lgui framework 使用说明

零、下载说明

> 在linux环境下执行 git clone https://github.com/d-power/LGUI3.git
> 如果直接下载了zip包, 在linux环境下执行 unzip LGUI3-master.zip
> 在windows环境下解压zip包，会导致软链接丢失，需要在linux环境下执行 ./build.sh 已重新链接。
> 在Windows下运行时需要用libs_win替换libs
> 在Linux机器上运行时需要用libs_arm替换libs

一、框架简介

- lgui framework全部使用lua语言实现，采用MVC设计思想
- lua是最适合嵌入式设备执行的动态语言，有支持热更新，无需重新编译，可动态配置等优点
- lgui framework参考了react和微信/支付宝小程序的部分设计思想
- lgui framework只专注于实现界面交互逻辑，具体的业务逻辑，需要开发者使用C语言实现
- lgui framework提供了与C库双向交互的方式
- lgui framework的实现就是围绕table的使用设计的

二、目录结构

```
.
├── README.md
├── app.lua
├── config
│   ├── env.lua
│   ├── event.lua
│   ├── language-ZH.lua
│   ├── router.lua
│   └── service.lua
├── libs
│   ├── lgui_dispatch.lua
│   ├── lgui_draw.lua
│   ├── lgui_parser.lua
│   ├── lgui_private.lua
│   └── lgui_public.lua
├── main.lua
├── pages
│   ├── home.lua
│   └── homeview.lua
├── resources
│   ├──
│   └──
├── service
│   ├──
│   └──
├── start.lua
├── utils
│   ├──
└── └──
```

三、基本概念

(1) view

> 表示lgui中最基本的界面布局，是每张界面的必要元素，view文件放置于pages文件夹中

(2) ctrl

> 表示view中的每个控件，每个view是由多个ctrl组合而来

(3) controller

> 表示每个view对应的控制逻辑，controller文件放置于pages文件夹中

(4) service

> lua framework处理与C库交互事件与逻辑的模块，通常由开发者自行实现

(5) page

> 一个page表示一张独立界面，每个page中包含一个view和一个controller

(6) router

> page的索引，在config/router.lua中配置，当set_page时，需要传入page。若router中配置如下：
>
> ```
> local router =
> {
>     -- page(home)
>     home = "pages/home",
>     -- page(example)
>     example = "pages/setting/setting",
> }
> return router
> ```
>
> pages文件夹中应该存在以下文件：
>
> ```
> -- controller
> pages/home.lua
> -- view
> pages/homeview.lua
> -- controller
> pages/setting/setting.lua
> -- view
> pages/setting/settingview.lua
> ```
>
> 当需要切换为setting界面时，传入example，则lgui会自动加载pages/setting/settingview.lua和pages/setting/setting.lua

(7) app

> 由用户实现的自定义逻辑，即app.lua

(8) lgui entry

> main.lua是lgui框架的入口脚本，使用lua main.lua即可启动界面程序。与使用sh build.sh命令执行shell脚本同理

(9) lgui c library

> 由C语言实现的lgui绘图库，实际的绘图、刷新、控件管理，都是由这个c库完成的。lgui framework是基于lgui c library的封装

(10) lua table

> lua语言的核心数据结构，相当于c++的hash map或python的dict，即key-value表示的键值对，key和value都可以是任意类型（nil除外），一个典型的table可以表示为
>
> ```
> local t = {key1 = value1, key2 = value2}
> ```

四、lgui framework文件说明

> main.lua : 程序入口，用户无需修改其中代码

> app.lua : 全局用户代码，app.lua内的参数和函数，可以由任意界面通过get_app()使用

> start.lua : 启动第一个home界面，加载app.lua后才会加载start.lua

> config/ : 存放配置文件

> config/env.lua : 用户环境配置，包含以下参数

参数名	参数类型	必填(Y/N)	说明
path	string	Y	lgui framework绝对路径
cpath	string	Y	依赖c库的路径
ttf	string	Y	ttf字库文件
fb	string	Y	framebuffer设备节点
rorate	number	N	是否旋转，0：不旋转，1：90度，2：180度，3：270度
disp_flush_period	number	N	显示刷新频率(ms)，0表示默认(20)
img_cache_num	number	N	图片缓存最大数量，0表示默认(24)
img_cache_max_size	number	N	单张图片缓存的最大字节。默认大小为屏幕大小
ttf_cache_num	number	N	界面字型的最大数量，0表示默认(64)
touch_period	number	N	触摸屏取读频率(ms)，0表示默认(25)
touch_longpress_period	number	N	触摸屏长按多久上报长按事件(ms)，0表示默认(500)
key_period	number	N	按键取读频率(ms)，0表示默认(50)
timer_period	number	N	timer上报频率(ms)，会触发service timer，0表示默认(关闭)
background	string	Y	背景图片
log_level	number	Y	打印等级
log_title	table	Y	打印等级对应的title
log_color	table	Y	打印等级对应的color
shared_file	table	Y	需要预加载 shared 文件
screensaver_time	table	Y	无触摸进入屏保的超时时间(s)。当触发屏保时，会触发service lgui_event_screensaver。0表示默认(关闭)。

> config/event.lua : 框架事件列表，用户无需关注

> config/language-ZH.lua : 语言文件，命名规则为language-XX.lua，用户可以通过set_language("XX")函数改变语言

> config/router.lua : 路由文件，已在**基本概念 (6) router**中说明

> config/service.lua : 服务配置文件，配置C库传递给lgui framework的事件。其中，key表示service文件名，table内的字符串表示实际的事件名。实例如下：
>
> ```
> local service =
> {
>     timer = {"lgui_event_timer"},
> }
> return service
> ```
>
> 在service文件夹中，则应该存在service/timer.lua文件夹，内容如下：
>
> ```
> local timer =
> {
>     lgui_event_timer = function(x, y, z)
>     end,
> }
> return timer
> ```

