

Super GP Library v3.0.0

-Widgets

Introduction:

V3.0.0 is the sixteenth version of SGL. Now there are ten widgets available. For further details of widgets, please refer to the widget instruction file. In 3.x versions, SGL inherits more windows utilities including menus, sounds, files and so on.

The main purpose of the SGL is to build an excellent graphic coding environment, wish all the users have a good coding time!

Upcoming:

Full widget kit is now coding.

SGWL(SGL for Web) and SGAL(SGL for Android) is now developing.

Instruction videos can be acquired from

<https://v.douyu.com/author/PDAPmLyG87xN> and

<http://space.bilibili.com/33137499/#!/index>

Widget declaration:

```
typedef struct _w{
    enum _control type;

    vecTwo pos;
    vecTwo size;

    int visible;
    int priority;
    int status;
    int style;

    int hide;
    int value;
    SGstring name;
    SGstring content;
    SGstring helpMessage;

    bitMap *cover;
    struct _w *associate;

    mouseMoveCall mouseIn, mouseOut;
    mouseClickCall mouseDown, mouseUp, mouseClick;
    keyCall keyDown, keyUp, keyPress;

}widgetObj;
```

`enum _control` type is the type of this widget. It will be initialized when `newWidget()` works.

`vecTwo` pos is the coordinate of the top left corner.

`vecTwo` size contains the width and height of this widget.

`int` visible is to control whether this widget is active or not. Programmer should set its value when initializing. Then we can use `showWidget()` and `ceaseWidget()` to change its value conveniently.

`int` priority is used by the system. Programmers should not change its value.

`int` status is to record whether it is passed or not , whether it is pressed or not, whether it is selected or and whether it is focused or not. Programmers should not change its value, too.

`int` style is the style of its appearance. Now SGL only supports `SG_DESIGN`.

`int` hide has different meanings in different type of widgets.

`int` value has different meanings in different type of widgets.

`SGstring` name is the name of this widget.

`SGstring` content is the string contains the message

of this widget. It has different meanings in different type of widgets, too.

`SGstring` `helpMessage` has no use now.

`bitMap` *cover points to a bitmap of the area covered by this widget.

`struct _w` *associate points to another widget which associate with this widget. Association between different types of widget has different meanings.

`mouseCalls` and `keyCalls` is the functions to execute when mouse moves or clicked and when key pressed.

Widget functions:

The function **`newWidget(int type, SGstring name)`** is to get a widget pointer with default values. Parameter `type` is one in enum `_control` defined in `winsgl.h` and `name` is the unique identifier of this widget.

The function **`registerWidget(widgetObj *obj)`** is to tell the system that `obj` has been fully prepared and should be shown on the screen. Then the program will copy the block which pointed by `obj`. Remember that the widget pointer in the system does not equal to the parameter `obj`.

The function **getWidgetByName(char *name)** returns the widget pointer with the identifier name. In this way we can change the attributes of each widget. Thus when registering, name should be unique.

The function **showWidget(char *name)** is to set the visibility to true.

The function **ceaseWidget(char *name)** is to set the visibility to false.

The function **deleteWidgetByName(char *name)** is to remove the widget with identifier name for ever.

For more callback function information, please refer to the instruction video.

Widget types:

SG_BUTTON is a button for users to click. It's a rectangle by default. When clicked, those `mouseClickCall` functions will execute. Programmers just need to write another function to respond. Remember to add `mouseClickDefault(w, x, y, status);` at the beginning of the new callback function.

SG_INPUT is an input line for users to input a string. It's a rectangle by default. The content of the struct is

the string that the user input.

`SG_DIALOG` is to show some message on the screen, like the `messageBox` in `WINAPI`. A small rectangular window will appear when `showWidget()` executes. The content of the struct is the string that shown in it. Programmer just need to decide when to call `showWidget()` is enough. When clicked the crossing icon, the widget will be ceased automatically.

`SG_OUTPUT` is a rectangle to show some message. Again the content is the string that it shows.

`SG_LIST` is a dropdown list for users to select one. The content is the names of the items separated by `'\0'`. So when initializing, we need to use `memcpy` not `strcpy`. The value of the struct is the index of the chosen item.

`SG_LABEL` is a one-line string. The content of the struct is what it showed.

`SG_CHECK` gets a boolean value whether the user

chose this item or not. The value of the struct shows if it is selected. The content of the struct will be shown on the right of the circle.

`SG_PROCESS` is a process bar. The value of it is the percent shown on the screen. Unlike `SG_DIALOG`, it won't be ceased automatically. The content of the struct is similar to the one in `SG_DIALOG`.

`SG_OPTION` is a list appears when click the right button of the mouse. If the mouse position is in its associate widget, the list will appear. Like `SG_LIST`, the names of the items should be separated by '\0' in the content of the struct.

`SG_DRAG` is a bar to adjust values. The max value is 100 so it can be also seen as percentage. Drag its button to change the value and get it through code.