CODEEEEE - Gizelle Mendoza

READ the MFM

Reference Flowchars and Design:
https://excalidraw.com/#room=d5b2ef45e2ee62f96f25,mt3_PStLTJ_g79ltTAvUiQ

- Register a physical button press in LVGL as an event
- Connect button event to a widget

- Focus screen:
  - See poster

By maybe wednesday

Code adjustments for initializing LVGL UI generated by squareline(void set up of porting.ino):

```
lv_disp_t *display1 = lv_disp_get_default();

lv_disp_set_rotation(display1, LV_DISP_ROT_90);
```

Notes on Buttons:

Events:

- Besides Widgets, events can registered from displays and input devices as well
- Do this by changing the prefix of the functions from lv_obj_ to lv_display_ or lv_indev_

Example:
https://forum.lvgl.io/t/how-can-i-trigger-an-event-by-pressing-a-hardware-button-changing-the-screen/11630

Related to the input devices:

These are sent when an object is pressed/released etc. by the user. They are used not only for *Pointers* but can be used for *Keypad*, *Encoder* and *Button* input devices as well. Visit the overview of input devices section to learn more about them.

An input device usually means:

- Pointer-like input device like touchpad or mouse
- Keypads like a normal keyboard or simple numeric keypad
- Encoders with left/right turn and push options
- External hardware buttons which are assigned to specific points on the screen

- **LV_EVENT_PRESSED** The object has been pressed
- **LV_EVENT_PRESSING** The object is being pressed (sent continuously while pressing)
- **LV_EVENT_PRESS_LOST** The input device is still being pressed but is no longer on the object
- **LV_EVENT_SHORT_CLICKED** Released before LV_INDEV_LONG_PRESS_TIME time. Not called if dragged.
- **LV_EVENT_LONG_PRESSED** Pressing for LV_INDEV_LONG_PRESS_TIME time. Not called if dragged.
- **LV_EVENT_LONG_PRESSED_REPEAT** Called after LV_INDEV_LONG_PRESS_TIME in every LV_INDEV_LONG_PRESS_REP_TIME ms. Not called if dragged.
- **LV_EVENT_CLICKED** Called on release if not dragged (regardless to long press)
- **LV_EVENT_RELEASED** Called in every case when the object has been released even if it was dragged. Not called if slid from the object while pressing and released outside of the object. In this case, LV_EVENT_PRESS_LOST is sent.

More notes:

Squareline Studio simplifies UI creation and integrates with LVGL.

- Dont code UI components since Squareline hands it
- Focus on linking hardware interactions

Steps:

1. Create UI layout in squareline(use widgets)
    a. Set properties of widgets and assing event callback using squareline
    b. Export UI files
2. Open exported files in arduino
    a. This is the generated Arduino-compatible project with LVGL and the UI design files.

Code from chat that works:

(this works to assign a specific page and button to switch another specific page)

(need a better way to set buttons functionality)

Certainly! The `if` conditions in the `toggle` logic within the `loop()` function are used to alternate between two screens (`ui_Focus_Page` and `ui_Test_Screen`) based on the state of the `toggle` variable. Let me break it down step by step:

Code Fragment:

cpp Copy code

```
if (buttonState == LOW && lastButtonState == HIGH) {
    if (toggle) {
        _ui_screen_change(&ui_Focus_Page, LV_SCR_LOAD_ANIM_MOVE_LEFT, 500, 0, NUL
    } else {
        _ui_screen_change(&ui_Test_Screen, LV_SCR_LOAD_ANIM_MOVE_RIGHT, 500, 0, N
    }
    toggle = !toggle; // Alternate between screens
}
```

1. Button Press Detection:

cpp Copy code

```
if (buttonState == LOW && lastButtonState == HIGH)
```

- `buttonState`: Represents the current state of GPIO 13, read using `digitalRead(13)`.
    - `LOW`: Button is pressed (since we used `INPUT_PULLUP`, the pin is normally HIGH).
    - `HIGH`: Button is not pressed.
- `lastButtonState`: Tracks the button's state from the previous loop iteration.
    - This comparison (`LOW && HIGH`) ensures the code runs **only when the button transitions from not pressed to pressed**. This prevents repeated triggers due to button bouncing or holding the button down.

Here the pink is what I think makes the pages lag when pressing the button maybe??? So like we have to hold pin 13 down for a while for it to change pages back and forth.

2. Toggle State Check:

cpp Copy code

```
if (toggle) {
    _ui_screen_change(&ui_Focus_Page, LV_SCR_LOAD_ANIM_MOVE_LEFT, 500, 0, NULL);
} else {
    _ui_screen_change(&ui_Test_Screen, LV_SCR_LOAD_ANIM_MOVE_RIGHT, 500, 0, NULL
}
```

- **Purpose**: Based on the value of `toggle`, the program decides which screen to load:
    - `if (toggle)`: When `toggle` is `true`, it switches to the `ui_Focus_Page` screen with a leftward animation.
    - `else`: When `toggle` is `false`, it switches to the `ui_Test_Screen` with a rightward animation.

