Creating a text box and keyboard

Detailed Tutorial

In this section, we will learn how to create text boxes and numeric keyboards in lvgl.

The text box component is Textarea: Text area (lv_textarea) — LVGL documentation

The keyboard component is a Button matrix: Button matrix (lv_btnmatrix) — LVGL documentation

First, we copy the code of the lvgl basic structure and add touch-related code.

We first declare a function to create a text box create_textarea() and a function to create a small keyboard create_btnm()

The code is as follows:

```
# 创建一个文本输入区域 Create a text input area
def create_textarea():
    # 在当前活动屏幕上创建文本输入框对象
    # Create a textarea object on the current active screen
    ta = lv.textarea(lv.scr_act())

    # 将文本框对齐到屏幕顶部中间位置,x偏移0,y偏移10像素
    # Align the textarea to the top-middle of screen with x offset 0 and y offset
10 pixels
    ta.align(lv.ALIGN.TOP_MID, 0, 20)

    # 设置文本框为单行模式
    # Set textarea to single line mode
    ta.set_one_line(True)

    return ta

# 创建一个按钮矩阵 Create a button matrix
def create_btnm(ta):
    # 定义按钮矩阵的布局映射,包含数字键和功能键
    # Define button matrix layout map including number keys and function keys
    btnm_map = [
        "Q", "W", "E", "R", "T", "Y", "U", "I", "O", "P", "\n",
        "A", "S", "D", "F", "G", "H", "J", "K", "L", "\n",
        "Shift", "Z", "X", "C", "V", "B", "N", "M", lv.SYMBOL.BACKSPACE, "\n",
        "alt", "Space", lv.SYMBOL.NEW_LINE, ""
    ]

    # 在当前活动屏幕上创建按钮矩阵对象
    # Create a button matrix object on the current active screen
    btnm = lv.btnmatrix(lv.scr_act())
```

```
    # 设置按钮矩阵的大小为200x150像素
    # Set the size of button matrix to 200x150 pixels
    btnm.set_size(640, 300)

    # 将按钮矩阵对齐到屏幕底部中间位置,x偏移0,y偏移-10像素
    # Align the button matrix to bottom-middle of screen with x offset 0 and y
offset -10 pixels
    btnm.align(lv.ALIGN.BOTTOM_MID, 0, -10)

    # 添加按钮点击事件处理函数
    # Add event handler for button clicks
    btnm.add_event(lambda e: btnm_event_handler(e, ta), lv.EVENT.VALUE_CHANGED,
None)

    # 清除点击焦点标志,以保持文本框始终获得焦点
    # Clear click focusable flag to keep textarea focused when buttons are
clicked
    btnm.clear_flag(lv.obj.FLAG.CLICK_FOCUSABLE)

    # 设置按钮矩阵的布局映射
    # Set the layout map for button matrix
    btnm.set_map(btnm_map)
```

> Because our keyboard events need to operate on the text box component, when we create the text box, we return the text box object as the return value, and then pass it as a parameter to the keyboard callback function.

Then implement the button callback function btnm_event_handler(), and then handle the button pressing event in the callback function

