t first_tick_idx_is_major

Internal (set during drawing): true if `first_tick_idx_in_section` represents a major tick.

uint32_t last_tick_idx_is_major

Internal (set during drawing): true if `last_tick_idx_in_section` represents a major tick.

struct **_lv_scale_t**

Membri Pubblici

lv_obj_t **obj**

Base Widget part of Scale

lv_ll_t **section_ll**

Linked list for the sections (stores `lv_scale_section_t`)

const char **txt_src

Optional list of text strings for major ticks when custom labels are provided.

lv_scale_mode_t **mode**

Orientation and layout of scale.

int32_t range_min

Scale's minimum value

int32_t range_max

Scale's maximum value

uint32_t total_tick_count

Total number of ticks (major and minor)

uint32_t major_tick_every

Frequency of major ticks to minor ticks

uint32_t label_enabled

Draw labels for major ticks?

uint32_t post_draw

false: drawing occurs during `LV_EVENT_DRAW_MAIN`; true : drawing occurs during `LV_EVENT_DRAW_POST`.

uint32_t draw_ticks_on_top

Draw ticks on top of main line?

uint32_t angle_range

Degrees between low end and high end of scale

int32_t rotation

Clockwise angular offset from 3-o'clock position of low end of scale

int32_t custom_label_cntNumber of custom labels provided in `txt_src`**int32_t last_tick_width**

Width of last tick in pixels

int32_t first_tick_width

Width of first tick in pixels

14.27.28 slider

lv_slider.h**Enum**enum **lv_slider_mode_t***Values:*enumerator **LV_SLIDER_MODE_NORMAL**enumerator **LV_SLIDER_MODE_SYMMETRICAL**enumerator **LV_SLIDER_MODE_RANGE**enum **lv_slider_orientation_t***Values:*enumerator **LV_SLIDER_ORIENTATION_AUTO**enumerator **LV_SLIDER_ORIENTATION_HORIZONTAL**enumerator **LV_SLIDER_ORIENTATION_VERTICAL**enum **_lv_property_slider_id_t***Values:*enumerator **LV_PROPERTY_SLIDER_VALUE**enumerator **LV_PROPERTY_SLIDER_LEFT_VALUE**enumerator **LV_PROPERTY_SLIDER_RANGE**

enumerator **LV_PROPERTY_SLIDER_MIN_VALUE**

enumerator **LV_PROPERTY_SLIDER_MAX_VALUE**

enumerator **LV_PROPERTY_SLIDER_MODE**

enumerator **LV_PROPERTY_SLIDER_IS_DRAGGED**

enumerator **LV_PROPERTY_SLIDER_IS_SYMMETRICAL**

enumerator **LV_PROPERTY_SLIDER_END**

Funzioni

`lv_obj_t *lv_slider_create(lv_obj_t *parent)`

Create a slider object

Parametri

- **parent** -- pointer to an object, it will be the parent of the new slider.

Ritorna

- pointer to the created slider

`void lv_slider_set_value(lv_obj_t *obj, int32_t value, lv_anim_enable_t anim)`

Set a new value on the slider

Parametri

- **obj** -- pointer to a slider object
- **value** -- the new value
- **anim** -- LV_ANIM_ON: set the value with an animation; LV_ANIM_OFF: change the value immediately

`void lv_slider_set_start_value(lv_obj_t *obj, int32_t value, lv_anim_enable_t anim)`

Set a new value for the left knob of a slider

Parametri

- **obj** -- pointer to a slider object
- **value** -- new value
- **anim** -- LV_ANIM_ON: set the value with an animation; LV_ANIM_OFF: change the value immediately

`void lv_slider_set_range(lv_obj_t *obj, int32_t min, int32_t max)`

Set the minimum and the maximum values of a bar

Parametri

- **obj** -- pointer to the slider object
- **min** -- minimum value
- **max** -- maximum value

`void lv_slider_set_min_value(lv_obj_t *obj, int32_t min)`

Set the minimum values of a bar

Parametri

- **obj** -- pointer to the slider object
- **min** -- minimum value

`void lv_slider_set_max_value(lv_obj_t *obj, int32_t max)`

Set the maximum values of a bar

Parametri

- **obj** -- pointer to the slider object
- **max** -- maximum value

`void lv_slider_set_mode(lv_obj_t *obj, lv_slider_mode_t mode)`

Set the mode of slider.

Parametri

- **obj** -- pointer to a slider object
- **mode** -- the mode of the slider. See `lv_slider_mode_t`

`void lv_slider_set_orientation(lv_obj_t *obj, lv_slider_orientation_t orientation)`

Set the orientation of slider.

Parametri

- **obj** -- pointer to a slider object
- **orientation** -- slider orientation from `lv_slider_orientation_t`

`int32_t lv_slider_get_value(const lv_obj_t *obj)`

Get the value of the main knob of a slider

Parametri

obj -- pointer to a slider object

Ritorna

the value of the main knob of the slider

`int32_t lv_slider_get_left_value(const lv_obj_t *obj)`

Get the value of the left knob of a slider

Parametri

obj -- pointer to a slider object

Ritorna

the value of the left knob of the slider

`int32_t lv_slider_get_min_value(const lv_obj_t *obj)`

Get the minimum value of a slider

Parametri

obj -- pointer to a slider object

Ritorna

the minimum value of the slider

`int32_t lv_slider_get_max_value(const lv_obj_t *obj)`

Get the maximum value of a slider

Parametri

obj -- pointer to a slider object

Ritorna

the maximum value of the slider

`bool lv_slider_is_dragged(const lv_obj_t *obj)`

Give the slider is being dragged or not

Parametri

obj -- pointer to a slider object

Ritorna

true: drag in progress false: not dragged

`lv_slider_mode_t lv_slider_get_mode(lv_obj_t *slider)`

Get the mode of the slider.

Parametri

slider -- pointer to a slider object

Ritorna

see `lv_slider_mode_t`

`lv_slider_orientation_t lv_slider_get_orientation(lv_obj_t *slider)`

Get the orientation of slider.

Parametri

obj -- pointer to a slider object

Ritorna

slider orientation from `lv_slider_orientation_t`

`bool lv_slider_is_symmetrical(lv_obj_t *obj)`

Give the slider is in symmetrical mode or not

Parametri

- `obj` -- pointer to slider object

Ritorna

true: in symmetrical mode false : not in

`lv_observer_t *lv_slider_bind_value(lv_obj_t *obj, lv_subject_t *subject)`

Bind an integer or float Subject to a Slider's value.

Parametri

- `obj` -- pointer to Slider
- `subject` -- pointer to Subject

Ritorna

pointer to newly-created Observer

Variabili

`const lv_obj_class_t lv_slider_class`

lv_slider_private.h

`struct _lv_slider_t`

Membri Pubblici

`lv_bar_t bar`

Add the ancestor's type first

`lv_area_t left_knob_area`

`lv_area_t right_knob_area`

`lv_point_t pressed_point`

`int32_t *value_to_set`

Which bar value to set

`uint8_t dragging`

1: the slider is being dragged

`uint8_t left_knob_focus`

1: with encoder now the right knob can be adjusted

14.27.29 span

lv_span.h

Typedef

`typedef struct _lv_span_coords_t lv_span_coords_t`

Coords of a span

Enum

enum **lv_span_overflow_t**

Values:

enumerator **LV_SPAN_OVERFLOW_CLIP**

enumerator **LV_SPAN_OVERFLOW_ELLIPSIS**

enumerator **LV_SPAN_OVERFLOW_LAST**

Fence member

enum **lv_span_mode_t**

Values:

enumerator **LV_SPAN_MODE_FIXED**

fixed the obj size

enumerator **LV_SPAN_MODE_EXPAND**

Expand the object size to the text size

enumerator **LV_SPAN_MODE_BREAK**

Keep width, break the too long lines and expand height

enumerator **LV_SPAN_MODE_LAST**

Fence member

Funzioni

void **lv_span_stack_init**(void)

void **lv_span_stack_deinit**(void)

lv_obj_t ***lv_spangroup_create**(lv_obj_t *parent)

Create a spangroup object

Parametri

parent -- pointer to an object, it will be the parent of the new spangroup

Ritorna

pointer to the created spangroup

lv_span_t ***lv_spangroup_add_span**(lv_obj_t *obj)

Create a span string descriptor and add to spangroup.

Parametri

obj -- pointer to a spangroup object.

Ritorna

pointer to the created span.

void **lv_spangroup_delete_span**(lv_obj_t *obj, lv_span_t *span)

Remove the span from the spangroup and free memory.

Nota

Note that before calling `lv_spangroup_delete_span` `lv_observer_remove` needs to be called manually as LVGL can't remove the binding automatically.

Parametri

- **obj** -- pointer to a spangroup object.
- **span** -- pointer to a span.

`void lv_span_set_text(lv_span_t *span, const char *text)`

Set a new text for a span. Memory will be allocated to store the text by the span. As the spangroup is not passed a redraw (invalidation) can't be triggered automatically. Therefore `lv_spangroup_refresh(spangroup)` needs to be called manually,

Parametri

- **span** -- pointer to a span.
- **text** -- pointer to a text.

`void lv_span_set_text_fmt(lv_span_t *span, const char *fmt, ...)`

Set a new text for a span using a printf-like formatting string. Memory will be allocated to store the text by the span. As the spangroup is not passed a redraw (invalidation) can't be triggered automatically. Therefore `lv_spangroup_refresh(spangroup)` needs to be called manually,

Parametri

- **span** -- pointer to a span.
- **fmt** -- printf-like format string

`void lv_span_set_text_static(lv_span_t *span, const char *text)`

Set a static text. It will not be saved by the span so the 'text' variable has to be 'alive' while the span exist. As the spangroup is not passed a redraw (invalidation) can't be triggered automatically. Therefore `lv_spangroup_refresh(spangroup)` needs to be called manually,

Set a static text. It will not be saved by the span so the 'text' variable has to be 'alive' while the span exist.

Parametri

- **span** -- pointer to a span.
- **text** -- pointer to a text.
- **span** -- pointer to a span.
- **text** -- pointer to a text.