> libs/ : lgui framework的核心调度库和函数库，用户无需关心

> pages/ : 界面文件，已在**基本概念 (6) router**中说明

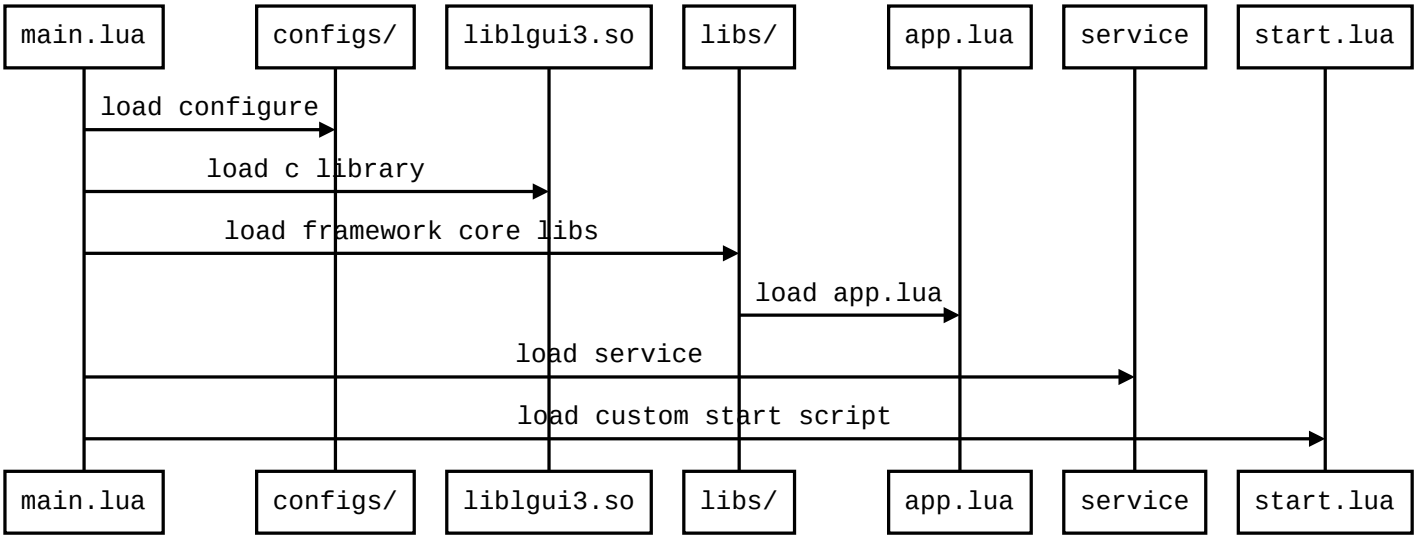
> resources/ : 存放用户的资源文件，比如图片，字库等等

> service/ : 存放用户自定义的服务逻辑

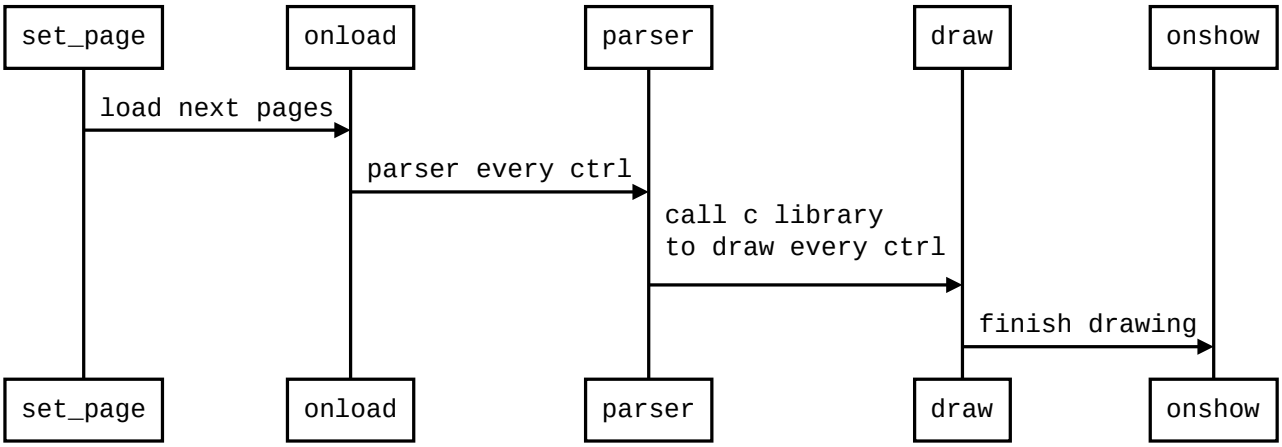
> share/ : 共享ctrl目录，在view中可以载入共享控件，以减少多张界面布局重复时的代码量

五、时序图

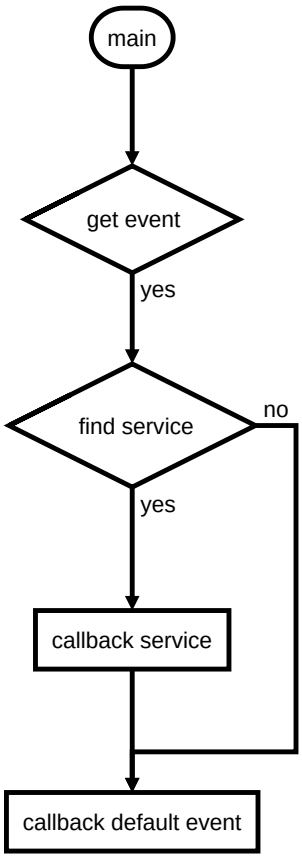
(1) lgui framework starting



(2) page loading



(3) service callback


main
get event
yes
find service
no
yes
callback service
callback default event

六、lgui framework特性描述

1、ctrl加载顺序

每个view中，按照数组索引顺序加载，即写在前面的ctrl，先加载，在下层，写在后面的ctrl，后加载，在上层。
例如：

```
local view =
{
    {
        type = "img",
        ...
    },
    {
        type = "btn",
        ...
    },
}
return view
```

btn控件将在img控件上层绘图，若btn控件与img有重叠，则btn重叠部分会盖在img上。

2、setdata特性

view中，任意ctrl的任意属性（type除外），都可以被 **"{{}}"** 替换，在解析view时，会自动使用controller中data的对应成员替换。第一次加载view时，替换为用户实际指定属性。已加载过view，通过set_data(obj)接口改变data中某个成员的值时，lgui framework会自动寻找关联控件，并控制c library更新该控件。
例如：

```
-- homeview.lua
local view =
{
    {
        type = "img",
        position = {x = "{{pos_x}}", y = "{{pos_y}}"}
        attr = {},
    }
}
return view


-- home.lua
local controller =
{
    data =
    {
        pos_x = 10,
        pos_y = 20,
    },
    onload = function() end,
    onshow = function() end,
    ondestroy = function() end,
}
return controller
```

加载view时，img控件的x值会被10替换，y值会被20替换
界面加载后，若用户想更改该控件位置，可以调用

```
set_data({pos_x = 100, pos_y = 200})
```

lua framework则会自动更新该控件的位置
其他属性同理

3、view中ctrl事件关联controller

view中的每个ctrl都可以关联到一个或多个controller函数（通过action.bind字段实现）
注：若ctrl type没有响应事件，则设置action.bind是无效的（比如img控件，是无响应的）
例如：

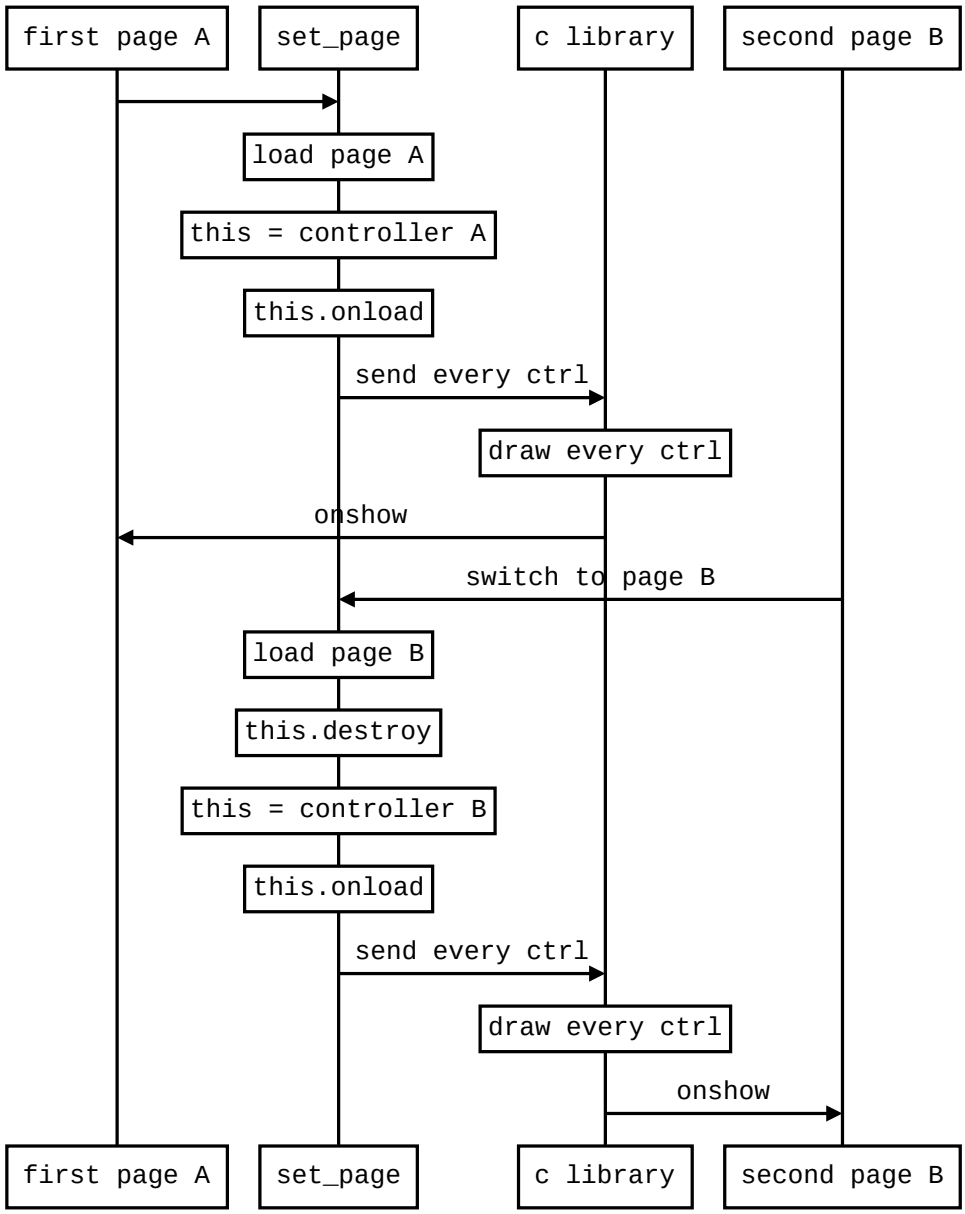
```
-- homeview.lua
local view =
{
    {
        type = "btn",
        position = {x = 0, y = 0}
        attr = {res_release = "0.png", res_press = "1.png", res_disable = "2.png"},
        action = {bind = {up = "btn_up", down = "btn_down", long = "btn_long"}}
    }
}
return view


-- home.lua
local controller =
{
    data      = {},
    onload    = function() end,
    onshow    = function() end,
    ondestroy = function() end,

    btn_up    = function()
        log(3, "btn up")
    end,

    btn_down  = function()
        log(3, "btn down")
    end,

    btn_long  = function()
        log(3, "btn longpress")
    end,
}
return controller
```

当用户针对btn产生不同触摸行为时，lgui framework会自动回调bind中对应的函数

4、this特性

this是一个全局变量（可以在任意位置使用），表示当前的controller。当希望获取当前controller的某个data时，可以直接使用this.data.xx，当希望使用当前controller中函数时，this.somefunc()。切换一张新界面时（例如page A切换到page B），首先回调A.destroy，紧接着this=B。此时this已经被替换为新的controller。

时序图如下：



5、热更新特性

> 由于每次view刷新，都是动态加载的，所以可以很容易做到热更新，例如当前处于home page，通过网络下载setting page并替换到router指定位置，在加载setting page时，即实际画替换后的界面。在此过程中，程序无需重启。

