

# Super GP Library v2.2.0

## -Instruction

### Introduction:

V2.2.0 is the fifteenth version of SGL. Now there are ten widgets available. For further details of widgets, please refer to the widget instruction file. Another improvement is the font. This version allows users to use all kinds of fonts that supported by Windows.

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>

## 1. Initialize

In Sample->Initialize->empty.c

The **sgSetup()** function is to initialize the system and malloc some pointers. It will run only once in the beginning of the program. Remember not to draw anything in this function, or it will cause a crash.

The **sgLoop()** function is the main loop. It will run infinitely which means while(1)sgLoop(); In the latest version, sgLoop() will definitely run one time per 10ms.

The **initWindow(int width, int height, char \*title, int mode)** function is to initialize the window with its width, height and title. If parameter mode is BIT\_MAP, we enter bit mode, or else if parameter mode is TEXT\_MAP, we enter text mode. After v2.0.0, the parameter width should be a integer which can be divide by 8, or it will cause some strange bugs.

In Sample->Initialize->colorful.c

Use **setColor(int r, int g, int b)** to set one RGB color. Then use **clearScreen()** to fill the whole screen with the color set before.

## 2. Draw

In Sample->Draw->pixel.c

The function **putPixel(int x, int y)** is to draw a point on the screen. The coordinate (0, 0) is the top-left point. And the coordinate(Screen->buffer->sizeX, Screen->buffer->sizeY) is the bottom-right point.

The function **getPixel(int x, int y)** is to get the color of the point (x, y). The return type RGB is defined in winsgl.h

In Sample->Draw->figure.c

The function **putQuad(int x1, int y1, int x2, int y2, int mode)** is to draw a rectangle with top-left point (x1, y1) and bottom-right point (x2, y2).

The function **putCircle(int x, int y, int r, int mode)** is to draw a circle with centre (x, y) and radius r.

The function **putEllipse(int xc, int yc, int a, int b, int mode)** is to draw an oval with centre (xc, yc) and semiaxis a on direction x, b on direction y.

The parameter mode in both function can be set as SOLID\_FILL or EMPTY\_FILL. SOLID\_FILL means to fill the whole figure with the set color while EMPTY\_FILL means just draw the outline.

The function **putLine(int x1, int y1, int x2, int y2)** is to draw a line between (x1, y1) and (x2, y2).

The function **floodFill(int x, int y, RGB c)** is to fill the area connected to (x, y) it will stop at the points with color c.

In Sample->Draw->bmp.c

The function **loadBmp(int x, int y, char \*filename)** is to load a bmp file named "filename" with its top-left at (x, y).

The function **getImage(int left, int top, int right, int bottom, bitMap \*bitmap)** is to copy an area of the screen. The points information is saved in bitmap. So it need to be declared and malloced by the users.

Remember left must be no greater than right and as well as top and bottom. After get and put the Image, bitmap->data must be freed, or it will cause memory leak.

The function **putImage(int left, int top, bitMap \*bitmap, int op)** is to paste the points saved in bitmap. The parameter op can be set as COPY\_PUT, AND\_PUT, OR\_PUT, XOR\_PUT and NOT\_PUT. To learn more about it, please read Sample->Advance->mouse.c.

In Sample->Draw->string.c

The function **putString(Cstring str, int x, int y)** is to put down a string with its left-top corner (x, y).

The function **putChar(char ch, int x, int y)** is similar to putString function, but it uses the old font library so the character put by this function may be a little ugly.

The function **putNumber(int n, int x, int y, char lr)** is to put down a number string. If lr is 'l', then (x, y) is the left-up point of this string. Else if lr is 'r', then (x, y) is the right-up point of this string.

In Sample->Draw->function.c

The function **funcMap(int x1, int x2, int y1, int y2, float(\*vect)(float x))** is to draw the curve of the given function "vect". [x1, x2] is its domain and [y1, y2] is its codomain. The base point is the left-top corner of the window.

In Sample->Draw->chinese.c

The function **putChinese((byte \*ch, int x, int y)** is to draw the chinese character stored in the first two byte of parameter "ch". If ch[0] or ch[1] is below 0xA1, which means that the first two byte do not represent a chinese character, nothing will happen.

To be easily used, the function **putString(char \*str, int x, int y)** is extended. Just put the chinese text in str and it will be printed.

### 3. Bios

In Sample->Bios->key.c

The function **initKey()** need to be added in the sgSetup() if other key functions will be used.

The function **bioskey(int cmd)** is the main key function. If cmd is 1, the return value is whether the key buffer is empty. Else if cmd is 0, the return value is the earliest key information. If the highest digit of the information is 1, it's a key-up signal. If the highest digit of the information is 0, it's a key-down signal. That is, when one key is pressed, a down-key will be send to the key buffer, and then a up-key will be send to the key buffer. The low 8 digit are the ascii of each key. Remember never use biosKey(0) separately, it must appear in if(biosKey(1)){} branch.