`void lv_spangroup_set_span_text(lv_obj_t *obj, lv_span_t *span, const char *text)`

Set a new text for a span. Memory will be allocated to store the text by the span.

Parametri

- **obj** -- pointer to a spangroup widget.
- **span** -- pointer to a span.
- **text** -- pointer to a text.

`void lv_spangroup_set_span_text_static(lv_obj_t *obj, lv_span_t *span, const char *text)`

Set a new text for a span. Memory will be allocated to store the text by the span.

Parametri

- **obj** -- pointer to a spangroup widget.
- **span** -- pointer to a span.
- **text** -- pointer to a text.

`void lv_spangroup_set_span_text_fmt(lv_obj_t *obj, lv_span_t *span, const char *fmt, ...)`

Set a new text for a span using a printf-like formatting string. Memory will be allocated to store the text by the span.

Parametri

- **obj** -- pointer to a spangroup widget.
- **span** -- pointer to a span.
- **fmt** -- printf-like format string

`void lv_spangroup_set_span_style(lv_obj_t *obj, lv_span_t *span, const lv_style_t *style)`

Copy all style properties of style to the bbuilt-in static style of the span.

Parametri

- **obj** -- pointer_to a spangroup
- **span** -- pointer to a span.
- **style** -- pointer to a style to copy into the span's built-in style

`void lv_spangroup_set_align(lv_obj_t *obj, lv_text_align_t align)`

DEPRECATED. Use the text_align style property instead Set the align of the spangroup.

Parametri

- **obj** -- pointer to a spangroup object.
- **align** -- see [lv_text_align_t](#) for details.

`void lv_spangroup_set_overflow(lv_obj_t *obj, lv_span_overflow_t overflow)`

Set the overflow of the spangroup.

Parametri

- **obj** -- pointer to a spangroup object.
- **overflow** -- see [lv_span_overflow_t](#) for details.

`void lv_spangroup_set_indent(lv_obj_t *obj, int32_t indent)`

Set the indent of the spangroup.

Parametri

- **obj** -- pointer to a spangroup object.
- **indent** -- the first line indentation

`void lv_spangroup_set_mode(lv_obj_t *obj, lv_span_mode_t mode)`

DEPRECATED, set the width to LV_SIZE_CONTENT or fixed value to control expanding/wrapping" Set the mode of the spangroup.

Parametri

- **obj** -- pointer to a spangroup object.
- **mode** -- see [lv_span_mode_t](#) for details.

`void lv_spangroup_set_max_lines(lv_obj_t *obj, int32_t lines)`

Set maximum lines of the spangroup.

Parametri

- **obj** -- pointer to a spangroup object.
- **lines** -- max lines that can be displayed in LV_SPAN_MODE_BREAK mode. < 0 means no limit.

`lv_style_t *lv_span_get_style(lv_span_t *span)`

Get a pointer to the style of a span's built-in style. Any lv_style_set_... functions can be applied on the returned style.

Parametri

span -- pointer to the span

Ritorna

pointer to the style. (valid as long as the span is valid)

`const char *lv_span_get_text(lv_span_t *span)`

Get a pointer to the text of a span

Parametri

span -- pointer to the span

Ritorna

pointer to the text

`lv_span_t *lv_spangroup_get_child(const lv_obj_t *obj, int32_t id)`

Get a spangroup child by its index.

Parametri

- **obj** -- The spangroup object
- **id** -- the index of the child. 0: the oldest (firstly created) child 1: the second oldest child count-1: the youngest -1: the youngest -2: the second youngest

Ritorna

The child span at index **id**, or NULL if the ID does not exist

`uint32_t lv_spangroup_get_span_count(const lv_obj_t *obj)`

Get number of spans

Parametri

obj -- the spangroup object to get the child count of.

Ritorna

the span count of the spangroup.

*lv_text_align_t lv_spangroup_get_align(lv_obj_t *obj)*

Get the align of the spangroup.

Parametri

obj -- pointer to a spangroup object.

Ritorna

the align value.

*lv_span_overflow_t lv_spangroup_get_overflow(lv_obj_t *obj)*

Get the overflow of the spangroup.

Parametri

obj -- pointer to a spangroup object.

Ritorna

the overflow value.

*int32_t lv_spangroup_get_indent(lv_obj_t *obj)*

Get the indent of the spangroup.

Parametri

obj -- pointer to a spangroup object.

Ritorna

the indent value.

*lv_span_mode_t lv_spangroup_get_mode(lv_obj_t *obj)*

Get the mode of the spangroup.

Parametri

obj -- pointer to a spangroup object.

*int32_t lv_spangroup_get_max_lines(lv_obj_t *obj)*

Get maximum lines of the spangroup.

Parametri

obj -- pointer to a spangroup object.

Ritorna

the max lines value.

*int32_t lv_spangroup_get_max_line_height(lv_obj_t *obj)*

Get max line height of all span in the spangroup.

Parametri

obj -- pointer to a spangroup object.

*uint32_t lv_spangroup_get_expand_width(lv_obj_t *obj, uint32_t max_width)*

Get the text content width when all span of spangroup on a line.

Parametri

- **obj** -- pointer to a spangroup object.
- **max_width** -- if text content width \geq max_width, return max_width to reduce computation, if max_width == 0, returns the text content width.

Ritorna

text content width or max_width.

*int32_t lv_spangroup_get_expand_height(lv_obj_t *obj, int32_t width)*

Get the text content height with width fixed.

Parametri

- **obj** -- pointer to a spangroup object.
- **width** -- the width of the span group.

*lv_span_coords_t lv_spangroup_get_span_coords(lv_obj_t *obj, const lv_span_t *span)*

Get the span's coords in the spangroup.

Nota

Before calling this function, please make sure that the layout of span group has been updated. Like calling `lv_obj_update_layout()` like function.

```
+-----+
|Heading +-----+
| | Pos | | Heading |
+-----+-----+
| |
| |
| |
| |
| Middle +-----+
| | | Trailing ||
| +| Pos || |
| | +-----+
+-----+-----+
| Trailing |
+-----+
```

Parametri

- **obj** -- pointer to a spangroup object.
- **span** -- pointer to a span.

Ritorna

the span's coords in the spangroup.

`lv_span_t *lv_spangroup_get_span_by_point(lv_obj_t *obj, const lv_point_t *point)`

Get the span object by point.

Parametri

- **obj** -- pointer to a spangroup object.
- **point** -- pointer to point containing absolute coordinates

Ritorna

pointer to the span under the point or NULL if not found.

`void lv_spangroup_refresh(lv_obj_t *obj)`

Update the mode of the spangroup.

Parametri

- **obj** -- pointer to a spangroup object.

`lv_observer_t *lv_spangroup_bind_span_text(lv_obj_t *obj, lv_span_t *span, lv_subject_t *subject, const char *fmt)`

Bind an integer, string, or pointer Subject to a Spangroup's Span.

Nota

If **fmt** == NULL strings and pointers (\0 terminated string) will be shown as text as they are, integers as d, floats as %0.1f

Parametri

- **obj** -- pointer to Spangroup
- **span** -- pointer to Span
- **subject** -- pointer to Subject
- **fmt** -- optional printf-like format string with 1 format specifier (e.g. "%d °C") or NULL to bind to the value directly.

Ritorna

pointer to newly-created Observer

Variabili

`const lv_obj_class_t lv_spangroup_class`

```
struct _lv_span_coords_t
#include <lv_span.h> Coords of a span
```

Membri Pubblici

lv_area_t **heading**

lv_area_t **middle**

lv_area_t **trailing**

lv_span_private.h

struct **_lv_span_t**

Membri Pubblici

char ***txt**

a pointer to display text

lv_style_t **style**

display text style

uint32_t **static_flag**

the text is static flag

lv_point_t **trailing_pos**

int32_t **trailing_height**

struct **_lv_spangroup_t**

#include <lv_span_private.h> Data of label

Membri Pubblici

lv_obj_t obj

int32_t **lines**

int32_t **indent**

first line indent

int32_t **cache_w**

the cache automatically calculates the width

int32_t **cache_h**

similar cache_w

lv_ll_t **child_ll**

```
uint32_t overflow
details see lv\_span\_overflow\_t

uint32_t refresh
the spangroup need refresh cache_w and cache_h
```

14.27.30 spinbox

lv_spinbox.h

Define

LV_SPINBOX_MAX_DIGIT_COUNT

Funzioni

lv_obj_t *lv_spinbox_create(lv_obj_t *parent)

Create a spinbox object

Parametri

- **parent** -- pointer to an object, it will be the parent of the new spinbox

Ritorna

- pointer to the created spinbox

void lv_spinbox_set_value(lv_obj_t *obj, int32_t v)

Set spinbox value

Parametri

- **obj** -- pointer to spinbox
- **v** -- value to be set

void lv_spinbox_set_rollover(lv_obj_t *obj, bool rollover)

Set spinbox rollover function

Parametri

- **obj** -- pointer to spinbox
- **rollover** -- true or false to enable or disable (default)

void lv_spinbox_set_digit_format(lv_obj_t *obj, uint32_t digit_count, uint32_t sep_pos)

Set spinbox digit format (digit count and decimal format)

Parametri

- **obj** -- pointer to spinbox
- **digit_count** -- number of digit excluding the decimal separator and the sign
- **sep_pos** -- number of digit before the decimal point. If 0, decimal point is not shown

void lv_spinbox_set_digit_count(lv_obj_t *obj, uint32_t digit_count)

Set the number of digits

Parametri

- **obj** -- pointer to spinbox
- **digit_count** -- number of digits

void lv_spinbox_set_dec_point_pos(lv_obj_t *obj, uint32_t dec_point_pos)

Set the position of the decimal point

Parametri

- **obj** -- pointer to spinbox
- **dec_point_pos** -- 0: there is no separator, 2: two integer digits

void lv_spinbox_set_step(lv_obj_t *obj, uint32_t step)

Set spinbox step

Parametri

- **obj** -- pointer to spinbox

- **step** -- steps on increment/decrement. Can be 1, 10, 100, 1000, etc the digit that will change.