6、lgui中的背景特性

> lgui3中，背景作为一个特殊控件存在，与每个page中的view相对独立，同时又是全局的根对象，即未指定父控件的ctrl，自动成为背景的子。

7、ctrl position特性

> lgui3中，控件位置有两种设置方式，绝对位置和相对位置。
> 绝对位置即position.x / position.y

> 当position的align属性不为nil，则ctrl会被设置为相对位置，绝对位置的x / y会被忽略
> 相对位置的具体特性，参考**ctrl的父子特性**

8、图片资源自动解析特性

> lgui3中，可以支持bmp、png、jpeg(jpg)三种图片资源格式，用户只需传递图片名即可，c library会自动解析图片格式。解析步骤分两步（1）判断文件后缀名（2）校验文件头。若用户的资源文件实际是jpg，但文件名后缀写为png，c library会跳过该图片，不会显示。

9、ctrl的父子特性

> lgui3中，ctrl可以有父子或参考关系，当指定了name属性，则表示这个ctrl可以作为参考控件或父控件。
>
> - 当一个ctrl作为参考控件（ref ctrl），其他ctrl可以根据ref ctrl的位置，设置为相对位置（只作为位置参考，不作为父子）。当一个ctrl作为父控件时，该ctrl隐藏（或移动）时，关联的子控件，也会隐藏（或移动），子控件的显示范围在父控件之内。
> - 当一个ctrl，指定了父控件，未指定参考控件，则自动参考父控件的位置设置相对位置。
> - 当一个ctrl，指定了参考控件，未指定父控件，则该ctrl是背景的子，根据参考控件设置相对位置。
> - 当一个ctrl，即有参考控件，又有父控件，且参考控件与父控件不同时，控件位置跟随参考控件，变化跟随父控件。
> - 举例：

```lua
-- homeview.lua
local util_align = require "utils/align"
local view =
{
    {
        type        = "img",
        -- 未指定参考控件，则参考背景，ALIGN_CENTER表示居中
        -- 通常背景是全屏，所以该img控件会基于全屏幕居中显示
        position    = {align = util_align.ALIGN_CENTER},
        attr        = {res = "62.bmp"},
        -- 指定name属性，表示这个img可以作为其他控件的父控件或参考控件
        name        = "ctrl_0"
    },
    {
        type        = "img",
        -- ctrl_0是参考控件，参考ctrl_0的位置，居中显示
        position    = {ref = "ctrl_0", align = util_align.ALIGN_CENTER},
        -- 这个控件的父控件是背景
        attr        = {res = "0.bmp"},
    },
    {
        type        = "img",
        -- 未指定参考控件，则参考父控件的位置，居中显示
        position    = {align = util_align.ALIGN_CENTER},
        -- ctrl_0是该控件的父控件，且ctrl_0隐藏或移动时，会关联到这个控件
        attr        = {res = "1.bmp", parent = "ctrl_0"},
    },
}
return view
```

10、service抢占默认事件特性

> 用户可以在service中，定义lgui framework的默认事件，则当lgui触发默认事件时，会优先获得该事件，待service执行完，lgui framework才会执行对应事件函数。

例如用户可以在service中自行实现lgui_event_longpress事件的接收函数，则当lgui_event_longpress事件触发时，service会先收到，先执行，待service执行完，lgui framework才会执行lgui_event_longpress的默认处理。

11、文本重新着色特性

对于单行的文本，可以使用命令(\x1b)实现文本的重新着色。例如 "写一个\x1bff0000 红色\x1b文本"。 将文本“红色”重新着色为红色。除了拼接字符，框架还提供了函数 get_recolor_str 辅助拼接着色文本，详情查看函数说明。
请注意，文本重新着色只能在一行中进行，如果着色的文本中存在’\n‘或 使用 MODE_BREAK 模式导致换行，则新行的文本不会重新着色
目前能使用这一特性的控件：list、text

12、common页面特性

lgui3 中除了普通的页面，框架会虚拟common界面。虚拟的common界面会在普通界面的顶部。当普通界面切换时，common界面不会被删除。common界面的关闭时机由用户决定。
common 界面主要用于多个界面存在的共性控件。共性控件可以在common中实现，这样就不用在每个界面重复实现共性控件的配置以及方法。例如：弹窗，顶部状态栏等等。
配置方法：common界面的配置方式与普通界面一样。在config/router.lua中配置名字以及对应的文件路径。
支持同时添加多个common界面。已页面的名字区分。
全局参数 common 中存放common界面的controller。使用 common[name] 获取名字为 name 的common界面controller。
使用函数 common_add 添加common页面。
使用函数 common_destroy 删除common界面。
使用函数 common_hidden 隐藏common界面。
使用函数 common_show 显示common界面。
使用函数 common_check 判断common界面是否存在。
函数的使用方法详情请看函数说明

13、手势导航特性

从边缘向内滑动时, 会将手势导航事件发送到当前页面, 判断并执行 gesture 函数，并将方向以参数的形式返回。
当开启手势导航特性时, 左、右、下 边缘 20 像素内，点击事件被手势控件吸收，覆盖的控件不会被触发。
目前支持左滑("left")、右滑上滑("right")、上滑("up")。
默认不开启手势导航，颜色为透明。
使用函数 lgui_gesture_open() 开启手势导航。
使用函数 lgui_gesture_close() 关闭手势导航。
使用函数 lgui_gesture_set_color(color) 设置颜色。

七、函数说明

1、lua framework API

1.1 log(level, ...)

说明： 打印函数
参数：

参数名	参数类型	必填(Y/N)	说明
level	number	Y	打印等级
...	any	N	可变参数

返回： nil

1.2 get_language_type()

说明： 获取当前语言类型
参数： nil
返回：

类型	说明
string	语言类型描述

1.3 set_language(lang)

说明： 设置语言
参数：

参数名	参数类型	必填(Y/N)	说明
lang	string	Y	语种

返回： nil

1.4 get_language_str(str)

说明： 根据指定字符串寻找当前语种对应的值。例如当前语种是EN，语言文件翻译关系为：测试 = test，则get_language_str("测试")，返回test
参数：

参数名	参数类型	必填(Y/N)	说明
str	string	Y	翻译文件的key

返回：

类型	说明
string	翻译文件的value

1.5 set_data(obj)

说明： 接收一个table对象，解析table中的key-value关系，当key是page.data中的成员时，page.data.key = value，且如果key关联了ctrl，则会自动更新相关的ctrl。
参数：

参数名	参数类型	必填(Y/N)	说明
obj	table	Y	新的键值关系表

返回： nil

1.6 set_page(page, msg)

说明： 更新界面
参数：

参数名	参数类型	必填(Y/N)	说明
page	string	Y	跳转界面的page（router中的key）
msg	any	N	携带到下一个界面的msg

返回： nil

1.7 get_app()

说明： 获取app对象
参数： nil
返回：

类型	说明
table	app对象

1.8 get_split(str, reps)

说明： 把一个str根据reps分片
参数：

参数名	参数类型	必填(Y/N)	说明
str	string	Y	需要分片的字符串
reps	string	Y	以reps做分割

返回：

类型	说明
table	分片后的集合

1.9 get_language_tab(tab)

说明： 根据一个tab的value，获取当前语种的映射表。内部是对get_language_str的封装
参数：

参数名	参数类型	必填(Y/N)	说明
tab	table	Y	文字集合

返回：

类型	说明
table	语言映射集合

1.10 set_background(image)

说明： 改变背景图片
参数：

参数名	参数类型	必填(Y/N)	说明
image	string	Y	图片文件

返回： nil

1.11 get_now_page_name()

说明： 获取当前页面名字
参数： nil
返回：

类型	说明

类型	说明
string	当前页面名字

1.12 get_recolor_str(str, recolor, part)

说明： 获取重新着色文本, 返回拼接后的文本
参数：

参数名	参数类型	必填(Y/N)	说明
str	string	Y	原始文本
recolor	number	Y	重新着色的颜色
part	string	N	不填写时着色全部文本，填写时只着色填写的文本。注：part必须存在于str中

返回：

类型	说明
string	返回拼接后的文本

1.13 common_add(name, msg)

说明： 添加common界面
参数：

参数名	参数类型	必填(Y/N)	说明
name	string	Y	common页面的名字
msg	table	N	携带到common界面的msg

返回：

1.14 common_destroy(name)

说明： 删除common界面
参数：

参数名	参数类型	必填(Y/N)	说明
name	string	Y	common页面的名字

返回：

1.15 common_hidden(name)

说明： 隐藏common界面
参数：

参数名	参数类型	必填(Y/N)	说明
name	string	Y	common页面的名字

返回：

1.16 common_show(name)

说明： 显示common界面
参数：

参数名	参数类型	必填(Y/N)	说明
name	string	Y	common页面的名字

返回：

1.17 common_check(name)

说明： 检查common界面是否存在
参数：

参数名	参数类型	必填(Y/N)	说明
name	string	Y	common页面的名字

返回：

类型	说明
boolean	返回common界面是否存在

1.18 lgui_screensaver_reset()

说明： 重新计时屏保时间
参数：
返回：

1.19 lgui_gesture_open()

