Document of the unnamed gui-lib

关于这个混搭库的一个说明

目录

关于这个图形库

这个库是这个图形库系统的第一部分，graphics 库，基于《c 语言的科学和艺术》中提供的图形库，以及翁凯老师在中国大学 mooc c 语言程序设计进阶课程中提供的 ACLlib 库（github 地址：https://github.com/wengkai/ACLLib/），同时参考了刘新国老师对于书中图形库的改进版本进行编写，使用 win32 api 实现。其中，本图形库的主要绘图逻辑使用 acllib 中的绘图函数并稍作改动，并在此基础上实现了大部分书中图形库的其他 api 接口（如字体属性、大小，绘图信息保存、获取与重载，窗口信息设定与获取等等），应该能比较好地兼容

了两个库的特性；同时添加和封装了一些其他功能，如更简单的文字绘制函数、声音函数，菜单创建函数等等。

如果您是 graphic 的用户，值得注意的是，在坐标系的选择上，本库的实现最后还是采用了屏幕坐标系而非自然坐标系，即以左上角为原点，向下为 y 轴正方向，向右为 x 轴正方向。

另外，第一部分的库可以在不使用第二部分的库的基础上独立使用。

原先《c 语言的科学和艺术》中提供的图形库一方面年代相对来说很落后，api 的实现方式很老旧，也对于低效能的硬件做了太多不必要的兼容；同时可能由于硬件和操作系统的瓶颈原因，或者教学需要 api 使 api 相对简单，使用擦除的方式进行重新绘制，但这样在频繁进行图像绘制时不仅会使整体速度下降，而且也较难以实现图形不带任何副作用的移动和删除。因此，在了解了一点 win32api 的基础知识，并结合手头其他几个教学用的简单图形库实现后，我结合几个库的特点重新写了一个图形库系统。希望这个库能做到，和原先《c 语言的科学和艺术》中提供的图形库相比，在底层方面更加贴近原生 api 的实现方式，牺牲一部分的封装和对老硬件的兼容性，带来更丰富的 api 接口和更快的运行速度；另一方面，它也能为上层图形库进一步面向对象的抽象和封装提供良好的底层实现。

自定义数据类型：

基本数据类型：

布尔值：以 int 定义

```
typedef enum { FALSE, TRUE } bool;
```

颜色：采用 RGB 三原色方式，有相关内置宏定义颜色，也可以用 createColor 函数生成

```
typedef COLORREF Color;
```

字符串类型：

```
typedef char* string;
```

图像信息结构体：（小写为结构体，大写指针）

```
typedef struct
{
    HBITMAP hbitmap;
    int width;
    int height;
} Image,*IMAGE;
```

枚举类型:

枚举类型为画笔或画刷的形态（其中前者用来画线条，后者用来填充区域），以及事件类型。

画笔类型:

```
typedef enum
{
    PEN_STYLE_SOLID,
    PEN_STYLE_DASH,            /* -------  */
    PEN_STYLE_DOT,             /* .......  */
    PEN_STYLE_DASHDOT,         /* _._._._  */
    PEN_STYLE_DASHDOTDOT, /* _.._.._  */
    PEN_STYLE_NULL
} ACL_Pen_Style;
```

画刷类型:

```
typedef enum
{
    BRUSH_STYLE_SOLID = -1,
    BRUSH_STYLE_HORIZONTAL,        /* ----- */
    BRUSH_STYLE_VERTICAL,      /* ||||| */
    BRUSH_STYLE_FDIAGONAL,     /* \\\\\ */
    BRUSH_STYLE_BDIAGONAL,     /* ///// */
    BRUSH_STYLE_CROSS,             /* +++++ */
    BRUSH_STYLE_DIAGCROSS,     /* xxxxx */
    BRUSH_STYLE_NULL
} ACL_Brush_Style;
```

鼠标按键类型:

```
typedef enum
{
    NO_BUTTON = 0,
    LEFT_BUTTON,
    MIDDLE_BUTTON,
    RIGHT_BUTTON
} ACL_Mouse_Button;
```

鼠标事件类型:

```
typedef enum
{
    BUTTON_DOWN,
    BUTTON_DOUBLECLICK,
    BUTTON_UP,
```

```
    ROLL_UP,
    ROLL_DOWN,
    MOUSEMOVE
} ACL_Mouse_Event;
```

键盘事件类型:

```
typedef enum
{
    KEY_DOWN,
    KEY_UP
} ACL_Keyboard_Event;
```

回调函数类型:

键盘回调函数

```
typedef void(*KeyboardEventCallback) (int key, int event);
```

第一个参数为按键的虚拟键码（注意：不是 ASCII 码）。 第二个参数为发生的事件（按下或弹起。参见 acllib.h 中 ACL_Key_Event。

虚拟键码，参见：

http://msdn.microsoft.com/en-us/library/windows/desktop/dd375731%28v=vs.85%29.aspx。

或

WinUser.h

字符回调函数

```
typedef void(*CharEventCallback) (char c);
```

按键产生文字输入时，出发此事件，参数为字符 ASCII 码，或对应中文字符编码

鼠标回调函数

```
typedef void(*MouseEventCallback) (int x, int y, int button, int event);
```