void lv_spinbox_set_range(lv_obj_t *obj, int32_t min_value, int32_t max_value)

Set spinbox value range

Parametri

- **obj** -- pointer to spinbox
- **min_value** -- minimum value, inclusive
- **max_value** -- maximum value, inclusive

void lv_spinbox_set_min_value(lv_obj_t *obj, int32_t min_value)

Set the minimum value

Parametri

- **obj** -- pointer to spinbox
- **min_value** -- the minimum value

void lv_spinbox_set_max_value(lv_obj_t *obj, int32_t max_value)

Set the maximum value

Parametri

- **obj** -- pointer to spinbox
- **max_value** -- the maximum value

void lv_spinbox_set_cursor_pos(lv_obj_t *obj, uint32_t pos)

Set cursor position to a specific digit for edition

Parametri

- **obj** -- pointer to spinbox
- **pos** -- selected position in spinbox

void lv_spinbox_set_digit_step_direction(lv_obj_t *obj, lv_dir_t direction)

Set direction of digit step when clicking an encoder button while in editing mode

Parametri

- **obj** -- pointer to spinbox
- **direction** -- the direction (LV_DIR_RIGHT or LV_DIR_LEFT)

bool lv_spinbox_get_rollover(lv_obj_t *obj)

Get spinbox rollover function status

Parametri

obj -- pointer to spinbox

int32_t lv_spinbox_get_value(lv_obj_t *obj)

Get the spinbox numeral value (user has to convert to float according to its digit format)

Parametri

obj -- pointer to spinbox

Ritorna

value integer value of the spinbox

int32_t lv_spinbox_get_step(lv_obj_t *obj)

Get the spinbox step value (user has to convert to float according to its digit format)

Parametri

obj -- pointer to spinbox

Ritorna

value integer step value of the spinbox

void lv_spinbox_step_next(lv_obj_t *obj)

Select next lower digit for edition by dividing the step by 10

Parametri

obj -- pointer to spinbox

void lv_spinbox_step_prev(lv_obj_t *obj)

Select next higher digit for edition by multiplying the step by 10

Parametri
obj -- pointer to spinbox

void lv_spinbox_increment(lv_obj_t *obj)
Increment spinbox value by one step

Parametri
obj -- pointer to spinbox

void lv_spinbox_decrement(lv_obj_t *obj)
Decrement spinbox value by one step

Parametri
obj -- pointer to spinbox

lv_observer_t *lv_spinbox_bind_value(lv_obj_t *obj, lv_subject_t *subject)
Bind an integer subject to a Spinbox's value.

Parametri

- **obj** -- pointer to Spinbox
- **subject** -- pointer to Subject

Ritorna
pointer to newly-created Observer

Variabili

const lv_obj_class_t **lv_spinbox_class**

lv_spinbox_private.h

```
struct _lv_spinbox_t
#include <lv_spinbox_private.h> Data of spinbox
```

Membri Pubblici

lv_textarea_t ta
Ext. of ancestor

int32_t value

int32_t range_max

int32_t range_min

int32_t step

uint32_t digit_count

uint32_t dec_point_pos
if 0, there is no separator and the number is an integer

uint32_t rollover
Set to true for rollover functionality

uint32_t digit_step_dir
the direction the digit will step on encoder button press when editing

14.27.31 spinner

lv_spinner.h

Funzioni

`lv_obj_t *lv_spinner_create(lv_obj_t *parent)`

Create a spinner widget

Parametri

• **parent** -- pointer to an object, it will be the parent of the new spinner.

Ritorna

the created spinner

`void lv_spinner_set_anim_params(lv_obj_t *obj, uint32_t t, uint32_t angle)`

Set the animation time and arc length of the spinner. The animation is suited for angle values between 180 and 360.

Parametri

- **obj** -- pointer to a spinner
- **t** -- the animation time in milliseconds
- **angle** -- the angle of the arc in degrees

`void lv_spinner_set_anim_duration(lv_obj_t *obj, uint32_t t)`

Set the animation time of the spinner

Parametri

- **obj** -- pointer to a spinner
- **t** -- the animation time in milliseconds

`void lv_spinner_set_arc_sweep(lv_obj_t *obj, uint32_t angle)`

Set the animation arc length of the spinner. The animation is suited to values between 180 and 360.

Parametri

- **obj** -- pointer to a spinner
- **angle** -- the angle of the arc in degrees

Variabili

`const lv_obj_class_t lv_spinner_class`

lv_spinner_private.h

`struct _lv_spinner_t`

Membri Pubblici

`lv_arc_t arc`

Add the ancestor's type first

`uint32_t duration`

Anim duration in ms

`uint32_t angle`

Anim angle in degrees

14.27.32 switch

lv_switch.h

Define

LV_SWITCH_KNOB_EXT_AREA_CORRECTION

Switch knob extra area correction factor

Enum

enum **lv_switch_orientation_t**

Values:

enumerator **LV_SWITCH_ORIENTATION_AUTO**

enumerator **LV_SWITCH_ORIENTATION_HORIZONTAL**

enumerator **LV_SWITCH_ORIENTATION_VERTICAL**

Funzioni

lv_obj_t *lv_switch_create(lv_obj_t *parent)

Create a switch object

Parametri

- **parent** -- pointer to an object, it will be the parent of the new switch

Ritorna

pointer to the created switch

void lv_switch_set_orientation(lv_obj_t *obj, lv_switch_orientation_t orientation)

Set the orientation of switch.

Parametri

- **obj** -- pointer to switch object
- **orientation** -- switch orientation from `lv_switch_orientation_t`

lv_switch_orientation_t lv_switch_get_orientation(lv_obj_t *obj)

Get the orientation of switch.

Parametri

obj -- pointer to switch object

Ritorna

switch orientation from `lv_switch_orientation_t`

Variabili

const lv_obj_class_t **lv_switch_class**

lv_switch_private.h

struct **_lv_switch_t**

Membri Pubblici

lv_obj_t obj

```
int32_t anim_state

lv_switch_orientation_t orientation
Orientation of switch
```

14.27.33 table

lv_table.h

Define

LV_TABLE_CELL_NONE

Enum

enum **lv_table_cell_ctrl_t**

Values:

enumerator **LV_TABLE_CELL_CTRL_NONE**

enumerator **LV_TABLE_CELL_CTRL_MERGE_RIGHT**

enumerator **LV_TABLE_CELL_CTRL_TEXT_CROP**

enumerator **LV_TABLE_CELL_CTRL_CUSTOM_1**

enumerator **LV_TABLE_CELL_CTRL_CUSTOM_2**

enumerator **LV_TABLE_CELL_CTRL_CUSTOM_3**

enumerator **LV_TABLE_CELL_CTRL_CUSTOM_4**

Funzioni

lv_obj_t *lv_table_create(lv_obj_t *parent)

Create a table object

Parametri

parent -- pointer to an object, it will be the parent of the new table

Ritorna

pointer to the created table

void lv_table_set_cell_value(lv_obj_t *obj, uint32_t row, uint32_t col, const char *txt)

Set the value of a cell.

Nota

New rows/columns are added automatically if required

Parametri

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row_cnt -1]
- **col** -- id of the column [0 .. col_cnt -1]

- **txt** -- text to display in the cell. It will be copied and saved so this variable is not required after this function call.

`void lv_table_set_cell_value_fmt(lv_obj_t *obj, uint32_t row, uint32_t col, const char *fmt, ...)`

Set the value of a cell. Memory will be allocated to store the text by the table.

Nota

New rows/columns are added automatically if required

Parametri

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row_cnt -1]
- **col** -- id of the column [0 .. col_cnt -1]
- **fmt** -- printf-like format

`void lv_table_set_row_count(lv_obj_t *obj, uint32_t row_cnt)`

Set the number of rows

Parametri

- **obj** -- table pointer to a Table object
- **row_cnt** -- number of rows

`void lv_table_set_column_count(lv_obj_t *obj, uint32_t col_cnt)`

Set the number of columns

Parametri

- **obj** -- table pointer to a Table object
- **col_cnt** -- number of columns.

`void lv_table_set_column_width(lv_obj_t *obj, uint32_t col_id, int32_t w)`

Set the width of a column

Parametri

- **obj** -- table pointer to a Table object
- **col_id** -- id of the column [0 .. LV_TABLE_COL_MAX -1]
- **w** -- width of the column

`void lv_table_set_cell_ctrl(lv_obj_t *obj, uint32_t row, uint32_t col, lv_table_cell_ctrl_t ctrl)`

Add control bits to the cell.

Parametri

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row_cnt -1]
- **col** -- id of the column [0 .. col_cnt -1]
- **ctrl** -- OR-ed values from ::lv_table_cell_ctrl_t

`void lv_table_clear_cell_ctrl(lv_obj_t *obj, uint32_t row, uint32_t col, lv_table_cell_ctrl_t ctrl)`

Clear control bits of the cell.

Parametri

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row_cnt -1]
- **col** -- id of the column [0 .. col_cnt -1]
- **ctrl** -- OR-ed values from ::lv_table_cell_ctrl_t

`void lv_table_set_cell_user_data(lv_obj_t *obj, uint16_t row, uint16_t col, void *user_data)`

Add custom user data to the cell.

Parametri

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row_cnt -1]
- **col** -- id of the column [0 .. col_cnt -1]
- **user_data** -- pointer to the new user_data. Should be allocated by `lv_malloc`, and it will be freed automatically when the table is deleted or when the cell is dropped due to lower row or column count.

```
void lv_table_set_selected_cell(lv_obj_t *obj, uint16_t row, uint16_t col)
```

Set the selected cell

Parametri

- **obj** -- pointer to a table object
- **row** -- id of the cell row to select
- **col** -- id of the cell column to select

```
const char *lv_table_get_cell_value(lv_obj_t *obj, uint32_t row, uint32_t col)
```

Get the value of a cell.

Parametri

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row_cnt -1]
- **col** -- id of the column [0 .. col_cnt -1]

Ritorna

text in the cell

```
uint32_t lv_table_get_row_count(lv_obj_t *obj)
```

Get the number of rows.

