Readings on LVGL (Light and Versatile Graphics Library)

A free and open-source graphics library providing everything you need to create an embedded GUI with easy-to-use graphical elements, beautiful visual effects, and a low memory footprint.

Introduction

- Key features:
    - Building blocks such as buttons, charts, etc
    - Advanced graphics w/ animations
    - Multi-language support
    - Hardware independent
    - Scalable: able to operate w/ little memory
    - Written in C for mac compatibility (C++ compatible)
    - tutorials, examples, themes included

- Requirements:
    - Mentions RAM
    - Basic C(C++) knowledge
        - Structs
        - Pointers
        - callbacks

- All repositories of LVGL projects are hosted on github: https://github.com/lvgl
    - Will find these repositories there:
        - lvgl: The library itself with many examples and demos.
        - lv_drivers: Display and input device drivers
        - blog: Source of the blog's site
        - sim: Source of the online simulator's site
        - lv_port_*: LVGL ports to development boards or environments
        - lv_binding_*: Bindings to other languages
- Includes FAQ

LVGL Basics

- Major Concepts:

    Display vs Screen:
        - Display or display panel is the physical hardware displaying pixels
        - Display object (lv_display) object is an object in RAM that represents a display meant to be used by LVGL
        - Screen is the “root” widget in widget trees and is attached to a particular display
    - Default display: first lv_display object created
    - Screen widget: any widget created without a parent - the root of each widget tree

- Widgets:
    - After LVGL is initialized(Connecting LVGL to Your Hardware), to create interactive user interface, an app next creates tree of widgets used to display interface
    - Widgets are “intelligent” LVGL elements (labels, switches, sliders, etc)
    - To build widget tree, app needs pointer to a screen widget
    - To create new screen widget, create widget passing NULL as the parent argument (base widget)
        - Ex: any widget can contain other widgets such as a button widget having a label widget as a child
    - Cannot delete the active screen widget

- Creating Widgets:
    - Create a pointer

- Modifying widgets
    - Using different functions that can edit widget

- Deleting widgets
    - lv_obj_delete(lv_obj_t * widget)

- Events
    - Used to inform the app that something has happened with a widget
    - Mentions callbacks here
    - The event codes can be grouped into these categories: - Input device events - Drawing events - Other events - Special events - Custom events
- Layouts:
    - Flex: It can arrange items (child Widgets) into rows or columns (tracks), handle wrapping, adjust the spacing between items and tracks, handle *grow* to make item(s) fill remaining space with respect to min/max width and height.
    - Note that the Flex layout feature of LVGL needs to be globally enabled with **LV_USE_FLEX** in lv_conf.h.
    - Grid: It can arrange items (child Widgets) into a 2D "table" that has rows and columns (tracks). An item can span multiple columns or rows. The track's size can be set in pixels, to the largest item (**LV_GRID_CONTENT**), or to a fraction of the available free space (i.e. Free [FR] Units) to distribute free space proportionally.
    - To make a Widget a Grid container call `lv_obj_set_layout(widget, LV_LAYOUT_GRID)`.
- Scrolling:

    - In LVGL scrolling works very intuitively: if a Widget is outside its parent content area (the size without padding), the parent becomes scrollable and scrollbar(s) will appear. That's it.
    - Any widget can be scrollable, widget cna either be scrolled horizontally in one stroke while diagonal scrolling is not possible

Annotate the Hello World C code in the "Basic Examples" Section

Driver Doc

- ST7796 LCD Controller driver: single chip controller/driver for LCD
- Capable of connecting directly to an external microprocessor, accepts 8,9,16, and 18 bit parallel interface
    - Main difference between the serial and parallel interfaces is how they transmit data.
    - In serial interface, the data is sent or received one bit at a time over a series of clock pulses.
    - In parallel mode the interface sends and receives 4 bits, 8 bits, or 16 bits of data at a time over multiple transmission lines.
- Accepts Serial Peripheral Interface
    - (SPI) de facto standard (with many variants) for synchronous serial communication, used primarily in embedded systems for short-distance wired communication between integrated circuits
- Provides MIPI
    - "Mobile Industry Processor Interface," is a standardized interface standard used to connect different components within a mobile device, like cameras and displays, primarily designed for high-speed serial data transmission with low power consumption, often found in smartphones, tablets, and other embedded systems
- Mentions RAM (no external clock to minimize power consumption)
- The ST7796 LCD controller driver implements display initialization, supports display rotation and implements the display flush callback (means to clear it?)
- Configuring the Driver
    - Has code here to create an LCD display with ST7796 driver

Base Widgets (skim over)

- Widget basics:
    - Basic building block of LVGL user interface
    - All widgets referenced using lv_obj_t pointer as a handle
    - Think of base widget as widget class from which all other widgets inherit
    - Can set/get attributes, size and style, with lv_obj_set and lv_obj_get

```
/* 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 */
```

        - Widget types also have special features

```
/* 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) */
```

- Working mechanisms:
    - Mentions creating and deleting widgets as well as moving parent and child together on screen
    - Mentions deleting a widget and creating

- Screens
    - don’t confuse with a Display(lv_display)
    - Are any widget created without a parent (they form the root for the widget tree) and normally the base widget is used for this purpose since it has all the features most screens used
    - An image widget(lv_image) can also be used to create a wallpaper background for the widget tree
    - All screens:
        - Are automatically attached to the default display current when the screen was created (?)
        - Occupy full area of associated object
        - Cannot be moved: functions such as lv_obj_set_pos() and lv_obj_set_size() **cannot** be used on screens
        - Each display (lv_display) object can have multiple screens associated with it but not vice versa
        -

```
   Display
      |
     --- (one or more)
     /|\
Screen Widgets  (root of a Widget Tree)
      |
      O  (zero or more)
     /|\
Child Widgets
```