说明： 启动手势导航
参数：
返回：

1.20 lgui_gesture_close()

说明： 关闭手势导航
参数：
返回：

1.21 lgui_gesture_set_color(color)

说明： 设置手势导航滑动时显示动画色块的颜色
参数：

参数名	参数类型	必填(Y/N)	说明
color	number	Y	设置的颜色

返回：

1.22 lgui_gesture_set_part(part)

说明： 设置手势控件可触发的动作
参数：

参数名	参数类型	必填(Y/N)	说明
part	number	Y	可触发的动作(可取值参考 utils/gesture.lua)

1.23 set_screensaver_time(times)

说明： 重新设置屏保时间
参数：

参数名	参数类型	必填(Y/N)	说明
times	number	Y	设置的时长. 单位: s

返回：

1.24 get_type_by_index(index)

说明： 通过控件的index值获取控件类型
参数：

参数名	参数类型	必填(Y/N)	说明
index	number	Y	控件的index值

返回：

类型	说明
string	返回控件的类型。如："btn", "img"

1.25 lgui_touch_report_enable(enable)

说明： 是否开启坐标点上报。开启时将产生 lgui_event_touch 事件。
参数：

参数名	参数类型	必填(Y/N)	说明
enable	boolean	Y	true:开启 false 关闭

返回：
无

说明: 当开启坐标点上报时，将产生lgui_event_touch事件
参数:

参数名	参数类型	说明
x	string	x坐标
y	string	y坐标
state	string	触摸状态

1.26 lgui_timer_create(info)

说明: 创建自定义 timer
参数:

参数名	参数类型	必填(Y/N)	说明
info	table	Y	timer参数
info.name	string	Y	timer的名字，为timer的唯一标识符, 不能为空
info.exec_cb	function	Y	timer的执行函数
info.duration	string	Y	timer的周期时长
info.exec_count	string	Y	timer执行一定次数自动销毁, 为 nil 时，timer需用户自行销毁

返回:

类型	说明
boolean	true:success false:fail

> 例如:

```
-- param table 包含创建 timer 是传入的info数据。创建时可在info内部自定义成员
local function onesec_timer(param)
   print(param.name, os.date())
end

lgui_timer_create({
     name = "1sec_test",
     exec_cb = onesec_timer,
     duration = 200,
})
```

1.27 lgui_timer_delete(name)

说明: 删除自定义 timer
参数:

参数名	参数类型	必填(Y/N)	说明
name	string	Y	timer的名字

返回：

类型	说明
boolean	true:success false:fail

1.28 lgui_timer_pause(name)

说明： 暂停自定义 timer
参数：

参数名	参数类型	必填(Y/N)	说明
name	string	Y	timer的名字

返回：

类型	说明
boolean	true:success false:fail

1.29 lgui_timer_reumse(name)

说明： 恢复自定义 timer
参数：

参数名	参数类型	必填(Y/N)	说明
name	string	Y	timer的名字

返回：

类型	说明
boolean	true:success false:fail

1.30 lgui_set_img_cache_max_size(max_size)

说明： 设置单张图片缓存的最大字节
参数：

参数名	参数类型	必填(Y/N)	说明
max_size	number	Y	单张图片缓存的最大字节

2、c library API <lgui3.h>

2.1 LGUI_C_API bool lgui_send2lua(lgui_event_s *event);

说明： 发送事件给lua framework
参数：

参数名	参数方向	说明
event	in	事件结构

返回：

类型	说明
bool	成功返回true，失败返回false

2.2 LGUI_C_API bool lgui_pairs_table(lua_State *L, lgui_table_map_s *c, int index);

说明： 迭代lua传递到C的table并根据映射表赋值给指定地址
参数：

参数名	参数方向	说明
L	in	lua状态机
c	in	lua table和C的映射表
index	in	table在L栈中的位置

返回：

类型	说明
bool	成功返回true，失败返回false

示例：

Lua

```lua
local test_result = {math = 60, physics = 85}
local test_student = {name = "zhangsan", age = 15, result = test_result}
test_c_func(test_student)
```

C

```
typedef struct
{
    int math;
    int physics;
} result;

typedef struct
{
    char *name;
    int age;
    result res;
} student;

int test_c_func(lua_State* L)
{
    student stu;
    // 对应lua中的test_result，因为test_result是table，所以单独做一个映射表
    lgui_table_map_s result_map[] =
    {
        // 第一个参数是Lua的数据类型
        // 第二个参数是Lua table中的key
        // 第三个参数若找到对应的key，value赋值给指定的地址
        {LUA_TNUMBER, (int)"math",    &stu.res.math},
        {LUA_TNUMBER, (int)"physics", &stu.res.physics},
        LGUI_TABLE_MAP_END,
    };
    // 对应lua中的test_student
    lgui_table_map_s student_map[] =
    {
        // 此处的stu.name指向的内容会跟随lua的垃圾回收
        // 若想持久化保存在C中，需自己另行拷贝
        {LUA_TSTRING, (int)"name",    &stu.name},
        {LUA_TNUMBER, (int)"age",     &stu.age},
        // table类型，嵌入另一个映射表
        {LUA_TTABLE,  (int)"result",  result_map},
        LGUI_TABLE_MAP_END,
    };
    // 第三个参数是因为lua只传递了一个table参数，所以该table在栈顶，即-1
    // lua调用C接口传递下来的参数，会顺序压栈，即第一个参数在栈底，最后一个参数在栈顶
    lgui_pairs_table(L, student_map, -1);
}
```

2.3 LGUI_C_API void lgui_push_table(lua_State *L, lgui_table_map_s *c);

说明： 根据映射表，把C的映射关系，打包成table传递给lua，即lgui_pairs_table的逆操作，打包后的table，在栈顶（-1位置）

参数：

参数名	参数方向	说明
L	in	lua状态机
c	in	lua table和C的映射表

返回： void

示例:

C

```c
typedef struct
{
    int     wifi_rssi;
    char    *wifi_name;
    bool    wifi_connect;
} push;

int test_push(lua_State *L)
{
  push p;

  p.wifi_name = "wifi1";
  p.wifi_connect = false;
  p.wifi_rssi = 80;

  // 结构体3个成员+1个结尾
  lgui_table_map_s *map = (lgui_table_map_s *)calloc(4 + 1, sizeof(lgui_table_map_s));

  map[0].key = (int)"rssi";
  map[0].value = (void*)&p.wifi_rssi;
  map[0].value_type = LUA_TNUMBER;

  map[1].key = (int)"name";
  map[1].value = (void*)p.wifi_name;
  map[1].value_type = LUA_TSTRING;

  map[2].key = (int)"connect";
  map[2].value = (void*)&p.wifi_connect;
  map[2].value_type = LUA_TBOOLEAN;

  lgui_table_map_s *map_t = (lgui_table_map_s *)calloc(1 + 1, sizeof(lgui_table_map_s));
  map_t[0].key = (int)"t1";
  map_t[0].value = (void*)"t1_value";
  map_t[0].value_type = LUA_TSTRING;
  map_t[1].value_type = LUA_TNONE;

  map[3].key = (int)NULL;
  map[3].value = (void*)map_t;
  map[3].value_type = LUA_TTABLE;

  map[4].value_type = LUA_TNONE;

  // lua table 结构如下
  // {{ t1 = "t1_value" } , name = "wifi1" , rssi = 80.0 , connect = false }

  lgui_push_table(L, map);

  free(map);

  return 1;
}
```

2.4 LGUI_C_API bool lgui_dispaly_deinit(void);

说明： 关闭LGUI3显示

2.5 LGUI_C_API bool lgui_dispaly_reinit(void);

说明： 重新打开LGUI3显示

八、控件说明

> lgui framework中，所有控件公共属性如下，控件详细描述中，只会列举不同的参数。
> 属性命缩写规则：
> c = color
> act = action
> w = width
> h = height
> src = source
> dest = destination
> clk = click
> dis = disable
> chk = check
> rel = release
> slt = select
> def = default
> ctrl = control
> dir = direction
> cnt = count
> rpt = report

属性	类型	默认值	必填(Y/N)	说明
type	string		Y	控件类型，bar/btn/img ...
position	table		Y	控件位置
position.x	number	0	N	绝对位置的X值
position.y	number	0	N	绝对位置的Y值
position.align	number		N	相对位置，参考utils/align.lua，若指定了该参数， 则绝对位置的x,y会被忽略
position.alignx	number	0	N	相对位置对齐后的X偏移
position.aligny	number	0	N	相对位置对齐后的Y偏移
position.ref	string		N	相对位置的参考控件，若不指定，则参考父控件， 若不存在父控件，则参考背景
name	string		N	控件名字，需在当前view中唯一， 其他控件用该名字关联参考控件或父控件
attr	table		Y	控件属性
attr.hidden	boolean	false	N	是否隐藏
attr.touchable	boolean	true	N	控件是否开启触摸事件, 默认开启
attr.dis_scroll_propagation	boolean	false	N	用于禁用当前控件的滑动传递功能