Parametri

- **obj** -- table pointer to a Table object

Ritorna

number of rows.

```
uint32_t lv_table_get_column_count(lv_obj_t *obj)
```

Get the number of columns.

Parametri

- **obj** -- table pointer to a Table object

Ritorna

number of columns.

```
int32_t lv_table_get_column_width(lv_obj_t *obj, uint32_t col)
```

Get the width of a column

Parametri

- **obj** -- table pointer to a Table object
- **col** -- id of the column [0 .. LV_TABLE_COL_MAX -1]

Ritorna

width of the column

```
bool lv_table_has_cell_ctrl(lv_obj_t *obj, uint32_t row, uint32_t col, lv_table_cell_ctrl_t ctrl)
```

Get whether a cell has the control bits

Parametri

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row_cnt -1]
- **col** -- id of the column [0 .. col_cnt -1]
- **ctrl** -- OR-ed values from ::lv_table_cell_ctrl_t

Ritorna

true: all control bits are set; false: not all control bits are set

```
void lv_table_get_selected_cell(lv_obj_t *obj, uint32_t *row, uint32_t *col)
```

Get the selected cell (pressed and or focused)

Parametri

- **obj** -- pointer to a table object
- **row** -- pointer to variable to store the selected row (LV_TABLE_CELL_NONE: if no cell selected)
- **col** -- pointer to variable to store the selected column (LV_TABLE_CELL_NONE: if no cell selected)

```
void *lv_table_get_cell_user_data(lv_obj_t *obj, uint16_t row, uint16_t col)
```

Get custom user data to the cell.

Parametri

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row_cnt -1]
- **col** -- id of the column [0 .. col_cnt -1]

Variabili

```
const lv_obj_class_t lv_table_class
```

lv_table_private.h

```
struct _lv_table_cell_t
#include <lv_table_private.h> Cell data
```

Membri Pubblici

lv_table_cell_ctrl_t **ctrl**

void *user_data

Custom user data

char txt[1]

Variable length array

```
struct _lv_table_t
```

```
#include <lv_table_private.h> Table data
```

Membri Pubblici

lv_obj_t **obj**

uint32_t **col_cnt**

uint32_t **row_cnt**

*lv_table_cell_t *****cell_data**

*int32_t ****row_h**

*int32_t ****col_w**

uint32_t **col_act**

uint32_t **row_act**

14.27.34 tabview

lv_tabview.h

Funzioni

`lv_obj_t *lv_tabview_create(lv_obj_t *parent)`

Create a tabview widget

Parametri

- **parent** -- pointer to a parent widget

Ritorna

the created tabview

`lv_obj_t *lv_tabview_add_tab(lv_obj_t *obj, const char *name)`

Add a tab to the tabview

Parametri

- **obj** -- pointer to a tabview widget
- **name** -- the name of the tab, it will be displayed on the tab bar

Ritorna

the widget where the content of the tab can be created

`void lv_tabview_set_tab_text(lv_obj_t *obj, uint32_t idx, const char *new_name)`

Change the name of the tab

Parametri

- **obj** -- pointer to a tabview widget
- **idx** -- the index of the tab to rename
- **new_name** -- the new name as a string

`lv_obj_t *lv_tabview_set_tab_translation_tag(lv_obj_t *obj, const char *tag)`

Add a tab with a translation tag to the tabview.

Parametri

- **obj** -- pointer to a tabview widget
- **tag** -- translation key used for the tab label; will be displayed on the tab bar

Ritorna

the widget where the content of the tab can be created

`void lv_tabview_set_active(lv_obj_t *obj, uint32_t idx, lv_anim_enable_t anim_en)`

Show a tab

Parametri

- **obj** -- pointer to a tabview widget
- **idx** -- the index of the tab to show
- **anim_en** -- LV_ANIM_ON/OFF

`void lv_tabview_set_tab_bar_position(lv_obj_t *obj, lv_dir_t dir)`

Set the position of the tab bar

Parametri

- **obj** -- pointer to a tabview widget
- **dir** -- LV_DIR_TOP/BOTTOM/LEFT/RIGHT

`void lv_tabview_set_tab_bar_size(lv_obj_t *obj, int32_t size)`

Set the width or height of the tab bar

Parametri

- **obj** -- pointer to tabview widget
- **size** -- size of the tab bar in pixels or percentage. will be used as width or height based on the position of the tab bar)

`uint32_t lv_tabview_get_tab_count(lv_obj_t *obj)`

Get the number of tabs

Parametri

- **obj** -- pointer to a tabview widget

Ritorna

the number of tabs

`uint32_t lv_tabview_get_tab_active(lv_obj_t *obj)`

Get the current tab's index

Parametri

- obj** -- pointer to a tabview widget

Ritorna

the zero based index of the current tab

lv_obj_t *lv_tabview_get_tab_button(lv_obj_t *obj, int32_t idx)

Get a given tab button by index

Parametri

- **obj** -- pointer to a tabview widget
- **idx** -- zero based index of the tab button to get. < 0 means start counting tab button from the back (-1 is the last tab button)

Ritorna

pointer to the tab button, or NULL if the index was out of range

lv_obj_t *lv_tabview_get_content(lv_obj_t *obj)

Get the widget where the container of each tab is created

Parametri

- obj** -- pointer to a tabview widget

Ritorna

the main container widget

lv_obj_t *lv_tabview_get_tab_bar(lv_obj_t *obj)

Get the tab bar where the buttons are created

Parametri

- obj** -- pointer to a tabview widget

Ritorna

the tab bar

Variabili

const lv_obj_class_t **lv_tabview_class**

lv_tabview_private.h

struct **_lv_tabview_t**

Membri Pubblici

lv_obj_t **obj**

uint32_t **tab_cur**

lv_dir_t **tab_pos**

14.27.35 textarea**lv_textarea.h****Define**

LV_TEXTAREA_CURSOR_LAST

LV_PART_TEXTAREA_PLACEHOLDER

Enum

enum **lv_property_textarea_id_t**

Values:

enumerator **LV_PROPERTY_TEXTAREA_TEXT**

enumerator **LV_PROPERTY_TEXTAREA_PLACEHOLDER_TEXT**

enumerator **LV_PROPERTY_TEXTAREA_CURSOR_POS**

enumerator **LV_PROPERTY_TEXTAREA_CURSOR_CLICK_POS**

enumerator **LV_PROPERTY_TEXTAREA_PASSWORD_MODE**

enumerator **LV_PROPERTY_TEXTAREA_PASSWORD_BULLET**

enumerator **LV_PROPERTY_TEXTAREA_ONE_LINE**

enumerator **LV_PROPERTY_TEXTAREA_ACCEPTED_CHARS**

enumerator **LV_PROPERTY_TEXTAREA_MAX_LENGTH**

enumerator **LV_PROPERTY_TEXTAREA_INSERT_REPLACE**

enumerator **LV_PROPERTY_TEXTAREA_TEXT_SELECTION**

enumerator **LV_PROPERTY_TEXTAREA_PASSWORD_SHOW_TIME**

enumerator **LV_PROPERTY_TEXTAREA_LABEL**

enumerator **LV_PROPERTY_TEXTAREA_TEXT_IS_SELECTED**

enumerator **LV_PROPERTY_TEXTAREA_CURRENT_CHAR**

enumerator **LV_PROPERTY_TEXTAREA_END**

Funzioni

lv_obj_t *lv_textarea_create(lv_obj_t *parent)

Create a text area object

Parametri

parent -- pointer to an object, it will be the parent of the new text area

Ritorna

pointer to the created text area

void lv_textarea_add_char(lv_obj_t *obj, uint32_t c)

Insert a character to the current cursor position. To add a wide char, e.g. 'Á' use lv_text_encoded_conv_wc('Á')

Parametri

- **obj** -- pointer to a text area object
- **c** -- a character (e.g. 'a')

void lv_textarea_add_text(lv_obj_t *obj, const char *txt)

Insert a text to the current cursor position

Parametri

- **obj** -- pointer to a text area object
- **txt** -- a '\0' terminated string to insert

void lv_textarea_delete_char(lv_obj_t *obj)

Delete a the left character from the current cursor position

Parametri

- **obj** -- pointer to a text area object

void lv_textarea_delete_char_forward(lv_obj_t *obj)

Delete the right character from the current cursor position

Parametri

- **obj** -- pointer to a text area object

void lv_textarea_set_text(lv_obj_t *obj, const char *txt)

Set the text of a text area

Parametri

- **obj** -- pointer to a text area object
- **txt** -- pointer to the text

void lv_textarea_set_placeholder_text(lv_obj_t *obj, const char *txt)

Set the placeholder text of a text area

Parametri

- **obj** -- pointer to a text area object
- **txt** -- pointer to the text

void lv_textarea_set_cursor_pos(lv_obj_t *obj, int32_t pos)

Set the cursor position

Parametri

- **obj** -- pointer to a text area object
- **pos** -- the new cursor position in character index < 0 : index from the end of the text
LV_TEXTAREA_CURSOR_LAST: go after the last character

void lv_textarea_set_cursor_click_pos(lv_obj_t *obj, bool en)

Enable/Disable the positioning of the cursor by clicking the text on the text area.

Parametri

- **obj** -- pointer to a text area object
- **en** -- true: enable click positions; false: disable

void lv_textarea_set_password_mode(lv_obj_t *obj, bool en)

Enable/Disable password mode

Parametri

- **obj** -- pointer to a text area object
- **en** -- true: enable, false: disable

void lv_textarea_set_password_bullet(lv_obj_t *obj, const char *bullet)

Set the replacement characters to show in password mode

Parametri

- **obj** -- pointer to a text area object
- **bullet** -- pointer to the replacement text

```
void lv_textarea_set_one_line(lv_obj_t *obj, bool en)
```

Configure the text area to one line or back to normal

Parametri

- **obj** -- pointer to a text area object
- **en** -- true: one line, false: normal

```
void lv_textarea_set_accepted_chars(lv_obj_t *obj, const char *list)
```

Set a list of characters. Only these characters will be accepted by the text area

Parametri

- **obj** -- pointer to a text area object
- **list** -- list of characters. Only the pointer is saved. E.g. "+-,0123456789"

```
void lv_textarea_set_max_length(lv_obj_t *obj, uint32_t num)
```

Set max length of a Text Area.

