Super GP Library v3.0.2
-Instruction

Introduction:

V3.0.2 is the eighteenth version of SGL. V3.x focus on system level coding. This time SGL add multi-thread and system menu and tray supporting. For different system and platform, these functions are useful, too.

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

Upcoming:

Full widget kit is now coding. Besides, partial OpenGL is developing which means some versions later we may write OpenGL program and SGL program in the same window.

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 putSring 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 botton'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().

In Sample->Bios->file.c

The function **selectFile(char name[], char start[], char format[])** is to use windows frame to choose a file. However, it just give the file name that user selected and put it in name[]. Parameter start is where to start search and format is the accepted file format. For example, format[] can be "All files\0*.*\0C/C++ file\0*.h;*.c;*.cpp\0\0". Note * can be any string so use it to represent file name and separate type name and type format array.

The function **selectSave (char name[], char start[], char format[])** is to use windows frame to save a file. Remember that it just put the input file name into parameter name[] other than save any files. Other parameters are similar to those in selectFile.

The function **selectDir (char name[], char start[])** is to use windows frame to select a directory. It puts the selected directory into parameter name[].

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 intn)** returns the current interrupt function of intn. In DOS, int8 represent clock interrupt and int9 represent keyboard interrupt. If intn is 8, it returns the current clock interrupt function. And if intn is 9, it returns the current keyboard interrupt function.

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

The function **dosInt(int intn, 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 frequence of clock interrupt. That is, every second the clock interrupt function will run f times.

6.Tools

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->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.

In Sample->Advance->clipboard.c

The function **copyText (const char *str)** is to copy str into system clipboard.

The function **pasteText ()** returns the content of the system clipboard. System clipboard is global which means it can transfer string between programs.

In Sample->Advance->clipboard.c

The function **setMouse (int x, int y)** is set the cursor position directly.

The function **getWidth (int obj)** is to get the width of the given obj. The parameter can be SG_WINDOW or SG_SCREEN while the size of screen never changes but the size of window can be changed by user.

In Sample->Advance->menu.c

The function **alertInfo(const char *info, const char *title, int mode)** is to show some message in a new small window. Parameter title is the title for this new small window and info is its content. The value of mode can be one of those in enum _alert.

The function **initMenu()** is to initialize the main menu list in the top of the window. The return value is the id

of main menu list.

The function **addMenuItem(const char *title, int id, void(*func)())** is to add one item to the parent menu list with the given id. What will happen after the item is clicked is in the given function pointer func.

The function **addMenuList(const char *title, int id)** is to add one menu list to the parent menu list with the given id. The return value is the new list id.

The function **addMenuSeparator(int id)** is to add a separator in the menu list with id.

The function **addTray()** is to add an icon in the right-bottom of screen.

The function **hideToTray()** is to minimize the window and hide it to tray. If we click the icon in the tray, the window will appear again.

The function **restroeFromTray()** is the inverse steps of hide. It shows the window again.

The function **initTrayMenu ()** to initialize the tray menu list which will appear after right click the tray icon. The return value is the tray menu id.

The function **addTrayMenuItem(const char *title, int id, void(*func)())** is similar to addMenuItem.

The function **addTrayMenuList(const char *title, int id)** is similar to addMenuList.

The function **int addTrayMenuSeparator(int id)** is similar to addMenuSeparator.

In Sample->Advance->thread.c

The function **createThread (vect func)** is to execute the func in a new thread. The new thread and the main thread execute parallelly.

In Sample->Advance->timer.c

The function **timerThread(vect func, int millis, int time)** is to set a timer which execute func every millis of millisecond. After the given times, it stops automatically.

7.Internet

In Sample->Socket->client.c and server.c

The function **createServer(int port)** is to create a server with the given port. Different program on a single computer should have different ports. The return value is the server socket.

The function **createClient(const char *server, int port)** is to create a client and connect to server:port. For default, localhost is 127.0.0.1. The return value is

the client server.

The function **acceptOne (SOCKET server)** is for server to wait for an connection. The return value is the connection socket. So far, there are three different sockets.

The function **socketSend(SOCKET s, char *buffer)** is to send data via a connection socked.

The function **socketReceive(SOCKET s, char *buffer, int len)** is to wait for the sending. The data received is put into buffer with max length len. If the connection is broken, the return value is SG_CONNECTION_FAILED, or else SG_NO_ERORR.

The function **closeSocket(SOCKET s)** is to close one connection socket after one connection failure or a compulsory close.

8.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.