属性	类型	默认值	必填(Y/N)	说明
attr.parent	string		N	指定父控件

1、bar

说明： 加载进度线形控件，用户可以通过set_data改变控件展示的进度
参数：

属性	类型	默认值	必填(Y/N)	说明
attr.w	number		Y	进度条宽度
attr.h	number		Y	进度条高度
attr.src	number	0	N	进度条的起始值
attr.dest	number	100	N	进度条的结束值
attr.value	number		Y	进度条的当前值
attr.time	number	0	N	value产生变化时，动画渐进效果的持续时间
attr.c	number		Y	进度条背景颜色
attr.c_act	number		Y	进度条颜色
attr.round	boolean	false	N	是否圆角

示例：

> view

```
local view =
{
    {
        type = "bar",
        position = {x = 10, y = 20},
        attr = {w = 200, h = 10, value = "{{bar_value}}", animi = true, time = 200, color = 0xff00ff00, c
    }
}

return view
```

> controller

```
local controller =
{
    data = {bar_value = 10},
    onshow = function() end,
}

return controller
```

2、btn

说明： 图片按钮控件

属性	类型	默认值	必填(Y/N)	说明
attr.state	number	0	N	btn的状态，参考utils/btn.lua
attr.w	number	0	N	按键的固定宽,设置之后图片将居中显示,超出的部分将会被裁剪
attr.h	number	0	N	按键的固定高,设置之后图片将居中显示,超出的部分将会被裁剪
attr.chk	boolean	false	N	是否作为check btn，比如开关，或btn有选中状态， 需初始化state状态
attr.res_rel	string		Y	按钮释放时的图片. 作为check btn时， 非check状态下释放时的图片
attr.res_clk	string		N	按钮按下时的图片. 作为check btn时， 非check状态下按下时的图片
attr.res_long	string		N	按钮长按时的图片
attr.res_dis	string		N	按钮禁用时的图片. 作为check btn时， 非check状态下禁用时的图片
attr.res_chk_rel	string		N	作为check btn时，check状态下释放时的图片
attr.res_chk_clk	string		N	作为check btn时，check状态下按下时的图片
attr.res_chk_dis	string		N	作为check btn时，check状态下禁用时的图片
action	table		N	控件动作
action.bind	table		N	动作绑定列表
action.bind.up	string		N	释放时，绑定controller中的函数名
action.bind.down	string		N	按下时，绑定controller中的函数名
action.bind.long	string		N	长按时，绑定controller中的函数名

示例：

> view

```
local view =
{
    {
        type = "btn",
        position = {align = utils_align.IN_TOP_MID, aligny = 100},
        attr = {res_rel = "button-1.png", res_clk = "button-2png"},
    },
}

return view
```

3、btnm

说明： 按键矩阵控件，通常用来构造键盘

属性	类型	默认值	必填(Y/N)	说明
attr.w	number		Y	控件宽度
attr.h	number		Y	控件高度
attr.h_content	number		Y	文字高度
attr.c	number		Y	背景颜色
attr.c_def	number		Y	按键常态颜色
attr.c_clk	number		N	按键按下的颜色
attr.c_dis	number		N	按键禁用的颜色
attr.c_chk	number		N	按键选中颜色
attr.c_content	number		Y	文本默认颜色
attr.c_content_clk	number		N	文本按下时的颜色
attr.c_content_dis	number		N	文本禁用时的颜色
attr.c_content_chk	number		N	文本选中时的颜色
attr.radius	number		N	按键的圆角半径
attr.inner	number	8	N	按键之间的填充的大小
attr.pad_x	number	8	N	键盘左右填充的大小
attr.pad_y	number	8	N	键盘顶部与底部填充的大小
attr.map	table		Y	矩阵文本，以"\n"作为换行分隔，""空字符为结尾
attr.map_ctrl	table		Y	矩阵文本控制，参考utils/btnm.lua
action	table		N	控件动作
action.bind	table		N	动作绑定列表
action.bind.up	string		N	释放时，绑定controller中的函数名
action.bind.down	string		N	按下时，绑定controller中的函数名
action.bind.long	string		N	长按时，绑定controller中的函数名
action.bind.change	string		N	按键矩阵触发完成时，绑定controller中的函数名

示例：

> view

```
local view =
{
   {
        type = "btnm",
        position = {align = utils_align.IN_TOP_MID, alignx = 0, aligny = 150},
        attr =
        {
            w = lcd_w - 100, h = lcd_h - 150, h_content = 35,
            -- 背景，按键常态，按键按下，按键禁用，按键选中的颜色
            c = 0xffffffff, c_def = 0xffe0e0e0, c_clk = 0xffbfbfbf, c_dis = 0xff808080, >c_chk = 0xff0000
            -- 文本，文本按下，文本禁用，文本选中的颜色
            c_content = 0xffff0000, c_content_clk = 0xff000000, c_content_dis = >0xff000000, c_content_chk
            map =
            {
                "1","2","3","\n",
                "4","5","6","\n",
                "7","8","9","\n",
                "*","0","#",
            },
            map_ctrl =
            {
                -- 矩阵第一行属性：禁用，可选中，按键抬起报告+长按不报告
                1 | utils_btnm.DIS, 1 | utils_btnm.CHK, 1 | utils_btnm.RPT_UP | >utils_btnm.NO_RPT_LONG,
                -- 矩阵第二行属性：选中，长按不报告，正常
                1 | utils_btnm.SLT, 1 | utils_btnm.NO_RPT_LONG, 1,
                -- 矩阵第三行属性：正常
                1, 1, 1,
                -- 矩阵第四行属性：隐藏，正常，正常
                1 | utils_btnm.HIDDEN, 3, 1,
            },
        },
        action = {bind = {change = "btnm_action"}},
   }
}

return view
```

4、checkbox

说明： 复选框控件

属性	类型	默认值	必填(Y/N)	说明
attr.h	number		Y	字体高度
attr.dis	boolean	false	N	是否禁用复选框
attr.chk	boolean	false	N	选中状态
attr.round	boolean	false	N	是否为圆形
attr.c_def	number		N	未选中状态的颜色
attr.c_slt	number		N	选中状态的颜色
attr.c_clk	number		N	按下时的颜色
attr.c_dis	number		N	禁用时的颜色
attr.c_content	number		Y	文本颜色

属性	类型	默认值	必填(Y/N)	说明
attr.content	string		Y	文本内容
attr.res	string		Y	复选框选中时显示的图片路径
attr.res_rel	string		Y	复选框中未选中时显示的图片路径
action	table		N	控件动作
action.bind	table		N	动作绑定列表
action.bind.up	string		N	释放时，绑定controller中的函数名
action.bind.down	string		N	按下时，绑定controller中的函数名
action.bind.change	string		N	复选框状态切换时，绑定controller中的函数名

示例：

> view

```
local view =
{
   {
       type = "checkbox",
       position = {align = utils_align.IN_TOP_MID, alignx = 0, aligny = 150},
       attr =
       {
           h = 50, dis = false, chk = false, res = "yes.png",
           c_def = 0xff808080, c_slt = 0xffffffff, c_clk = 0xffff0000, c_dis = 0xffffffff,
           c_content = 0xff123456, content = "中国",
       },
   },
}

return view
```

5、clock

说明： 时钟控件

属性	类型	默认值	必填(Y/N)	说明
attr.w	number		Y	控件宽
attr.h	number		Y	控件高
attr.res_bg	number		N	表盘背景图片
attr.res_sec	number		N	秒钟图片
attr.res_min	number		N	分针图片
attr.res_hour	number		N	时针图片
attr.hour_offset_x	number		N	时针图片旋转中心x轴偏移大小

属性	类型	默认值	必填(Y/N)	说明
attr.hour_offset_y	number		N	时针图片旋转中心y轴偏移大小
attr.min_offset_x	number		N	分针图片旋转中心x轴偏移大小
attr.min_offset_y	number		N	分针图片旋转中心y轴偏移大小
attr.sec_offset_x	number		N	秒针图片旋转中心x轴偏移大小
attr.sec_offset_y	number		N	秒针图片旋转中心y轴偏移大小
attr.align	number	0	N	时间文字在控件范围中的位置，参考utils/clock.lua
attr.c	number		Y	时间文字的颜色
attr.fmt	string		Y	时间格式，遵循C strftime函数中的format规则

> 使用图片进行时钟绘画时，图片的旋转中心为图片的底部中心点。
> 当 fmt 属性与 res_xxx 一起设置时，显示 fmt 格式的数字时钟，不会显示图片时钟。两种不能共存。

示例：

> view

```
local view =
{
    {
        type = "checkbox",
        position = {align = utils_align.IN_TOP_MID, alignx = 0, aligny = 150},
        attr =
        {
            h = 50, dis = false, chk = false, res = "yes.png",
            c_def = 0xff808080, c_slt = 0xffffffff, c_clk = 0xffff0000, c_dis = 0xffffffff,
            c_content = 0xff123456, content = "中国",
        },
    },
}

return view
```

6、img

说明： 图片控件

属性	类型	默认值	必填(Y/N)	说明
attr.w	number	0	N	图片宽度，若该宽度不等于图片原始宽度，则会自动缩放，<=0时不缩放
attr.h	number	0	N	图片高度，若该高度不等于图片原始高度，则会自动缩放，<=0时不缩放
attr.offset_x	number	0	N	图片x轴上偏移之后再显示, >0:向右偏移 <0:向左偏移
attr.offset_x	number	0	N	图片y轴上偏移之后再显示, >0:向下偏移 <0:向上偏移