Parametri

- **obj** -- pointer to a text area object
- **num** -- the maximal number of characters can be added (`lv_textarea_set_text` ignores it)

```
void lv_textarea_set_insert_replace(lv_obj_t *obj, const char *txt)
```

In `LV_EVENT_INSERT` the text which planned to be inserted can be replaced by another text. It can be used to add automatic formatting to the text area.

Parametri

- **obj** -- pointer to a text area object
- **txt** -- pointer to a new string to insert. If "" no text will be added. The variable must be live after the `event_cb` exists. (Should be `global` or `static`)

```
void lv_textarea_set_text_selection(lv_obj_t *obj, bool en)
```

Enable/disable selection mode.

Parametri

- **obj** -- pointer to a text area object
- **en** -- true or false to enable/disable selection mode

```
void lv_textarea_set_password_show_time(lv_obj_t *obj, uint32_t time)
```

Set how long show the password before changing it to '*'.

Parametri

- **obj** -- pointer to a text area object
- **time** -- show time in milliseconds. 0: hide immediately.

```
void lv_textarea_set_align(lv_obj_t *obj, lv_text_align_t align)
```

Deprecated:

Use the normal `text_align` style property instead Set the label's alignment. It sets where the label is aligned (in one line mode it can be smaller than the text area) and how the lines of the area align in case of multiline text area

Parametri

- **obj** -- pointer to a text area object
- **align** -- the align mode from `lv_text_align_t`

```
const char *lv_textarea_get_text(const lv_obj_t *obj)
```

Get the text of a text area. In password mode it gives the real text (not '*'s).

Parametri

obj -- pointer to a text area object

Ritorna

pointer to the text

```
const char *lv_textarea_get_placeholder_text(lv_obj_t *obj)
```

Get the placeholder text of a text area

Parametri

obj -- pointer to a text area object

Ritorna
pointer to the text

```
lv_obj_t *lv_textarea_get_label(const lv_obj_t *obj)
```

Get the label of a text area

Parametri
obj -- pointer to a text area object

Ritorna
pointer to the label object

```
uint32_t lv_textarea_get_cursor_pos(const lv_obj_t *obj)
```

Get the current cursor position in character index

Parametri
obj -- pointer to a text area object

Ritorna
the cursor position

```
bool lv_textarea_get_cursor_click_pos(lv_obj_t *obj)
```

Get whether the cursor click positioning is enabled or not.

Parametri
obj -- pointer to a text area object

Ritorna
true: enable click positions; false: disable

```
bool lv_textarea_get_password_mode(const lv_obj_t *obj)
```

Get the password mode attribute

Parametri
obj -- pointer to a text area object

Ritorna
true: password mode is enabled, false: disabled

```
const char *lv_textarea_get_password_bullet(lv_obj_t *obj)
```

Get the replacement characters to show in password mode

Parametri
obj -- pointer to a text area object

Ritorna
pointer to the replacement text

```
bool lv_textarea_get_one_line(const lv_obj_t *obj)
```

Get the one line configuration attribute

Parametri
obj -- pointer to a text area object

Ritorna
true: one line configuration is enabled, false: disabled

```
const char *lv_textarea_get_accepted_chars(lv_obj_t *obj)
```

Get a list of accepted characters.

Parametri
obj -- pointer to a text area object

Ritorna
list of accented characters.

```
uint32_t lv_textarea_get_max_length(lv_obj_t *obj)
```

Get max length of a Text Area.

Parametri
obj -- pointer to a text area object

Ritorna
the maximal number of characters to be add

```
bool lv_textarea_text_is_selected(const lv_obj_t *obj)
```

Find whether text is selected or not.

Parametri

obj -- pointer to a text area object

Ritorna

whether text is selected or not

bool lv_textarea_get_text_selection(lv_obj_t *obj)

Find whether selection mode is enabled.

Parametri

obj -- pointer to a text area object

Ritorna

true: selection mode is enabled, false: disabled

uint32_t lv_textarea_get_password_show_time(lv_obj_t *obj)

Set how long show the password before changing it to '*'

Parametri

obj -- pointer to a text area object

Ritorna

show time in milliseconds. 0: hide immediately.

uint32_t lv_textarea_get_current_char(lv_obj_t *obj)

Get a the character from the current cursor position

Parametri

obj -- pointer to a text area object

Ritorna

a the character or 0

void lv_textarea_clear_selection(lv_obj_t *obj)

Clear the selection on the text area.

Parametri

obj -- pointer to a text area object

void lv_textarea_cursor_right(lv_obj_t *obj)

Move the cursor one character right

Parametri

obj -- pointer to a text area object

void lv_textarea_cursor_left(lv_obj_t *obj)

Move the cursor one character left

Parametri

obj -- pointer to a text area object

void lv_textarea_cursor_down(lv_obj_t *obj)

Move the cursor one line down

Parametri

obj -- pointer to a text area object

void lv_textarea_cursor_up(lv_obj_t *obj)

Move the cursor one line up

Parametri

obj -- pointer to a text area object

Variabili

const lv_obj_class_t **lv_textarea_class**

lv_textarea_private.h

```
struct _lv_textarea_t
#include <lv_textarea_private.h> Data of text area
```

Membri Pubblici

lv_obj_t obj

lv_obj_t *label

Label of the text area

char *placeholder_txt

Place holder label. only visible if text is an empty string

char *pwd_tmp

Used to store the original text in password mode

char *pwd_bullet

Replacement characters displayed in password mode

const char *accepted_chars

Only these characters will be accepted. NULL: accept all

uint32_t max_length

The max. number of characters. 0: no limit

uint32_t pwd_show_time

Time to show characters in password mode before change them to '*'

int32_t valid_x

Used when stepping up/down to a shorter line. (Used by the library)

uint32_t pos

The current cursor position (0: before 1st letter; 1: before 2nd letter ...)

lv_area_t area

Cursor area relative to the Text Area

uint32_t txt_byte_pos

Byte index of the letter after (on) the cursor

uint8_t show

Cursor is visible now or not (Handled by the library)

uint8_t click_pos

1: Enable positioning the cursor by clicking the text area

struct _lv_textarea_t cursor

```

uint32_t sel_start
    Temporary values for text selection

uint32_t sel_end

uint8_t text_sel_in_prog
    User is in process of selecting

uint8_t text_sel_en
    Text can be selected on this text area

uint8_t pwd_mode
    Replace characters with '*'

uint8_t one_line
    One line mode (ignore line breaks)

```

14.27.36 tileview

lv_tileview.h

Funzioni

lv_obj_t *lv_tileview_create(lv_obj_t *parent)

Create a tileview object

Parametri

parent -- pointer to an object, it will be the parent of the new tileview

Ritorna

pointer to the created tileview

lv_obj_t *lv_tileview_add_tile(lv_obj_t *tv, uint8_t col_id, uint8_t row_id, lv_dir_t dir)

Add a tile to the tileview

Parametri

- **tv** -- pointer to the tileview object
- **col_id** -- column id of the tile
- **row_id** -- row id of the tile
- **dir** -- direction to move to the next tile

Ritorna

pointer to the added tile object

void lv_tileview_set_tile(lv_obj_t *tv, lv_obj_t *tile_obj, lv_anim_enable_t anim_en)

Set the active tile in the tileview.

Parametri

- **parent** -- pointer to the tileview object
- **tile_obj** -- pointer to the tile object to be set as active
- **anim_en** -- animation enable flag (LV_ANIM_ON or LV_ANIM_OFF)

void lv_tileview_set_tile_by_index(lv_obj_t *tv, uint32_t col_id, uint32_t row_id, lv_anim_enable_t anim_en)

Set the active tile by index in the tileview

Parametri

- **tv** -- pointer to the tileview object
- **col_id** -- column id of the tile to be set as active
- **row_id** -- row id of the tile to be set as active
- **anim_en** -- animation enable flag (LV_ANIM_ON or LV_ANIM_OFF)

```
lv_obj_t *lv_tileview_get_tile_active(lv_obj_t *obj)
```

Get the currently active tile in the tileview

Parametri

- **obj** -- pointer to the tileview object

Ritorna

- pointer to the currently active tile object

Variabili

```
const lv_obj_class_t lv_tileview_class
```

```
const lv_obj_class_t lv_tileview_tile_class
```

lv_tileview_private.h

```
struct _lv_tileview_t
```

Membri Pubblici

```
lv_obj_t obj
```

```
lv_obj_t *tile_act
```

```
struct _lv_tileview_tile_t
```

Membri Pubblici

```
lv_obj_t obj
```

```
lv_dir_t dir
```

14.27.37 win

lv_win.h

Funzioni

```
lv_obj_t *lv_win_create(lv_obj_t *parent)
```

Create a window widget

Parametri

- **parent** -- pointer to a parent widget

Ritorna

- the created window

```
lv_obj_t *lv_win_add_title(lv_obj_t *win, const char *txt)
```

Add a title to the window

Parametri

- **obj** -- pointer to a window widget
- **txt** -- the text of the title

Ritorna

- the widget where the content of the title can be created

`lv_obj_t *lv_win_add_button(lv_obj_t *win, const void *icon, int32_t btn_w)`

Add a button to the window

Parametri

- **obj** -- pointer to a window widget
- **icon** -- an icon to be displayed on the button
- **btn_w** -- width of the button

Ritorna

the widget where the content of the button can be created

`lv_obj_t *lv_win_get_header(lv_obj_t *win)`

Get the header of the window

Parametri

- **win** -- pointer to a window widget

Ritorna

the header of the window

`lv_obj_t *lv_win_get_content(lv_obj_t *win)`

Get the content of the window

Parametri

- **win** -- pointer to a window widget

Ritorna

the content of the window

Variabili

const lv_obj_class_t **lv_win_class**

lv_win_private.h

struct **_lv_win_t**

Membri Pubblici

`lv_obj_t obj`

14.28 xml

14.28.1 lv_xml.h

Define

`LV_XML_MAX_PATH_LENGTH`

Funzioni

`void lv_xml_init(void)`

`void lv_xml_deinit(void)`

`void *lv_xml_create(lv_obj_t *parent, const char *name, const char **attrs)`

`lv_obj_t *lv_xml_create_screen(const char *name)`

Create a Screen from XML.