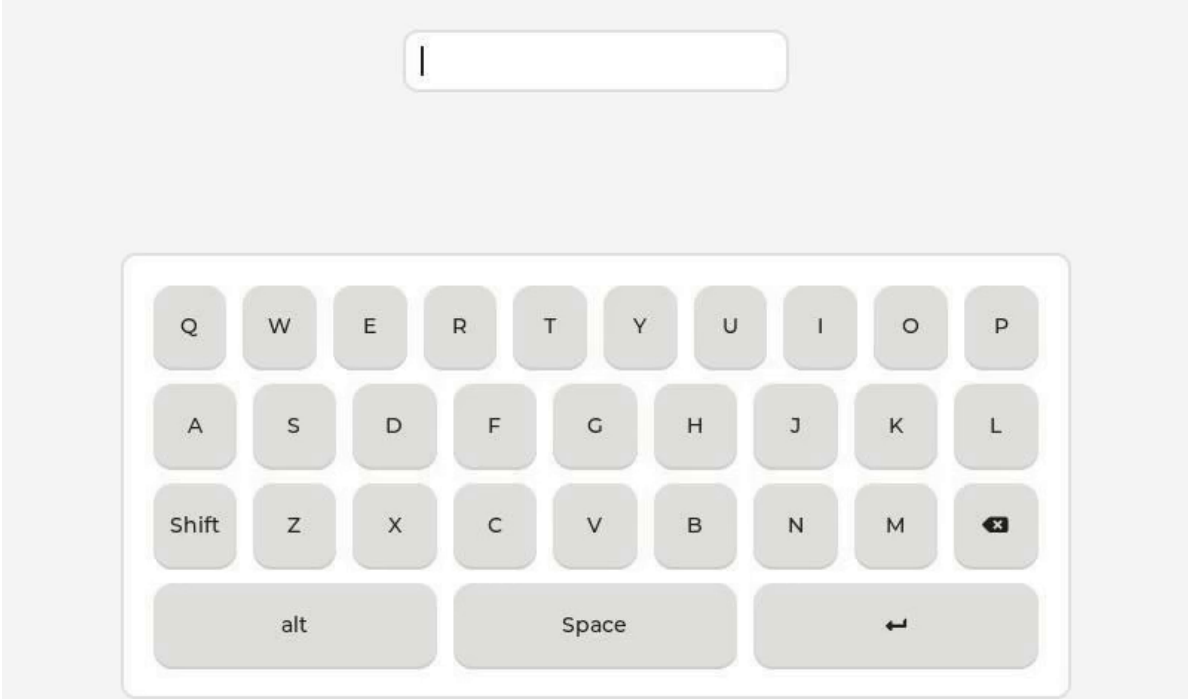
```
def btnm_event_handler(e, ta):
    # 获取触发事件的按钮矩阵对象
    # Get the button matrix object that triggered the event
    obj = lv.btnmatrix.__cast__(e.get_target())

    # 获取被选中按钮的文本
    # Get the text of the selected button
    txt = obj.get_btn_text(obj.get_selected_btn())

    # 根据按钮文本执行相应操作
    # Execute corresponding action based on button text
    if txt == lv.SYMBOL.BACKSPACE:
        # 如果是退格键,删除一个字符
        # If backspace, delete one character
        ta.del_char()
    elif txt == lv.SYMBOL.NEW_LINE:
        # 如果是回车键,发送READY事件
        # If new line, send READY event
        lv.event_send(ta, lv.EVENT.READY, None)
    elif txt in ["Shift", "alt"]:
        # 如果是Shift或alt键,不做任何操作
        # If Shift or alt, do nothing
        return
    elif txt == "Space":
```

```
        # 如果是空格键,添加空格
        # If Space, add a space character
        ta.add_text(" ")
    else:
        # 其他情况,将按钮文本添加到文本区域
        # Otherwise, add the button text to text area
        ta.add_text(txt)
```

We click Run and we can see the following effect:



Complete code

For the complete code, please refer to [Source Code Summary / 12.Lvgl / 05.lvgl_text]

```
import lvgl as lv
from media.display import *
import time  # 显式导入time模块 / Explicitly import time module

from machine import TOUCH
class touch_screen():
    def __init__(self):
        self.state = lv.INDEV_STATE.RELEASED

        self.indev_drv = lv.indev_create()
        self.indev_drv.set_type(lv.INDEV_TYPE.POINTER)
        self.indev_drv.set_read_cb(self.callback)
        self.touch = TOUCH(0)

    def callback(self, driver, data):
        x, y, state = 0, 0, lv.INDEV_STATE.RELEASED
        tp = self.touch.read(1)
        if len(tp):
            x, y, event = tp[0].x, tp[0].y, tp[0].event
            if event == 2 or event == 3:
                state = lv.INDEV_STATE.PRESSED
```

```
        data.point = lv.point_t({'x': x, 'y': y})
        data.state = state


# 显示屏分辨率配置 / Display resolution configuration
DISPLAY_WIDTH = 640   # 显示屏宽度 / Display width
DISPLAY_HEIGHT = 480  # 显示屏高度 / Display height

def disp_drv_flush_cb(disp_drv, area, color):
    """
    显示驱动刷新回调函数
    Display driver flush callback function

    Args:
        disp_drv: 显示驱动对象 / Display driver object
        area: 刷新区域 / Refresh area
        color: 颜色数据 / Color data
    """
    global disp_img1
    Display.show_image(disp_img1)  # 显示图像缓冲区 / Show image buffer
    time.sleep(0.01)  # 适当延时确保显示稳定 / Small delay to ensure stable display
    disp_drv.flush_ready()  # 通知LVGL刷新完成 / Notify LVGL that flush is complete

def display_init():
    """
    显示设备初始化函数
    Display device initialization function
    """
    # 初始化ST7701显示屏 / Initialize ST7701 display
    Display.init(
        Display.ST7701,
        width=DISPLAY_WIDTH,
        height=DISPLAY_HEIGHT,
        to_ide=True  # 启用IDE显示功能 / Enable IDE display function
    )

    # 初始化媒体管理器 / Initialize media manager
    MediaManager.init()

def lvgl_init():
    """
    LVGL初始化函数
    LVGL initialization function
    """
    global disp_img1

    # 初始化LVGL库 / Initialize LVGL library
    lv.init()

    # 创建显示缓冲区 / Create display buffer
    disp_img1 = image.Image(
        DISPLAY_WIDTH,
        DISPLAY_HEIGHT,
        image.BGRA8888  # 使用BGRA8888颜色格式 / Use BGRA8888 color format
    )

    # 创建显示驱动 / Create display driver
    disp_drv = lv.disp_create(DISPLAY_WIDTH, DISPLAY_HEIGHT)
```

```
    # 设置显示缓冲区 / Set display buffers
    disp_drv.set_draw_buffers(
        disp_img1.bytearray(),
        None,  # 单缓冲模式 / Single buffer mode
        disp_img1.size(),
        lv.DISP_RENDER_MODE.DIRECT  # 直接渲染模式 / Direct rendering mode
    )


    # 设置刷新回调函数 / Set flush callback
    disp_drv.set_flush_cb(disp_drv_flush_cb)
    tp = touch_screen()


def display_deinit():
    os.exitpoint(os.EXITPOINT_ENABLE_SLEEP)
    time.sleep_ms(50)
    # deinit display
    Display.deinit()
    # release media buffer
    MediaManager.deinit()


def btnm_event_handler(e, ta):
    # 获取触发事件的按钮矩阵对象
    # Get the button matrix object that triggered the event
    obj = lv.btnmatrix.__cast__(e.get_target())

    # 获取被选中按钮的文本
    # Get the text of the selected button
    txt = obj.get_btn_text(obj.get_selected_btn())

    # 根据按钮文本执行相应操作
    # Execute corresponding action based on button text
    if txt == lv.SYMBOL.BACKSPACE:
        # 如果是退格键,删除一个字符
        # If backspace, delete one character
        ta.del_char()
    elif txt == lv.SYMBOL.NEW_LINE:
        # 如果是回车键,发送READY事件
        # If new line, send READY event
        pass
    elif txt in ["Shift", "alt"]:
        # 如果是Shift或alt键,不做任何操作
        # If Shift or alt, do nothing
        return
    elif txt == "Space":
        # 如果是空格键,添加空格
        # If Space, add a space character
        ta.add_text(" ")
    else:
        # 其他情况,将按钮文本添加到文本区域
        # Otherwise, add the button text to text area
        ta.add_text(txt)

def lvgl_deinit():
    global disp_img1
```

```
    lv.deinit()
    del disp_img1

# 创建一个文本输入区域 Create a text input area
def create_textarea():
    # 在当前活动屏幕上创建文本输入框对象
    # Create a textarea object on the current active screen
    ta = lv.textarea(lv.scr_act())

    # 将文本框对齐到屏幕顶部中间位置,x偏移0,y偏移10像素
    # Align the textarea to the top-middle of screen with x offset 0 and y offset
10 pixels
    ta.align(lv.ALIGN.TOP_MID, 0, 20)

    # 设置文本框为单行模式
    # Set textarea to single line mode
    ta.set_one_line(True)

    return ta

# 创建一个按钮矩阵 Create a button matrix
def create_btnm(ta):
    # 定义按钮矩阵的布局映射,包含数字键和功能键
    # Define button matrix layout map including number keys and function keys
    btnm_map = [
        "Q", "W", "E", "R", "T", "Y", "U", "I", "O", "P", "\n",
        "A", "S", "D", "F", "G", "H", "J", "K", "L", "\n",
        "Shift", "Z", "X", "C", "V", "B", "N", "M", lv.SYMBOL.BACKSPACE, "\n",
        "alt", "Space", lv.SYMBOL.NEW_LINE, ""
    ]

    # 在当前活动屏幕上创建按钮矩阵对象
    # Create a button matrix object on the current active screen
    btnm = lv.btnmatrix(lv.scr_act())

    # 设置按钮矩阵的大小为200x150像素
    # Set the size of button matrix to 200x150 pixels
    btnm.set_size(640, 300)

    # 将按钮矩阵对齐到屏幕底部中间位置,x偏移0,y偏移-10像素
    # Align the button matrix to bottom-middle of screen with x offset 0 and y
offset -10 pixels
    btnm.align(lv.ALIGN.BOTTOM_MID, 0, -10)

    # 添加按钮点击事件处理函数
    # Add event handler for button clicks
    btnm.add_event(lambda e: btnm_event_handler(e, ta), lv.EVENT.VALUE_CHANGED,
None)

    # 清除点击焦点标志,以保持文本框始终获得焦点
    # Clear click focusable flag to keep textarea focused when buttons are
clicked
    btnm.clear_flag(lv.obj.FLAG.CLICK_FOCUSABLE)

    # 设置按钮矩阵的布局映射
    # Set the layout map for button matrix
    btnm.set_map(btnm_map)
```

```python
def main():
    """
    主函数
    Main function
    """
    try:
        # 初始化显示设备和LVGL / Initialize display device and LVGL
        display_init()
        lvgl_init()
        ta = create_textarea()
        create_btnm(ta)
        print("LVGL initialization completed")

        # LVGL主循环 / LVGL main loop
        while True:
            # 运行LVGL定时器处理程序 / Run LVGL timer handler
            period = lv.timer_handler_run_in_period(1)
            ta.add_state(lv.STATE.FOCUSED)
            time.sleep_ms(period)

    except Exception as e:
        display_deinit()
        lvgl_deinit()
        print(f"Error occurred: {e}")

if __name__ == "__main__":
    main()
```