属性	类型	默认值	必填 (Y/N)	说明
attr.width	number	-1	N	图片实际显示的宽度,小于图片宽度, 则切割显示, 小于0时, 不切割
attr.height	number	-1	N	图片实际显示的高度,小于图片高度, 则切割显示，小于0时, 不切割
attr.res	string		Y	图片文件

示例：

> view

```
local view =
{
   {
       type = "img",
       position = {align = utils_align.IN_TOP_MID, alignx = -200, aligny = 150},
       attr = {w = 120, h = 100,res = "62.png"}
   },
}

return view
```

7、list

说明： 列表控件

属性	类型	默认值	必填 (Y/N)	说明
attr.w	number		Y	控件宽度
attr.h	number		Y	控件高度
attr.slidepos	number		Y	控件初始滑动位置
attr.mode	number		N	list 右边滚动条的模式
attr.radius	number		N	list 背景圆角半径
attr.radius_btn	number		N	list内部按键的圆角半径。
attr.select	number		N	list单选模式下,当前选中的选项.(取值从1开始)
attr.single	boolean		N	是否为单选模式 (true:单选 false:多选)
attr.h_line	number		Y	列表中，每一行的高度(滑动方位为纵向时生效, 生效时必填)
attr.w_line	number		Y	列表中，每一行的宽度(滑动方位为横向时生效, 生效时必填)
attr.dir	number	0	N	列表滑动的方向, 可设值参考 utils/list.lua 的定义
attr.c	number		N	列表背景颜色

属性	类型	默认值	必填(Y/N)	说明
attr.c_def	number		N	每行作为一个按键时的颜色
attr.c_clk	number		N	按下时的颜色
attr.c_chk	number		N	选中时的颜色
attr.c_bar	number		N	页滚动条颜色
attr.c_edge	number		N	当列表到达最高或最低位置时, 显示类似圆圈效果的颜色
attr.btn_fit	number		N	列表中按键的fit属性
attr.inner	number		N	列表中按键之间填充的大小
attr.chk	boolean	false	N	是否开启选中模式
attr.keep	boolean	false	N	是否锁定列表，当set_data更新列表成员时， 保存list当前位置不变
attr.map	table		Y	
attr.map_ctrl	table		Y	根据n决定list中一行有多少个成员，即n==列
attr.map_ctrl[n].type	number		N	参考utils/list.lua
attr.map_ctrl[n].x	number		N	第n列位置的x值
attr.map_ctrl[n].y	number		N	第n列位置的y值
attr.map_ctrl[n].w	number		N	第n列的宽度
attr.map_ctrl[n].h	number		N	第n列的高度
attr.map_ctrl[n].content_height	number		N	
attr.map_ctrl[n].content_algin	number		N	
attr.map_ctrl[n].content_mode	number		N	
attr.map_ctrl[n].color	number		N	
action	table		N	控件动作
action.bind	table		N	动作绑定列表
action.bind.up	string		N	释放时，绑定controller中的函数名
action.bind.down	string		N	控件按键按下时，绑定controller中的函数名
action.bind.long	string		N	控件按键长按时，绑定controller中的函数名
action.bind.change	string		N	控件滑动时，绑定controller中的函数名

示例：

view

```
local view =
{
    {
        type = "",
        position = {x = 0, y = 0},
    }
}

return view
```

8、loading

说明： 加载进度圆形控件

属性	类型	默认值	必填(Y/N)	说明
w	number		Y	控件宽度（外圆直径）
w_inner	number		Y	控件宽度（内圆直径）
src	number	0	Y	起始角度
dest	number	0	Y	结束角度
value	number	0	Y	当前角度
c	number		N	背景颜色
c_bg	number		N	进度条背景颜色
c_act	number		N	进度条填充颜色

示例：

view

```
local view =
{
   {
        type = "loading",
        position = {align = utils_align.IN_TOP_MID, alignx = 0, aligny = 150},
        attr = {w = 100, w_inner = 10, src = 0, dest = 360, value = 50, c = 0xff808080, c_bg = 0xff808080,
   }
}

return view
```

9、mark

说明： 蒙层控件

属性	类型	默认值	必填(Y/N)	说明
w	number		Y	蒙层宽度

属性	类型	默认值	必填(Y/N)	说明
h	number		Y	蒙层高度
c	number		Y	蒙层颜色
radius	number		N	圆角半径
keep	number	false	N	是否保持当前配置的位置，而不是显示的最上面
action	table		N	控件动作
action.bind	table		N	动作绑定列表
action.bind.up	string		N	释放时，绑定controller中的函数名
action.bind.down	string		N	按下时，绑定controller中的函数名
action.bind.long	string		N	长按时，绑定controller中的函数名

示例：

> view

```
local view =
{
    {
        type = "mark",
        position = {align = utils_align.IN_TOP_MID},
        attr = {w = get_lcd_hor_res(), h = get_lcd_ver_res(), c = 0x80808080},
        action = {bind = {up = "mark_action"}},
    },
}

return view
```

10、mbox

说明： 消息框控件

属性	类型	默认值	必填(Y/N)	说明
attr.w	number		Y	控件宽度
attr.h	number		Y	控件高度
attr.h_title	number		Y	消息框标题文字高度
attr.h_body	number		Y	消息框正文文字高度
attr.h_btn	number		N	消息框按键高度
attr.h_content	number		N	消息框按键文字高度
attr.c_bg	number		N	消息框背景颜色
attr.c_title	number		N	消息框标题文字颜色

属性	类型	默认值	必填(Y/N)	说明
attr.c_body	number		N	消息框正文文字颜色
attr.c_content	number		N	消息框按键文字颜色
attr.c_btn_clk	number		N	消息框按键按下时的颜色
attr.c_btn_def	number		N	消息框按键抬起时的颜色
attr.time	number		N	消息框定时关闭时间
attr.raduis	number		N	设置背景圆角大小
attr.title	string		N	消息框标题文本
attr.body	string		N	消息框正文文本
attr.content	table		N	消息框按键矩阵
action	table		N	控件动作
action.bind	table		N	动作绑定列表
action.bind.up	string		N	释放时，绑定controller中的函数名
action.bind.down	string		N	按下时，绑定controller中的函数名
action.bind.change	string		N	控件触发时，绑定controller中的函数名

示例：

view

```
local view =
{
   {

       type = "mbox",
       position = {align = utils_align.IN_TOP_MID, alignx = 0, aligny = 150},
       attr =
       {
           w = 400, h = 200, h_title = 20, h_body = 25, h_content = 30, h_btn = 60,
           c_bg = 0xffffffff, c_title = 0xff000000, c_body = 0xff000000,
           c_btn_clk = 0xff808080, c_btn_def = 0xffff0000, c_content = 0xff000000,
           title = "mbox title", body = "mbox body", time = 2000,
           content = {"OK", "CANCEL", },
       },
       action = {bind = {change = "mbox_action"}}
   },
}

return view
```

11、text

说明： 文本控件

属性	类型	默认值	必填(Y/N)	说明
attr.w	number		Y	控件宽度
attr.h	number		Y	文本高度
attr.height	number	-1	Y	控件高度,不设置或小于0时，控件会根据文本高度自动调节控件高度
attr.content	string		Y	文本内容
attr.mode	number		N	文本模式，滚动/裁剪等等，参考utils/text.lua
attr.align	number		N	该控件范围内的文本对齐方式，参考utils/text.lua
attr.c	number		N	文本颜色
attr.c_bg	number		N	控件的背景颜色
attr.radius	number		N	控件的背景圆角半径
attr.dir	number		N	文本方向，参考utils/text.lua

示例：

> view

```
local view =
{
   {
       type        = "text",
       position    = {align = utils_align.IN_TOP_MID, alignx = -250, aligny = 150},
       attr        = {content = "保持控件大小，超出部分显示省略号", w = 200, h = 30, color = 0xff00ff00, mode
   },
}

return view
```

12、roller

说明： 滚动选择器控件

属性	类型	默认值	必填(Y/N)	说明
attr.w	number		Y	控件宽度
attr.h	number		Y	控件高度
attr.content	string		N	文本集合，每行使用"\n"分隔
attr.c	number		N	背景颜色
attr.c_slt	number		N	选中行的背景颜色
attr.c_content	number		N	未选中行的文本颜色
attr.c_slt_content	number		N	选中行的文本颜色

属性	类型	默认值	必填(Y/N)	说明
attr.opt	number		N	选中行文字对应文本集合中的序号
attr.align	number		N	文本对齐方式，参考utils/roller.lua
attr.cnt	number		N	控件中显示文本的行数，当文本集合中的行数 > count时，滚动选择
attr.mode	number		N	控件模式，参考utils/roller.lua
action	table		N	控件动作
action.bind	table		N	动作绑定列表
action.bind.up	string		N	释放时，绑定controller中的函数名
action.bind.down	string		N	按下时，绑定controller中的函数名
action.bind.change	string		N	选项改变时，绑定controller中的函数名