Nota

If required, can be loaded with `lv_screen_load()`.

Parametri

- name** -- The name of an already-registered Screen

Ritorna

Pointer to the created Screen

```
void *lv_xml_create_in_scope(lv_obj_t *parent, lv_xml_component_scope_t *parent_ctx,
                             lv_xml_component_scope_t *scope, const char **attrs)
```

```
void lv_xml_set_default_asset_path(const char *path_prefix)
```

Set a path to prefix the image and font file source paths.

In globals.xml usually the source path is like "images/logo.png". But on the actual device it can be located at e.g. "A:ui/assets/images/logo.png". By setting "A:ui/assets/" the path set in the XML files will be prefixed accordingly.

Parametri

- path_prefix** -- the path to be used as prefix

```
lv_result_t lv_xml_register_font(lv_xml_component_scope_t *scope, const char *name, const lv_font_t *font)
```

```
const lv_font_t *lv_xml_get_font(lv_xml_component_scope_t *scope, const char *name)
```

```
lv_result_t lv_xml_register_image(lv_xml_component_scope_t *scope, const char *name, const void *src)
```

```
const void *lv_xml_get_image(lv_xml_component_scope_t *scope, const char *name)
```

```
lv_result_t lv_xml_register_subject(lv_xml_component_scope_t *scope, const char *name, lv_subject_t
                                    *subject)
```

Map globally available subject name to an actual subject variable

Parametri

- name** -- name of the subject
- subject** -- pointer to a subject

Ritorna

LV_RESULT_OK: success

```
lv_subject_t *lv_xml_get_subject(lv_xml_component_scope_t *scope, const char *name)
```

Get a subject by name.

Parametri

- scope** -- If specified start searching in that component's subject list, and if not found search in the global space. If NULL search in global space immediately.
- name** -- Name of the subject to find.

Ritorna

Pointer to the subject or NULL if not found.

```
lv_result_t lv_xml_register_const(lv_xml_component_scope_t *scope, const char *name, const char *value)
```

```
const char *lv_xml_get_const(lv_xml_component_scope_t *scope, const char *name)
```

```
lv_result_t lv_xml_register_event_cb(lv_xml_component_scope_t *scope, const char *name, lv_event_cb_t cb)
```

```
lv_event_cb_t lv_xml_get_event_cb(lv_xml_component_scope_t *scope, const char *name)
```

```
lv_result_t lv_xml_register_timeline(lv_xml_component_scope_t *scope, const char *name)
```

```
void *lv_xml_get_timeline(lv_xml_component_scope_t *scope, const char *name)
```

14.28.2 lv_xml_base_types.h

Funzioni

lv_state_t **lv_xml_state_to_enum**(const char *txt)

Convert an state string to enum

Parametri

txt -- e.g. "pressed"

Ritorna

the related enum, e.g. LV_STATE_PRESSED

int32_t **lv_xml_to_size**(const char *txt)

Process inputs "content", "32", "32px", or "25%" and convert them to integer

Parametri

txt -- the input string

Ritorna

the integer size

lv_align_t **lv_xml_align_to_enum**(const char *txt)

Convert an align string to enum

Parametri

txt -- e.g. "center"

Ritorna

the related enum, e.g. LV_ALIGN_CENTER

lv_dir_t **lv_xml_dir_to_enum**(const char *txt)

Convert a direction string to enum

Parametri

txt -- e.g. "top"

Ritorna

the related enum, e.g. LV_DIR_TOP

lv_border_side_t **lv_xml_border_side_to_enum**(const char *txt)

Convert a direction string to enum

Parametri

txt -- e.g. "top"

Ritorna

the related enum, e.g. LV_BORDER_SIDE_TOP

lv_base_dir_t **lv_xml_base_dir_to_enum**(const char *txt)

Convert a base dir string to enum

Parametri

txt -- e.g. "rtl"

Ritorna

the related enum, e.g. LV_BASE_DIRRTL

lv_grad_dir_t **lv_xml_grad_dir_to_enum**(const char *txt)

Convert a grad dir string to enum

Parametri

txt -- e.g. "hor"

Ritorna

the related enum, e.g. LV_GRAD_DIR_HOR

lv_text_align_t **lv_xml_text_align_to_enum**(const char *txt)

Convert a text align string to enum

Parametri

txt -- e.g. "left"

Ritorna

the related enum, e.g. LV_TEXT_ALIGN_LEFT

lv_text_decor_t **lv_xml_text_decor_to_enum**(const char *txt)

Convert a text decor string to enum

Parametri

txt -- e.g. "underline"

Ritorna

the related enum, e.g. LV_TEXT_DECOR_UNDERLINE

lv_scroll_snap_t **lv_xml_scroll_snap_to_enum**(const char *txt)

Convert a scroll snap string to enum

Parametri

txt -- e.g. "start"

Ritorna

the related enum, e.g. LV_SCROLL_SNAP_START

lv_scrollbar_mode_t **lv_xml_scrollbar_mode_to_enum**(const char *txt)

Convert a scrollbar mode string to enum

Parametri

txt -- e.g. "active"

Ritorna

the related enum, e.g. LV_SCROLLBAR_MODE_ACTIVE

lv_flex_flow_t **lv_xml_flex_flow_to_enum**(const char *txt)

Convert a flex flow string to enum

Parametri

txt -- e.g. "row_wrap"

Ritorna

the related enum, e.g. LV_FLEX_FLOW_ROW_WRAP

lv_flex_align_t **lv_xml_flex_align_to_enum**(const char *txt)

Convert a flex align string to enum

Parametri

txt -- e.g. "space_between"

Ritorna

the related enum, e.g. LV_FLEX_ALIGN_SPACE_BETWEEN

lv_grid_align_t **lv_xml_grid_align_to_enum**(const char *txt)

Convert a grid align string to enum

Parametri

txt -- e.g. "space_between"

Ritorna

the related enum, e.g. LV_GRID_ALIGN_SPACE_BETWEEN

lv_layout_t **lv_xml_layout_to_enum**(const char *txt)

Convert a layout string to enum

Parametri

txt -- e.g. "flex"

Ritorna

the related enum, e.g. LV_LAYOUT_FLEX

lv_blend_mode_t **lv_xml_blend_mode_to_enum**(const char *txt)

Convert a blend mode string to enum

Parametri

txt -- e.g. "additive"

Ritorna

the related enum, e.g. LV_BLEND_MODE_ADDITIVE

lv_event_code_t **lv_xml_trigger_text_to_enum_value**(const char *txt)

Convert an event trigger string to enum

Parametri

txt -- e.g. "clicked"

Ritorna

the related enum, e.g. LV_EVENT_CLICKED

lv_screen_load_anim_t **lv_xml_screen_load_anim_text_to_enum_value**(const char *txt)

Convert a screen animation type string to enum

Parametri

txt -- e.g. "over_right"

Ritorna

the related enum, e.g. LV_SCREEN_LOAD_ANIM_OVER_RIGHT

lv_style_prop_t **lv_xml_style_prop_to_enum**(const char *txt)

Convert a style property string to enum

Parametri

txt -- e.g. "bg_color"

Ritorna

the related enum, e.g. LV_STYLE_BG_COLOR or LV_STYLE_PROP_INV if not found.

lv_state_t **lv_xml_style_state_to_enum**(const char *txt)

Convert a style state to enum

Parametri

txt -- e.g. "pressed"

Ritorna

the enum LV_STATE_PRESSED

lv_part_t **lv_xml_style_part_to_enum**(const char *txt)

Convert a style part to enum

Parametri

txt -- e.g. "knob"

Ritorna

the enum LV_PART_KNOB

lv_style_selector_t **lv_xml_style_selector_text_to_enum**(const char *str)

Convert ORed style parts and states to an ORed selector

Parametri

txt -- e.g. "knob|pressed"

Ritorna

the enum LV_PART_KNOB|LV_STATE_PRESSED

14.28.3 lv_xml_component.h

Funzioni

lv_obj_t ***lv_xml_component_process**(lv_xml_parser_state_t *state, const char *name, const char **attrs)

Process a component during parsing an XML. It create a widget and apply all the attributes

Parametri

- **state** -- the current parsing state
- **name** -- name of the component
- **attrs** -- attributes of the widget

Ritorna

lv_result_t **lv_xml_register_component_from_data**(const char *name, const char *xml_def)

Load the styles, constants, and other data of the Component. It needs to be called only once for each Component.

Parametri

- **name** -- The name as the component will be referenced later in other components
- **xml_def** -- The XML definition of the component as a NULL terminated string

Ritorna

LV_RESULT_OK: loaded successfully, LV_RES_INVALID: otherwise

`lv_result_t lv_xml_register_component_from_file(const char *path)`

Load the styles, constants, and other data of the Component. It needs to be called only once for each Component.

Parametri

`path` -- Path to an XML file

Ritorna

LV_RESULT_OK: loaded successfully, LV_RES_INVALID: otherwise

`lv_xml_component_scope_t *lv_xml_component_get_scope(const char *component_name)`

Get the scope of a Component which was registered by `lv_xml_register_component_from_data()` or `lv_xml_register_component_from_file()`

Parametri

`component_name` -- Name of the Component

Ritorna

Pointer to the scope or NULL if not found

`lv_result_t lv_xml_component_unregister(const char *name)`

Remove a component from from the list.

Parametri

`name` -- the name of the component (used during registration)

Ritorna

LV_RESULT_OK on successful unregistration, LV_RESULT_INVALID otherwise.

14.28.4 lv_xml_component_private.h

Typedef

`typedef void *(*lv_xml_component_process_cb_t)(lv_obj_t *parent, const char *data, const char **attr)`

Funzioni

`void lv_xml_component_init(void)`

Initialize the components system.