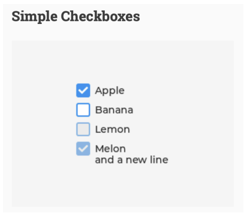
    - Creating screens:
        -

```
lv_obj_t * scr1 = lv_obj_create(NULL);
```

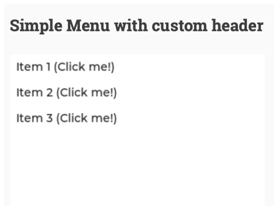
        - Screens can be deleted with lv_obj_delete(scr), but do **NOT** delete the active screen
        -

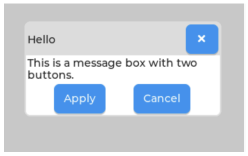
Widgets (skim over)

- Animation Image:
    - Instead of one source image, you set an array of multiple source images that supply "frames" in an animation
- Arc:
    - Consists of a background and a foreground(indicator and can be touch-adjusted) arc.
- Bar:
    - Has a background and an indicator. The length of the indicator against the background indicates the bar's current value.
    - Vertical bars can be created if the width of the Widget is smaller than its height.
- Button
    - Have no new features compared to the Base Widget. They are useful for semantic purposes and have slightly different default settings.
- Button Matrix
    - Lightweight way to display multiple Buttons in rows and columns
    - Buttons are not actually created but just virtually drawn on the fly.
- Calendar
    - It's a classic calendar that can(show days of month 7x7 matrix, show name of days, highlight the current day, highlight user-defined dates)
    - Calendar is an editable Widget which allows selecting and clicking the dates with encoder or keyboard navigation as well as pointer-type input devices.
- Canvas
    - A Canvas inherits from Image and extends it, enabling the user to draw anything(Rectangles, text, images, lines, arcs)
- Chart
    - Are used to visualize data.
    - Charts can show or hide individual data series,points,cursors
- Check Box:
    - It's created from a "tick box" and a label. When the Checkbox is clicked the tick box is toggled.



- Drop-down list:
    - Allows the user to select a value from a list
    - It's an editable Widget allowing list-item selection via encoder or keyboard navigation as well.
- Image:
    - Display images from flash (as arrays) or from files. Images can display symbols (LV_SYMBOL_...) as well.
- Image Button:

    - Is very similar to the simple 'Button' Widget.
    - It displays user-defined images for each state instead of drawing a rectangle.
    - You can set a left, right and center image, and the center image will be repeated to match the width of the Widget.
- Keyboard:
    - The Keyboard Widget is a special Button Matrix (lv_buttonmatrix) with predefined keymaps and other features to provide an on-screen virtual keyboard to write text into a Text Area (lv_textarea).
- Label:
    - A Label is the Widget used to display text.
- LED:
    - LEDs are rectangle-like (or circle) Widgets whose brightness can be adjusted. With lower brightness the color of the LED becomes darker.
- Line:
    - The Line Widget is capable of drawing straight lines between a set of points.
- List:
    - The List Widget is basically a rectangle with a vertical layout to which Buttons and Text can be added.
    - (**Sorting** a List using up and down buttons)
- Lottie
    - Is capable of parsing, rasterizing, and playing Lottie animations.
    - The Lottie animations are vector based animation. Think of it as the modern combination of SVG and GIF.
    - The animations can be downloaded from various sources, such as https://lottiefiles.com/ or you can create your own animations using, for example, Adobe After Effects.
- Menu
    - Used to easily create multi-level menus. It handles the traversal between pages automatically.
    - 

can be used to display task details in our project.
- Message Box:
    - Act as pop-ups. They are built from a content area with a helper to add text, an optional header and an optional footer with buttons.
    - 
Hello
This is a message box with two buttons.

if we wanted to do the task reminder notification on screen this could work ???
- Roller:

    - Allows you to simply select one option from a list by scrolling.
- Scale:
    - Allows you to have a linear scale with ranges and sections with custom styling.
- Slider:
    - Looks like a Bar supplemented with a knob. The knob can be dragged to set a value. Just like Bar, sliders can be vertical or horizontal.

Main Components (read last)
- Display: lv_display_t (**don't** confuse w/ **screen**) is a data type that represents a single display panel/hardware that displays LVGL-rendered pixels on device
- Have to do following for each display panel you want LVGL to use
    - create an lv_display_t object for it,
    - assign a Flush Callback for it, and
    - assign its Draw Buffer(s).
- Display object

Useful Additions to Project:

Scrolling (in examples)

Here is one recommended order of documents to read and things to play with while you are advancing your knowledge:

1. **If not already read, start with Introduction page of the documentation. (5 minutes)**
2. **Check out the Online Demos to see LVGL in action. (3 minutes)**
3. **If not already done, read the LVGL Basics (above). (15 minutes)**
4. **Set up an LVGL Simulator on PC. (10 minutes)**
5. **Have a look at some Examples and their code.**

6. Add LVGL to your project. See Add LVGL to Your Project or check out the ready-to-use Projects.
7. Read the Main Components pages to get a better understanding of the library. (2-3 hours)
8. Skim the documentation of Widgets to see what is available.
9. If you have questions go to the Forum.
10. Read the Contributing guide to see how you can help to improve LVGL. (15 minutes)