示例：

view

```
local view =
{
    {
        type = "roller",
        position = {align = utils_align.IN_TOP_MID, alignx = 0, aligny = 150},
        attr = {
                w = 300, h = 20, c = 0x80808080, c_slt = 0xFFFFFF00, c_content = 0xFF000000, c_slt_content
                opt = 0, align = utils_roller.ALIGN_CENTER, cnt = 5, mode = utils_roller.MODE_NORMAL,
                content = "roller1\nroller2\nroller3\nroller4\nroller5\nroller6"},
        action = {bind = {change = function(v, value) log(3, "select index: " .. value) end}}
    },
}

return view
```

13、slider

说明： 滑块选择器控件

属性	类型	默认值	必填(Y/N)	说明
attr.w	number		Y	控件宽度
attr.h	number		Y	控件高度
attr.c	number		N	控件背景颜色
attr.c_act	number		N	已选择区域颜色
attr.c_knob	number		N	滑块旋钮颜色
attr.res_knob	number		N	将图片设置成滑块旋钮

属性	类型	默认值	必填(Y/N)	说明
attr.round	boolean	false	N	是否圆角
attr.spin	boolean	false	N	是否只能使用滑动，不响应点击事件
attr.min	number		N	滑块控件的最小值
attr.max	number		N	滑块控件的最大值
attr.value	number		N	滑块控件的当前值
attr.left	number		N	双按钮模式有效
attr.radius	number		N	圆角半径
attr.mode	number		N	滑块选择器模式，参考utils/slider.lua
action	table		N	控件动作
action.bind	table		N	动作绑定列表
action.bind.up	string		N	释放时，绑定controller中的函数名
action.bind.down	string		N	按下时，绑定controller中的函数名
action.bind.change	string		N	滑块数值改变时，绑定controller中的函数名

示例：

view

```
local view =
{
   {
        type = "slider",
        position = {align = utils_align.IN_TOP_MID, alignx = 0, aligny = 150},
        attr = {w = 300, h = 20, c = 0x80808080, c_act = 0xFFFFFF00, c_knob = 0xFFFFFFFF, round = true,
                spin = false, min = 0, max = 100, value = 50, mode = utils_slider.MODE_NORMAL},
        action = {bind = {change = function(v, value) log(3, "now number: " .. value) set_data({slider_te›
   },
}

return view
```

14、spinner

说明： 圆形旋转器控件

属性	类型	默认值	必填(Y/N)	说明
attr.w	number		Y	外圈直径
attr.w_inner	number		Y	内圈直径
attr.angle	number		Y	角度
attr.c	number		Y	颜色

属性	类型	默认值	必填(Y/N)	说明
attr.c_bg	number		Y	背景颜色
attr.time	number		Y	旋转周期
attr.dir	number	0	N	旋转方向，参考utils/spinner.lua
attr.mode	number		N	控件模式，参考utils/spinner.lua

示例：

> view

```
local view =
{
   {
       type = "spinner",
       position = {align = utils_align.IN_TOP_MID, alignx = 0, aligny = 150},
       attr = { w = 80, w_inner = 60, angle = 20, c = 0xFFFF00FF, time = 1000, c_bg = 0x80808080,
               dir = utils_spinner.DIR_FORWARD, mode = utils_spinner.MODE_FILLSPIN}
   },
}

return view
```

15、page

说明： 页控件

属性	类型	默认值	必填(Y/N)	说明
attr.w	number		Y	控件宽度
attr.h	number		Y	控件高度
attr.c	number		N	页背景颜色
attr.c_bar	number		N	页滚动条颜色
attr.c_edge	number		N	当列表到达最高或最低位置时,显示类似圆圈效果的颜色
attr.round	boolean		N	是否使用圆角
attr.radius	number		N	圆角半径
attr.slideposx	number		N	当前page滑动的位置x(一般为负数)
attr.slideposy	number		N	当前page滑动的位置y(一般为负数)
attr.inner	number		N	当使用layout时，控件与控件之间的间距
attr.flash	boolean	false	N	是否开启边缘动画
attr.layout	number		N	页布局，参考utils/page.lua
attr.mode	number		N	页模式，参考utils/page.lua

示例：

view

```
local view =
{
   {
       type        = "page",
       position    = {align = utils_align.IN_TOP_MID, alignx = 0, aligny = 150},
       attr        = {w = 600, h = 400, c = 0xFFFFFFFF, c_bar = 0x80808080, layout = utils_page.LAYOUT_CO
                      mode = utils_page.MODE_AUTO},
       name        = "page0"
   },
   {
       type = "img",
       position = {},
       attr = {res = "gugong.jpg", w = 400, h = 300, parent = "page0"}
   },
   {
       type        = "text",
       position    = {align = utils_align.IN_TOP_MID, alignx = 0, aligny = 540},
       attr        = {parent = "page0", w = 530, h = 30, color = 0xff00ff00, align = utils_text.ALIGN_LEF
                      content = "故宫又称紫禁城。中国古代讲究“天人合一”的规划理念，用天上的星辰与都城规划相对应，以突
   },
}

return view
```

16、textarea

说明： 文本输入器控件

属性	类型	默认值	必填(Y/N)	说明
attr.w	number		Y	控件宽度
attr.h	number		Y	控件高度
attr.h_content	number		Y	文本高度
attr.curor	number		Y	光标显示时间，0不显示光标
attr.pwd	number		N	大于0启用密码模式，该参数决定了明文显示时常， 超过该时间后，密码将被隐藏
attr.max	number		N	文本最大长度
attr.single	boolean	false	N	是否仅显示一行
attr.align	number		N	文本对齐方式，参考utils/textarea.lua
attr.c_content	number		N	文本颜色
attr.c_preset	number		N	默认文本content_blank的颜色
attr.c_bar	number		N	滚动条的颜色
attr.c_cursor	number		N	光标颜色

属性	类型	默认值	必填(Y/N)	说明
attr.c	number		N	背景颜色
attr.radius	number		N	背景圆角半径
attr.content	string		N	文本内容
attr.content_blank	string		N	文本内容为空时，显示的默认文本
attr.content_filter	string		N	过滤字符，非content_filter将被忽略
action	table		N	控件动作
action.bind	table		N	动作绑定列表
action.bind.up	string		N	释放时，绑定controller中的函数名
action.bind.down	string		N	按下时，绑定controller中的函数名
action.bind.change	string		N	文本改变时，绑定controller中的函数名

示例：

> view

```
local view =
{
   {
        type = "textarea",
        position = {align = utils_align.IN_TOP_MID, alignx = -200, aligny = 150},
        attr = {w = get_lcd_hor_res()/2, h_content = 30, curor = 1000, max = 6, align = 1, pwd = 500,
                c_content = 0xFF000000, c = 0xFFFFFFFF, content = "123456",c_preset = 0xFF808080,
                content_blank = "plase input password", single = true},
        action      = {bind = {change = "ta_change"}}
   },
}

return view
```

17、qrcode

说明： 二维码控件

属性	类型	默认值	必填(Y/N)	说明
attr.ver	number		Y	二维码的版本
attr.w	number		Y	二维码像素之间的距离
attr.eclevel	number		Y	二维码容错等级(0-3)
attr.content	string		Y	二维码中的内容

> 二维码图片的宽度为 ((ver - 1) * 4 + 21 + w * 2) * w

示例：

view

```
local view =
{
   {
        type = "qrcode",
        position = {align = utils_align.IN_TOP_MID, alignx = 0, aligny = 150},
        attr = {ver = 4, w = 5, content = "hello world"}
   }
}

return view
```

18、tab

说明： 选项卡控件

属性	类型	默认值	必填 (Y/N)	说明
attr.w	number		Y	控件宽度
attr.h	number		Y	控件高度
attr.c_bar	number		N	页滚动条颜色
attr.anim_time	number	100	N	tab控件切换到下个tab所用的动画时间(ms)
attr.treshold	number	attr.w/16	N	tab控件滑动时切换到下个界面的阈值 (指最少拖动treshold距离切换到下个tab)
attr.tab	number		N	当前显示的tab编号，从0开始
attr.mode	number		N	tab编号的显示位置，参考utils/tab.lua

示例：

view

```
local view =
{
  {
       type        = "tab",
       position    = {align = utils_align.IN_TOP_MID, alignx = 0, aligny = 150},
       attr        = {w = get_lcd_hor_res(), h = get_lcd_ver_res() - 150,},
       action      = {bind = {up = "tab_up", down = "tab_down", change = "tab_act"}},
       name        = "tab_bg"
   },
   {
       type        = "tab",
       position    = {},
       attr        = {parent = "tab_bg"},
       name        = "tab0"
   },
   {
       type        = "tab",
       position    = {},
       attr        = {parent = "tab_bg"},
       name        = "tab1"
   },
   {
       type        = "img",
       position    = {x = 100, y = 0},
       attr        = {res = "62.png", w = 200, h = 200, parent = "tab0"},
   },
   {
       type        = "img",
       position    = {x = 100, y = 0},
       attr        = {res = "62.png", w = 200, h = 200, parent = "tab1"},
   },
}

return view
```