`void lv_xml_component_scope_init(lv_xml_component_scope_t *scope)`

Initialize the linked lists of a component context

Parametri

`scope` -- pointer to a component contexts

`struct _lv_xml_component_scope_t`

Membri Pubblici

`const char *name`

`lv_ll_t style_ll`

`lv_ll_t const_ll`

`lv_ll_t param_ll`

`lv_ll_t gradient_ll`

`lv_ll_t subjects_ll`

```
lv_ll_t timeline_ll  
  
lv_ll_t font_ll  
  
lv_ll_t image_ll  
  
lv_ll_t event_ll  
  
const char *view_def  
  
const char *extends  
  
uint32_t is_widget  
  
uint32_t is_screen  
  
struct lv_xml_component_scope_t *next  
  
struct lv_xml_const_t
```

Membri Pubblici

```
const char *name  
  
const char *value  
  
struct lv_xml_subject_t
```

Membri Pubblici

```
const char *name  
  
lv_subject_t *subject  
  
struct lv_xml_timeline_t
```

Membri Pubblici

```
const char *name  
  
lv_ll_t anim_ll  
  
struct lv_xml_param_t
```

Membri Pubblici

```
const char *name
```

```
const char *def
```

```
const char *type
```

```
struct lv_xml_grad_t
```

Membri Pubblici

```
const char *name
```

```
lv_grad_dsc_t grad_dsc
```

14.28.5 lv_xml_load.h**Funzioni**

`lv_result_t lv_xml_load_all_from_path(const char *path)`

Recurse into a directory, loading all XML components, screens, globals, and translations.

Parametri

- `path` -- the path to a directory to load files from

Ritorna

- `LV_RESULT_OK` if there were no issues or `LV_RESULT_INVALID` otherwise.

`lv_xml_load_t *lv_xml_load_all_from_data(const void *buf, uint32_t buf_size)`

Mount a data blob and recurse through it, loading all XML components, screens, globals, and translations.

Parametri

- `buf` -- the data blob
- `buf_size` -- the size of the data blob

Ritorna

- a handle that can be used to unload it later

`lv_xml_load_t *lv_xml_load_all_from_file(const char *file_path)`

Mount a data blob located at `file_path` in a filesystem and recurse through it, loading all XML components, screens, globals, and translations.

Parametri

- `file_path` -- the path of the data blob

Ritorna

- a handle that can be used to unload it later

`void lv_xml_unload(lv_xml_load_t *load)`

Unload XML data that was loaded by a function that returned `lv_xml_load_t *`. Any assets in the loaded data will not be accessible anymore.

Parametri

- `load` -- a loaded XML data handle, or `NULL` to unload all.

14.28.6 lv_xml_load_private.h**Funzioni**

`void lv_xml_load_init(void)`

`void lv_xml_load_deinit(void)`

14.28.7 lv_xml_parser.h

Enum

enum **lv_xml_parser_section_t**

Values:

enumerator **LV_PARSER_SECTION_NONE**

enumerator **LV_PARSER_SECTION_API**

enumerator **LV_PARSER_SECTION_CONSTS**

enumerator **LV_PARSER_SECTION_GRAD**

enumerator **LV_PARSER_SECTION_GRAD_STOP**

enumerator **LV_PARSER_SECTION_STYLES**

enumerator **LV_PARSER_SECTION_FONTS**

enumerator **LV_PARSER_SECTION_IMAGES**

enumerator **LV_PARSER_SECTION_SUBJECTS**

enumerator **LV_PARSER_SECTION_ANIMATION**

enumerator **LV_PARSER_SECTION_INCLUDE_TIMELINE**

enumerator **LV_PARSER_SECTION_TIMELINE**

enumerator **LV_PARSER_SECTION_VIEW**

Funzioni

void **lv_xml_parser_state_init**(lv_xml_parser_state_t *state)

void **lv_xml_parser_start_section**(lv_xml_parser_state_t *state, const char *name)

void **lv_xml_parser_end_section**(lv_xml_parser_state_t *state, const char *name)

void ***lv_xml_state_get_parent**(lv_xml_parser_state_t *state)

void ***lv_xml_state_get_item**(lv_xml_parser_state_t *state)

struct **_lv_xml_parser_state_t**

Membri Pubblici

```
const char *tag_name
lv_xml_component_scope_t scope
lv_ll_t parent_ll
lv_obj_t *parent
lv_obj_t *item
lv_obj_t *view
void *context
const char **parentAttrs
lv_xml_component_scope_t *parentScope
lv_xml_parser_section_t section
```

14.28.8 lv_xml_private.h

```
struct lv_xml_font_t
```

Membri Pubblici

```
const char *name
const lv_font_t *font
void (*font_destroy_cb)(lv_font_t*)
```

```
struct lv_xml_image_t
```

Membri Pubblici

```
const char *name
```

```
const void *src
```

```
struct lv_xml_event_cb_t
```

Membri Pubblici

```
const char *name
```

```
lv_event_cb_t cb
```

```
struct lv_xml_anim_timeline_include_t
```

```
#include <lv_xml_private.h> Store the data of <include_timeline>
```

Membri Pubblici

```
const char *target_name
```

Include the timeline of this widget

```
const char *timeline_name
```

Include this timeline

```
int32_t delay
```

```
struct lv_xml_anim_timeline_child_t
```

Membri Pubblici

```
bool is_anim
```

```
lv_anim_t anim
```

```
lv_xml_anim_timeline_include_t incl
```

```
union lv_xml_anim_timeline_child_t data
```

14.28.9 lv_xml_style.h**Typedef**

```
typedef struct lv_xml_style_t lv_xml_style_t
```

Funzioni

```
lv_result_t lv_xml_register_style(lv_xml_component_scope_t *scope, const char **attrs)
```

Add a style to ctx and set the style properties from attrs

Parametri

- **scope** -- add styles here. (Constants should be already added as style properties might use them)
- **attrs** -- list of attribute names and values

```
const char *lv_xml_style_string_process(char *txt, lv_style_selector_t *selector)
```

Decompose a string like "style1:pressed:checked:knob" to style name and selector

Parametri

- **txt** -- the input string
- **selector** -- store the selectors here

Ritorna

the style name or NULL on any error

```
lv_xml_style_t *lv_xml_get_style_by_name(lv_xml_component_scope_t *scope, const char *name)
```

Find a style by name which was added by `lv_xml_register_style`

Parametri

- **scope** -- the default context to search in
- **name** -- the name of the style. Can start with a component name prefix (e.g. `my_button.blue`) to overwrite the ctx

Ritorna

the style structure

```
lv_grad_dsc_t *lv_xml_component_get_grad(lv_xml_component_scope_t *scope, const char *name)
```

Get a gradient descriptor defined for a component

Parametri

- **scope** -- component context where the gradient should be found
- **name** -- name of the gradient

Ritorna

a gradient descriptor

```
struct _lv_xml_style_t
```

Membri Pubblici

```
const char *name
```

```
const char *long_name
```

```
lv_style_t style
```

14.28.10 lv_xml_test.h**Enum**

```
enum lv_xml_test_step_type_t
```

Values:

```
enumerator LV_XML_TEST_STEP_TYPE_NONE
```

```
enumerator LV_XML_TEST_STEP_TYPE_MOVE_TO
```

```
enumerator LV_XML_TEST_STEP_TYPE_PRESS
```

```
enumerator LV_XML_TEST_STEP_TYPE_RELEASE
```

```
enumerator LV_XML_TEST_STEP_TYPE_CLICK_AT
```

```
enumerator LV_XML_TEST_STEP_TYPE_CLICK_ON
```

```
enumerator LV_XML_TEST_STEP_TYPE_WAIT
```

enumerator **LV_XML_TEST_STEP_TYPE_FREEZE**

enumerator **LV_XML_TEST_STEP_TYPE_SCREENSHOT_COMPARE**

enumerator **LV_XML_TEST_STEP_TYPE SUBJECT_SET**

enumerator **LV_XML_TEST_STEP_TYPE SUBJECT_COMPARE**

enumerator **LV_XML_TEST_STEP_TYPE_SET_LANGUAGE**

Funzioni

`lv_result_t lv_xml_test_register_from_data(const char *xml_def, const char *ref_image_path_prefix)`

Load the styles, constants, another data of the test. It needs to be called only once for each test.

Parametri

- **xml_def** -- the XML definition of the test as a NULL terminated string
- **ref_image_path_prefix** -- prefix for the path of reference images

Ritorna

LV_RESULT_OK: loaded successfully, LV_RES_INVALID: otherwise

`lv_result_t lv_xml_test_register_from_file(const char *path, const char *ref_image_path_prefix)`

Load the styles, constants, another data of the test. It needs to be called only once for each test.

Parametri

- **path** -- path to an XML file
- **ref_image_path_prefix** -- prefix for the path of reference images

Ritorna

LV_RESULT_OK: loaded successfully, LV_RES_INVALID: otherwise

`void lv_xml_test_unregister(void)`

Free resources allocated for testing.

`void lv_xml_test_run_init(void)`

Switch to testing mode. Needs to be called to use `lv_xml_test_run_next()`

`bool lv_xml_test_run_next(uint32_t slowdown)`

Run the next test step.

Parametri

slowdown -- 0: max speed, 1: real speed, 2: half speed, ... ,10: ten times slower

Ritorna

true: the step passed; false: the step failed

`void lv_xml_test_run_stop(void)`

Leave testing mode.

`uint32_t lv_xml_test_run_all(uint32_t slowdown)`

Run all the test steps. It calls `lv_xml_test_run_init()`, `lv_xml_test_run_next()`, and `lv_xml_test_run_stop` internally so no further preparation or cleanup is needed.