The function **clearKeyBuffer()** is used to clear the key buffer.

In Sample->Bios->key.c

Compile and run it. We can see the exact key number after pressing and releasing each key. When pressing the key, the key number is its ascii code. But when releasing the key, the key number is defined by Microsoft Windows. Remember to close the input method when running the program.

In Sample->Bios->mouse.c

The function **initMouse()** is similar to **initKey()**.

The function **mousePos()** returns the current mouse position.

The function **mouseStatus(int b)** is used to get each button's status. Parameter b can be set to **SG\_LEFT\_BUTTON** or **SG\_RIGHT\_BUTTON** or **SG\_MIDDLE\_BUTTON**. The return value is either **SG\_DOWN** or **SG\_UP**.

The function **biosMouse(int cmd)** is similar to **biosKey(int cmd)**. But its return value is **vectThree**. The first two int is the coordinate of the click, and the third int is which button is clicked. The same requests as **biosKey()** function.

The function **clearMouseBuffer()** is similar to **clearKeyBuffer()**.

## 4. Time

In Sample->Time->time.c

The function **delay(int t)** is to wait for a while.

Parameter t is the microseconds that will be waited.

Take notes that when in delay function, the picture on the screen won't change. That is, do not put delay funtions among drawing functions.

The function **delayBegin()** and **delayEnd(int t)** are used in pairs. The former set the start time and the latter set the end time. When the program run into delayEnd, if the time gap between delayBegin and delayEnd is less than t, the program will wait. Or else, don't wait.

The function **random(int n)** returns a number between 0 and n-1 randomly.

## 5. Interrupt

In Sample->Advance->vect.c

This part is to emulate DOS interrupts.

The function **getVect(int intrn)** returns the current interrupt function of intrn. In DOS, int8 represent clock interrupt and int9 represent keyboard interrupt. If intrn



is 8, it returns the current clock interrupt function. And if intrn is 9, it returns the current keyboard interrupt function.

The function **setVect(int intrn, vect v)** is used to set a new interrupt function to intrn.

The function **dosInt(int intrn, int \*ret)** is used to get the current key ascii when in keyboard interrupt function.

The function **setFreq(int f)** is used to set the frequency of clock interrupt. That is, every second the clock interrupt function will run f times.

## 6. Tools

In Sample->Advance->sound.c

The function **loadWave(char \*filename, int mode)** is used to play a wave file. Parameter mode need to be set as SG\_LOOP. In the latest version, several bugs appeared in this function, so do not use it until the next version come out.

In Sample->Advance ->mouse.c

The function **hideMouse()** is to hide the default mouse icon of Windows. In the latest version, this

function needs to be added in `sgLoop()`.

The function **`showMouse()`** is to show the default mouse icon of Windows. In the latest version, this function needs to be added in `sgLoop()`.

In Sample->Advance->full.c

The function **`fullScreen()`** is to change the window size to the whole screen. This function needs to appear in `sgSetup` function, or there will be no effects. In the latest version, several bugs appeared in this function. Remember not to run a program with this function, or some exciting things will happen.

The function **`setMouse(int x, int y)`** is to set the mouse position.

In Sample->Advance ->buffer.c

The function **`setActivePage(int page)`** is to set the active page. Parameter `page` can be either 0 or 1. Then all drawings will operate on this page.

The function **`setVisualPage(int page)`** is to set the VisualPage. Parameter `page` can be either 0 or 1. Then this page will be shown on the screen.

## 7.Text

In Sample->Write->hello.c

The function **setBfc(int bgc, int fgc)** is similar to **setColor**. In this function, **bgc** stands for background color, **fgc** stands for foreground color. These two parameters can be set from 0 to 15, each of which stands for one color in the enum **\_color** branch.

The function **clearText()** is similar to **clearScreen**. In this function, all chars on the screen will be set to **'\0'**, and their color is the one that set in **setBfc**.

The function **writeChar(char c, int x, int y)** is to put one character at position (x, y) with the color set in **setBfc**.

In Sample->Write->color.c

The function **writeString(char \*s, int x, int y)** is similar to **putString**. This time, the string can change its line automatically.

The function **setCharFgc(char color, int x, int y)** is to change the foreground color of the char in position (x, y). The parameter **color** can be set from 0 to 15, each of which stands for one color in the enum **\_color** branch.

The function **setCharBgc(char color, int x, int y)** is to change the background color of the char in position

(x, y). The parameter color can be set from 0 to 15, each of which stands for one color in the enum `_color` branch.

The function **`setCharColor(char color, int x, int y)`** is to change the background and foreground color of the char in position (x, y). The parameter color can be set from 0 to 255, the high 4 bit is the background color and the low 4 bit is the foreground color.

In Sample->Write->move.c

The function **`getText(int left, int top, int right, int bottom, textMap *text)`** is similar to `getImage`. And the notes are same.

The function **`putText(int left, int top, textMap *text)`** is similar to `putImage`. And the notes are same.