19、vdec

说明： 视频解码控件

属性	类型	默认值	必填(Y/N)	说明
attr.w	number		Y	控件宽度
attr.h	number		Y	控件高度

示例：

view

```
local view =
{
   {
       type = "vdec",
       position = {align = utils_align.IN_TOP_MID, alignx = 0, aligny = 150},
       attr = {w = get_lcd_hor_res() / 2, h =get_lcd_ver_res() / 2}
   }
}

return view
```

20、blank

说明： 空白控件，通常用来做任意位置响应

属性	类型	默认值	必填 (Y/N)	说明
attr.w	number		Y	控件宽度
attr.h	number		Y	控件高度
attr.c_rel	number		Y	按钮释放时的颜色,作为check btn时， 非check状态下释放时的颜色
attr.c_clk	number		Y	按钮按下时的颜色. 作为check btn时， 非check状态下按下时的颜色
attr.c_dis	number		Y	按钮禁用时的颜色. 作为check btn时， 非check状态下禁用时的颜色
attr.c_chk_rel	number		Y	作为check btn时，check状态下释放时的颜色
attr.c_chk_clk	number		Y	作为check btn时，check状态下按下时的颜色
attr.c_chk_dis	number		Y	作为check btn时，check状态下禁用时的颜色
attr.radius	number		Y	圆角半径
attr.state	number		Y	btn的状态，参考utils/btn.lua
attr.chk	number		Y	是否作为check btn
attr.dis	boolean	false	N	是否禁用按键
action	table		N	控件动作
action.bind	table		N	动作绑定列表
action.bind.up	string		N	释放时，绑定controller中的函数名
action.bind.down	string		N	按下时，绑定controller中的函数名
action.bind.long	string		N	长按时，绑定controller中的函数名

注： w, h属性的特殊值说明

值	说明
-1	紧密包裹子控件,根据子控件的最大宽高显示
-2	根据父控件缩放大小,对齐父控件的边缘示
-3	先根据父控件缩放大小,对齐父控件的边缘, 如果子控件有对象超过则扩充大小

示例：

> view

```
local view =
{
   {
        type = "blank",
        position = {align = utils_align.IN_TOP_MID, alignx = 0, aligny = 220 + math.floor(get_lcd_ver_res(
        attr = {w = math.floor(get_lcd_hor_res() / 3), h =get_lcd_ver_res() / 3,},
        action = {bind = {up = function() log(3, "blank2 up") end}}
   },
}

return view
```

21、swiper

说明：

属性	类型	默认值	必填(Y/N)	说明
attr.w	number		Y	控件宽度
attr.h	number		Y	控件高度
attr.h_content	number		N	文本高度
attr.c	number		Y	文本颜色
attr.x_area	number		N	滚动区域x位置
attr.y_area	number		N	滚动区域y位置
attr.w_area	number		N	滚动区域宽
attr.h_area	number		N	滚动区域高
attr.type	number		N	控件类型
attr.mode	number		N	控件模式，参考utils/swiper.lua
attr.value	number		N	当前显示的编号
attr.cnt	number		N	显示个数
attr.slt	boolean	false	N	是否隐藏选中的选项
attr.map	table		Y	
action	table		N	控件动作
action.bind	table		N	动作绑定列表
action.bind.up	string		N	释放时，绑定controller中的函数名
action.bind.down	string		N	按下时，绑定controller中的函数名
action.bind.change	string		N	选项改变时，绑定controller中的函数名

示例：

```
view
```

```
local view =
{
    {
        type        = "swiper",
        position    = {align = utils_align.IN_TOP_MID, alignx = 0, aligny = 150},
        attr        = {w = 1024, h = 450, x_area = 0, y_area = 100, w_area = 1024, h_area = 250,
                       type = utils_swiper.TYPE_IMG, value = 0, cnt = 5, slt = false,
                       map = {
                            "setting/WiFi-2.png",
                            "setting/language-2.png",
                            "setting/date-2.png",
                            "setting/message-2.png",
                            "setting/system-2.png",
                            "setting/volue-2.png",
                            "setting/delall-2.png",
                            "setting/key-2.png",
                            "setting/color-2.png",
                            "setting/ring-2.png",
                        }},
    },
}

return view
```

22、shared

说明：

属性	类型	默认值	必填(Y/N)	说明
attr.file	number		Y	share 控件对应的lua文件名字
attr.func	number		Y	share lua文件中的函数名
attr.obj	number		N	对应函数的传参

示例：

view

```
local view =
{
    {
        type = "shared",
        attr = {file = "shared1", func = "back"},
    },
}

return view
```

23、chart

说明：

属性	类型	默认值	必填(Y/N)	说明
attr.w	number		Y	控件宽度

属性	类型	默认值	必填(Y/N)	说明
attr.h	number		Y	控件高度
attr.h_content	number		N	文本高度
attr.w_left	number		N	左边的空白宽度, 用于显示左边的刻度显示
attr.count	number		N	显示的点数
attr.x_options	number		N	x轴刻度显示模式
attr.y_options	number		N	y轴刻度显示模式
attr.grad	boolean		N	是否显示折线图下方的渐变色
attr.c_bg	number		N	背景颜色
attr.c_dotline	number		N	分割线的颜色
attr.c_content	number		N	文本颜色
attr.count	number		N	显示的点数
attr.y_min	number		N	y轴刻度的最小值
attr.y_max	number		N	y轴刻度的最大值
attr.mode	number		N	显示模式
attr.division	number		N	分割线的模式
attr.x_value	string		N	x轴刻度显示的文本
attr.y_value	string		N	x轴刻度显示的文本
attr.map	table		N	图表数据,第一个数据为颜色

示例：

```
view
```

```
local view =
{
   {
        type = "chart",
        position = {align = utils_align.IN_TOP_MID, alignx = 0, aligny = 150},
        attr = {w = 800, h = 400, h_content = 20, w_left = 50, count = 9, c_bg = 0x80808080, c_dotline = 0
                x_options = utils_chart.SKIP_LAST, x_value = "1\n2\n3\n4\n5\n6\n7\n8\n9",
                y_options = utils_chart.DRAW_LAST, y_min = 0, y_max = 100, y_value = "100\n90\n80\n70\n60\
                mode = utils_chart.MODE_LINE, division = utils_chart.DIV_ALL,
                map = {
                       {0xFF434343, 10, 20, 30, 40, 50, 60, 70, 80, 90},
                       {0xFFFFFF00, 90, 80, 70, 60, 50, 40, 30, 20, 10},
                       {0xFF00FFFF, 12, 85, 54, 65, 23, 43, 25, 85, 23}
                   }
            }
   }
}

return view
```

24、combobox

说明：

属性	类型	默认值	必填(Y/N)	说明
attr.w	number		Y	控件宽度
attr.h	number		Y	控件高度
attr.h_max	number		Y	选择列表的最大高度
attr.h_content	number		N	文本高度
attr.dir	number		N	选择列表的位置
attr.c	number		N	背景颜色
attr.c_slt	number		N	选中行的背景颜色
attr.c_bar	number		N	滚动条的颜色
attr.c_clk	number		N	按下时的颜色
attr.c_content	boolean		N	文本颜色
attr.opt	number		N	选中行文字对应文本集合中的序号
attr.c_slt_content	number		N	选中文本的颜色
attr.radius	number		N	背景的圆角半径
attr.content	string		N	文本集合，每行使用"\n"分隔
attr.res	string		N	另一侧的图片
action	table		N	控件动作
action.bind	table		N	动作绑定列表

属性	类型	默认值	必填(Y/N)	说明
action.bind.change	string		N	选项改变时，绑定controller中的函数名

示例：

> view

```
local view =
{
   {
       type = "combobox",
       position = {align = utils_align.IN_TOP_LEFT, alignx = 100, aligny = 150},
       attr = {w = 200, h = 50, h_max = 400, h_content = 30, c_content = 0xFFFFFFFF, dir = utils_combobo
           c = 0xFF808080, c_slt = 0xff00ff00,c_bar = 0x00000000, c_clk = 0xff0000ff, res = "down.png",
           content = "Apple\nBanana\nOrange\nMelon\nGrape\nRaspberry", opt = 3
       },
       action = {bind = {change = function(v, value, str) log(3, "select index: " .. value .." str:".. st
   },
}

return view
```

25、drawer

说明：

属性	类型	默认值	必填(Y/N)	说明
attr.act_h	number		N	触发区域的高度
attr.show_h	number		N	当前显示的高度
attr.dir	number		N	滑动方向, 可取值参考 utils/drawer.lua

> 注: 使用当前控件时, 屏幕上面30像素点的范围内不会触发其他控件

26、aimg

说明：

属性	类型	默认值	必填(Y/N)	说明
attr.times	number	-1	Y	循环次数, -1时无限循环
attr.interval	number		Y	切换图片的间隔
attr.map	table		Y	图片路径合集

示例：

> view

```
local = {
   {
       type = "aimg",
       position = {align = utils_align.CENTER},
       attr = {
           times = -1,
           interval = 100,
           map = {
               "aimg/aimg_1.png",
               "aimg/aimg_2.png",
               "aimg/aimg_3.png",
               "aimg/aimg_4.png",
               "aimg/aimg_5.png",
               "aimg/aimg_6.png",
               "aimg/aimg_7.png",
               "aimg/aimg_8.png",
           }
       },
   },
}

return view
```