3. Toggle State Update:

cpp Copy code

```
toggle = !toggle;
```

- **Purpose**: Toggles the value of `toggle` between `true` and `false` after each button press:
    - If `toggle` is `true`, it becomes `false`.
    - If `toggle` is `false`, it becomes `true`.
- **Why Important?**:
    - This alternation ensures the button press switches the screen each time:
        1. First press: Loads `ui_Focus_Page` (when `toggle` was `true`).
        2. Second press: Loads `ui_Test_Screen` (when `toggle` was `false`).
        3. Third press: Loads `ui_Focus_Page` again, and so on.

• `_ui_screen_change` **Parameters**:
  • `&ui_Focus_Page` or `&ui_Test_Screen` : Pointer to the screen to display.
  • `LV_SCR_LOAD_ANIM_MOVE_LEFT` or `LV_SCR_LOAD_ANIM_MOVE_RIGHT` : Animation type for the screen transition.
  • `500` : Animation duration in milliseconds.
  • `0` : Delay before starting the animation.
  • `NULL` : No callback function is provided (can be used to execute additional logic after the screen transition completes).	

4. Updating `lastButtonState` : cpp Copy code `lastButtonState = buttonState;` • **Purpose**: Ensures the code tracks the current button state for the next loop iteration. • This prevents repeated triggers and allows the `if (buttonState == LOW && lastButtonState == HIGH)` condition to work properly in the next cycle.	

~~Example of our question with no solution:~~

~~https://forum.squareline.io/t/physical-button-controlled-gui-design/1358~~

~~Example not physical button but shows how to add frames within a page:~~

~~https://www.figma.com/design/ELIjoaLu58ujOhXW2WrzEu/FF-Smart-Animation-Example?node-id=0-1&node-type=canvas&t=bKHTMxp1flrWHkoK-0~~

Notes on Images:

https://docs.lvgl.io/8.3/overview/image.html

Working with rollers(for tasks on screen):

8 am do the thing

9 am another thing

10 am cry

11 am eat

12 pm eat again

1 pm study

Code for the screen1 ui file:
lv_roller_set_options(ui_Roller1, "Option 1\nOption 2\nOption 3", LV_ROLLER_MODE_NORMAL);

Read:

https://docs.lvgl.io/8.4/widgets/core/label.html

Notes on labels:

- Labels are object type used to display text
- Allocate text on the label at runtime:
    - lv_label_set_text(label, "New text")
    - Allocate a buffer dybnamically and the provided string will be copiued into that buffer.
    - Then we dont need to keep the text we passed to the lv_label_set_text in scope after that function returns.

Read:

https://docs.arduino.cc/language-reference/en/functions/external-interrupts/attachInterrupt/

Notes:

Notes on events:

Sending events

To manually send events to an object, use lv_event_send(obj,<EVENT_CODE> &some_data).

For example, this can be used to manually close a message box by simulating a button press.

Input device events

- LV_EVENT_CLICKED Called on release if an object did not scroll (regardless of long press)

old code(swtihc page):

// if (ui_Test_Screen == current_screen) {
 	
// _ui_screen_change(&ui_Focus_Page, LV_SCR_LOAD_ANIM_NONE, 50, 100, NULL); // Load Focus Page
 	
// }
 	
// else {
 	
// _ui_screen_change(&ui_Test_Screen, LV_SCR_LOAD_ANIM_NONE, 50, 100, NULL); // Load Test Screen
 	
// }

old/first Up button code	old/first down button code

// //get text from top & center
// char* topTaskText = lv_label_get_text(ui_topTask);
// char* centerTaskText = lv_label_get_text(ui_centerTask);
 	
// //move center down and top to center
// lv_label_set_text(ui_bottomTask, centerTaskText);
	// //get text from top & center
	// char* bottomTaskText = lv_label_get_text(ui_bottomTask);
	// char* centerTaskText = lv_label_get_text(ui_centerTask);
	
	// //move center to top and bottom to center

```
// lv_label_set_text(ui_centerTask, topTaskText);

// // top task prev to top task
// lv_label_set_text(ui_topTask, taskList[taskCounter]);
```

```
// lv_label_set_text(ui_topTask, centerTaskText);
// lv_label_set_text(ui_centerTask, bottomTaskText);

// // new to bottom
// lv_label_set_text(ui_bottomTask, taskList[taskCounter]);
}
```

Switch creen trial functions:

```
// void switchScreen(){
//     Serial.println("Switch screen activated:");
//     lv_obj_t*  current_screen = lv_scr_act();

//     // To manually send events to an object, use
//     lv_event_send(ui_UpButton, LV_EVENT_CLICKED, tasksList[taskCounter]);
//   }
```