Parametri

slowdown -- 0: max speed, 1: real speed, 2: half speed, ... ,10: ten times slower

Ritorna

number of failed tests

`uint32_t lv_xml_test_get_step_count(void)`

Get the number of steps in a test

Ritorna

the number of <step>s

`lv_xml_test_step_type_t lv_xml_test_get_step_type(uint32_t idx)`

Get the type of a step

Parametri

`idx` -- the index of a step (< step_count)

Ritorna

element of `lv_xml_test_step_type_t`

`bool lv_xml_test_get_status(uint32_t idx)`

Check if the a step was passed. Can be called after `lv_xml_test_run()`

Parametri

`idx` -- the index of a step (< step_count)

Ritorna

true: the step passed, false: the step failed

14.28.11 lv_xml_translation.h

Funzioni

`lv_result_t lv_xml_register_translation_from_file(const char *path)`

Register translations from an XML file

Parametri

`path` -- path to an XML file (staring with a driver letter)

Ritorna

`LV_RESULT_OK`: no error

`lv_result_t lv_xml_register_translation_from_data(const char *xml_def)`

Register translations from an XML string

Parametri

`xml_def` -- the XML definition as a string

Ritorna

`LV_RESULT_OK`: no error

14.28.12 lv_xml_update.h

Funzioni

`lv_result_t lv_xml_update_from_data(const char *xml_def)`

Change the properties of a given widget by processing XML snippets. For example `<lv_slider name="my_slider_1" min_value="30" style_bg_color="0xff0000"/>` Note that the tag should be the underlying widget's name and the component's name.

Parametri

`xml_def` -- the XML to process as a string

Ritorna

`LV_RESULT_OK`: loaded successfully, `LV_RES_INVALID`: otherwise

14.28.13 lv_xml_utils.h

Funzioni

`const char *lv_xml_get_value_of(const char **attrs, const char *name)`

`int32_t lv_xml_atoi(const char *str)`

`int32_t lv_xml_atoi_split(const char **str, char delimiter)`

Convert sections of a string to int. The end of the string is indicated by the delimiter.

Parametri

- `str` -- pointer to a string, it will point to the character after the delimiter
- `delimiter` -- a character to indicate the end of the int

Ritorna

the int before the next delimiter

```
float lv_xml_atof(const char *str)
```

```
float lv_xml_atof_split(const char **str, char delimiter)
```

Convert sections of a string to float. The end of the string is indicated by the `delimiter`.

Parametri

- `str` -- pointer to a string, it will point to the character after the delimiter
- `delimiter` -- a character to indicate the end of the float

Ritorna

the float before the next delimiter

```
lv_color_t lv_xml_to_color(const char *str)
```

```
lv_opa_t lv_xml_to_opa(const char *str)
```

Concert percentage or integer opacity value from string to integer.

Parametri

- `str` -- e.g. "70%" or 180

Ritorna

0..255

```
bool lv_xml_to_bool(const char *str)
```

```
int32_t lv_xml_strtol(const char *str, char **endptr, int32_t base)
```

```
char *lv_xml_split_str(char **src, char delimiter)
```

Find a delimiter in a string, terminate the string on the delimiter and update the source string to the next part

Parametri

- `src` -- point to a variable containing the input string
- `delimiter` -- a delimiter character, e.g. ':'

Ritorna

the beginning of next section in the string closed at the delimiter

14.28.14 lv_xml_widget.h

Typedef

```
typedef void *(*lv_xml_widget_create_cb_t)(lv_xml_parser_state_t *state, const char **parentAttrs)
```

```
typedef void (*lv_xml_widget_apply_cb_t)(lv_xml_parser_state_t *state, const char **parentAttrs)
```

```
typedef struct _lv_widget_processor_t lv_widget_processor_t
```

Funzioni

```
lv_result_t lv_xml_register_widget(const char *name, lv_xml_widget_create_cb_t create_cb,
                                    lv_xml_widget_apply_cb_t apply_cb)
```

Register a Widget for the XML parser. When a Widget with a given name is created in XML `create_cb` will be called to create an instance, and then `apply_cb` will be called to apply the properties (e.g. `width=100"`).

Nota

E.g. Chart series, are also considered Widgets although they don't have `lv_obj_t *` type.

Parametri

- `name` -- Name of the Widget (e.g. "my_slider", referenced as <my_slider> in XML)
- `create_cb` -- Called to create an instance of the Widget
- `apply_cb` -- Called to apply its properties

Ritorna

Pointer to the created Widget

`lv_widget_processor_t *lv_xml_widget_get_processor(const char *name)`

Get a descriptor that was created when the Widget was registered.

Parametri

`name` -- The name that was used when the Widget was registered

Ritorna

The descriptor of the Widget

`lv_widget_processor_t *lv_xml_widget_get_extended_widget_processor(const char *extends)`

Get the descriptor of the Widget that is extended by a given Widget, Component, or Screen. E.g. in a Component <view extends="lv_slider"> return the descriptor of lv_slider.

Nota

If a component extends an other component which based on `lv_slider` lv_slider's descriptor will be returned.

Parametri

`extends` -- The name of a Component, Screen, or Widget whose ancestor Widget shall be returned

Ritorna

The descriptor of the extended Widget

struct _lv_widget_processor_t

Membri Pubblici

const char ***name**

`lv_xml_widget_create_cb_t create_cb`

`lv_xml_widget_apply_cb_t apply_cb`

struct _lv_widget_processor_t ***next**

14.28.15 parsers

lv_xml_arc_parser.h

Funzioni

`void *lv_xml_arc_create(lv_xml_parser_state_t *state, const char **attrs)`

`void lv_xml_arc_apply(lv_xml_parser_state_t *state, const char **attrs)`

lv_xml_bar_parser.h

Funzioni

`void *lv_xml_bar_create(lv_xml_parser_state_t *state, const char **attrs)`

`void lv_xml_bar_apply(lv_xml_parser_state_t *state, const char **attrs)`

lv_xml_button_parser.h**Funzioni**

```
void *lv_xml_button_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_button_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_buttonmatrix_parser.h**Funzioni**

```
void *lv_xml_buttonmatrix_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_buttonmatrix_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_calendar_parser.h**Funzioni**

```
void *lv_xml_calendar_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_calendar_apply(lv_xml_parser_state_t *state, const char **attrs)
```

```
void *lv_xml_calendar_header_arrow_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_calendar_header_arrow_apply(lv_xml_parser_state_t *state, const char **attrs)
```

```
void *lv_xml_calendar_header_dropdown_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_calendar_header_dropdown_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_canvas_parser.h**Funzioni**

```
void *lv_xml_canvas_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_canvas_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_chart_parser.h**Funzioni**

```
void *lv_xml_chart_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_chart_apply(lv_xml_parser_state_t *state, const char **attrs)
```

```
void *lv_xml_chart_series_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_chart_series_apply(lv_xml_parser_state_t *state, const char **attrs)
```

```
void *lv_xml_chart_cursor_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_chart_cursor_apply(lv_xml_parser_state_t *state, const char **attrs)
```

```
void *lv_xml_chart_axis_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_chart_axis_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_checkbox_parser.h

Funzioni

```
void *lv_xml_checkbox_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_checkbox_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_dropdown_parser.h

Funzioni

```
void *lv_xml_dropdown_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_dropdown_apply(lv_xml_parser_state_t *state, const char **attrs)
```

```
void *lv_xml_dropdown_list_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_dropdown_list_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_image_parser.h

Funzioni

```
void *lv_xml_image_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_image_apply(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_check_file(const char *filepath)
```

lv_xml_keyboard_parser.h

Funzioni

```
void *lv_xml_keyboard_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_keyboard_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_label_parser.h

Funzioni

```
void *lv_xml_label_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_label_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_obj_parser.h

Funzioni

```
void *lv_xml_obj_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_obj_apply(lv_xml_parser_state_t *state, const char **attrs)
```

```
void *lv_obj_xml_style_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_obj_xml_style_apply(lv_xml_parser_state_t *state, const char **attrs)
```

```
void *lv_obj_xml_remove_style_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_obj_xml_remove_style_apply(lv_xml_parser_state_t *state, const char **attrs)
```

```
void *lv_obj_xml_remove_style_all_create(lv_xml_parser_state_t *state, const char **attrs)
```

```

void lv_obj_xml_remove_style_all_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_event_cb_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_event_cb_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_subject_toggle_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_subject_toggle_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_subject_set_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_subject_set_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_subject_increment_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_subject_increment_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_bind_style_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_bind_style_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_bind_flag_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_bind_flag_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_bind_state_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_bind_state_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_screen_load_event_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_screen_load_event_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_screen_create_event_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_screen_create_event_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_play_timeline_event_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_play_timeline_event_apply(lv_xml_parser_state_t *state, const char **attrs)

```

lv_xml_qrcode_parser.h

Funzioni

```
void *lv_xml_qrcode_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_qrcode_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_roller_parser.h

Funzioni

```
void *lv_xml_roller_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_roller_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_scale_parser.h

Funzioni

```
void *lv_xml_scale_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_scale_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_xml_scale_section_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_scale_section_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_slider_parser.h

Funzioni

```
void *lv_xml_slider_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_slider_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_spangroup_parser.h

Funzioni

```
void *lv_xml_spangroup_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_spangroup_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_xml_spangroup_span_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_spangroup_span_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_spinbox_parser.h

Funzioni

```
void *lv_xml_spinbox_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_spinbox_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_spinner_parser.h

Funzioni

```
void *lv_xml_spinner_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_spinner_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_switch_parser.h

Funzioni

```
void *lv_xml_switch_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_switch_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_table_parser.h

Funzioni

```
void *lv_xml_table_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_table_apply(lv_xml_parser_state_t *state, const char **attrs)
```

```
void *lv_xml_table_column_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_table_column_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_xml_table_cell_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_table_cell_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_tabview_parser.h

Funzioni

```
void *lv_xml_tabview_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_tabview_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_xml_tabview_tab_bar_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_tabview_tab_bar_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_xml_tabview_tab_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_tabview_tab_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_xml_tabview_tab_button_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_tabview_tab_button_apply(lv_xml_parser_state_t *state, const char **attrs)
```

lv_xml_textarea_parser.h

Funzioni

```
void *lv_xml_textarea_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_textarea_apply(lv_xml_parser_state_t *state, const char **attrs)
```