READ:

- https://docs.lvgl.io/8.4/widgets/extra/msgbox.html

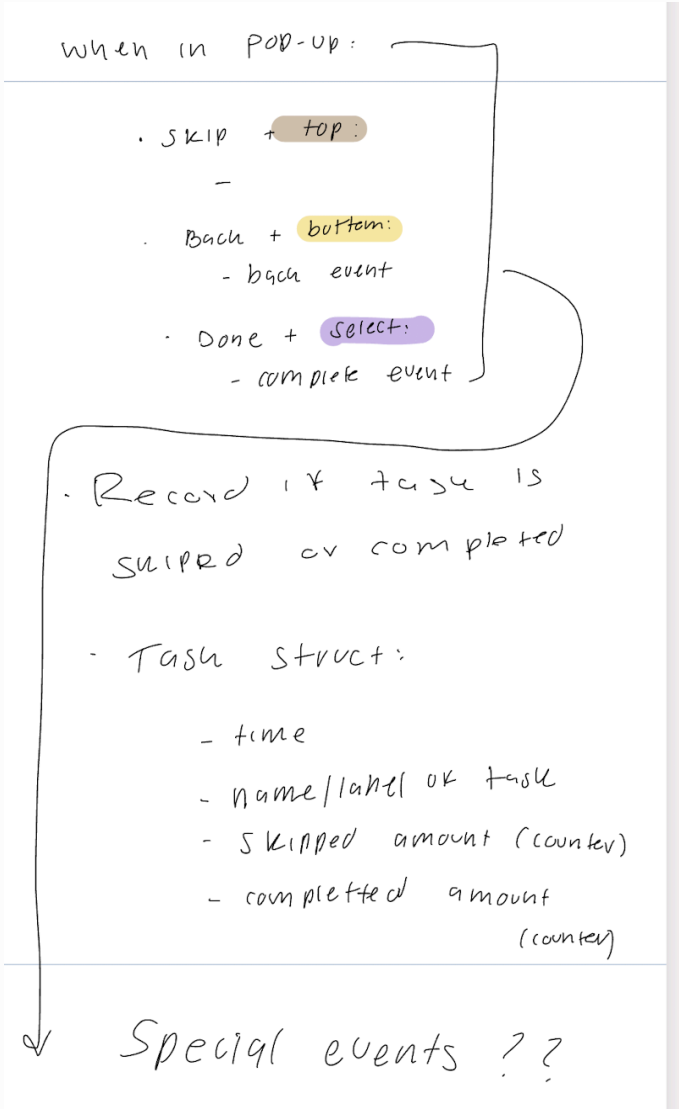
NOTES on message box:

- Message boxes act as pop-ups
- Built from:
    - Container
    - Title
    - Optional close button
    - Text
    - Optional button
- Text broken into multiple lines automatically
- The height will be set automatically to include the text and the buttons

- The message box can be modal(blocking clicks on the rest of the screen)
- Create a message box:
- lv_msgbox_create(parent, title, txt, btn_txts[], add_close_btn) creates a message box.
    - parent is NULL the message box will be modal.
    - title and txt are strings for the title and the text.
    - btn_txts[] is an array with the buttons' text. E.g. const char * btn_txts[] = {"Ok","Cancel", NULL}
    - add_close_btn can be true or false to add/don't add a close button.

lv_event_send(the object(widget), LV_EVENT_CLICKED, NULL);

Notes on what needs to be done for decision buttons:



How to make a task struct:

https://www.tutorialspoint.com/structs-in-arduino-program

```
sturct task{
        int time;
        String name; // probably a c-string (char array)
        int skippedAmount;
        int completedAmount;
}
task taskList[number_of_tasks] = {all, the, tasks};
taskList[aNumber].time = some_time;
taskList[aNumber].name = aName;
if(task_complete_clicked){
        taskList[aNumber].completedAmount++;
}
```

// should look something like this,,,, Gucci?

Task struct

Array of structs

10 instances of this

Does this look right ??? i put this before the void set up

Probably, did it compile?

```
// struct for task

struct task{

 char time[10];

 char name[100];

 int skippedAmount;

 int completedAmount;

}

// set up struct array

task taskList[10]= {

     {"8:00am", "Wake up", 0, 0},

     {"9:00am", "Eat", 0, 0},

     {"10:00am", "Go to school", 0, 0},

     {"11:00am", "Class", 0, 0},

     {"3:00pm", "Lunch", 0, 0},

     {"5:00pm", "Leave school", 0, 0},

     {"6:00pm", "Gym", 0, 0},

     {"7:00pm", "Dinner", 0, 0},

     {"8:00pm", "Shower", 0, 0},

     {"10:00pm", "Sleep", 0, 0},
```

```
};
```