前两个参数为鼠标指针的位置， button 为按键， event 为发生的事件（按下或弹起），鼠标移动时产生 move 事件。参见 acllib.h 中 ACL_Mouse_Button、 ACL_Mouse_Event。

定时器回调函数

```
typedef void(*TimerEventCallback) (int timerID);
```

参数为定时器序号。

菜单选择回调函数

```
typedef void(*MenuSelectCallback)();
```

没参数。

窗口与主函数：

主函数：

主函数，程序运行开始时被调用的函数，需要用户自己实现。

```
int main(void);
```

窗口

启动图形窗口：

```
void initWindow(const char *wndName, int x, int y, int width, int height);
```

说明：

初始化程序窗口。该函数必须在 Setup 函数中最先调用，并且只能够调用一次。

参数：

wndName：窗口标题

x：窗口左上角横坐标，若不希望指定窗口位置， 可传入 DEFAULT。

y：窗口左上角纵坐标， 若不希望指定窗口位置， 可传入 DEFAULT。

width ：窗口可绘制区域的宽度。

height ：窗口可绘制区域的高度。

改变窗口大小、位置：

```
void resetWindowSize(int x, int y, int width, int height);
```

说明：

该函数可以在任意时候被调用，无返回值。

参数

x: 窗口左上角横坐标.

y: 窗口左上角纵坐标.

width : 窗口可绘制区域的宽度。

height : 窗口可绘制区域的高度。

```
int getWindowWidth();
int getWindowHeight();
```

说明：

返回窗口绘图区域的宽/高。

```
double GetXResolution(void);
double GetYResolution(void);
```

说明：

返回在 x 轴或 y 轴方向上一英寸的像素值数量。

```
//screen size
```

```
int GetFullScreenWidth(void);
int GetFullScreenHeight(void);
```
说明:
返回整个屏幕大小的像素值。

```
void initConsole(void);
```
说明:
初始化终端窗口。 在该函数执行后，才能使用 printf 和 scanf。

菜单:

```
void createMyMenu();
```
说明:
该函数无返回值，在插入菜单前需要先调用该函数。

```
bool insertMenuByName(string name, MenuSelectCallback func);
```
说明:
该函数用以创建一个一级弹出菜单或二级菜单，返回值标识是否成功创建。
参数:
string name 菜单名称；可以使用“菜单 1/选项 1”的格式创建二级选项菜单，也可以只创建一级菜单，如“菜单/”. 菜单名中的空格会被忽略；在调用之前需要先调用创建菜单函数。菜单的顺序和创建顺序相关。
MenuSelectCallback func: 指向菜单选项函数的回调指针。

绘图:

开始/结束绘制

```
void beginPaint();
void endPaint();
```
说明:
beginPaint()函数负责初始化绘图操作。 3.2 - 3.6 中，所有实际绘制图形的函数，均应在调用 beginPaint()函数后调用。绘制结束后，调用 endPaint()函数后，结束绘图操作， 并且将刚才绘制的图形显示到屏幕上。
beginPaint() 与 endPaint() 必须成 对使 用， 每 一个 beginPaint 调用 均应 当对应 一 个 endPaint 调用。建议在同一函数中完成一组绘图操作，以防止函数调用不匹配。若未成对使用这一组函数， 如: 未首先调用 beginPaint， 便直接调用 endPaint 函数; 或者第一次调用 beginPaint 后， 未调用 endPaint， 又再次调用 beginPaint 函数。 这些情况下程序在运行时会弹出窗口提示错误。绘图区域左上角为原点， 向右为 x 轴正方向，向下为 y 轴正方向。

几何形状绘制

线条

在绘图区域中，有一个初始位置为(0,0)的绘图点。该点与线条绘制相关。

```
int GetCurrentX(void);
int GetCurrentY(void);
```

获取绘图点的 x、 y 坐标。

```
void moveTo(int x, int y);
void moveRel(int dx, int dy);
```

移动绘图点。 moveTo 将绘制点移动到屏幕坐标的对应位置， moveReal 将绘制点作(dx,dy)的相对位移。

```
void arc(int nLeftRect, int nTopRect, int nRightRect, int nBottomRect, \
int nXStartArc, int nYStartArc, int nXEndArc, int nYEndArc);
```

说明:
绘制一段圆（椭圆）弧。
前 4 个参数给出一个矩形的左上角、 右下角定点。 绘制的弧线内切该矩形。 后两个参数指定两个点，这两个点到矩形中心的连线作为圆弧起始、终止位置的线。该函数的参数与 chrod、 pie 相同，建议实际测试了解其使用方法。

```
void line(int x0, int y0, int x1, int y1);
void lineTo(int nXEnd, int nYEnd);
void lineRel(int dx, int dy);
```

说明:
绘制直线。
line 函数直接绘制一条从(x0,y0)到(x1,y1)的直线，与绘制点的位置无关。
lineTo 从当前绘图点位置到屏幕坐标(x,y)绘制一条直线。 lineReal 函数从当前绘图点位置到相对位移(dx,dy)的位置绘制一条直线。

```
void polyBezier(const POINT *lppt, int cPoints);
```

说明:
绘制一条贝塞尔曲线。 贝塞尔曲线的每个节点有两个控制点（两个端点各有一个控制点）。若需要绘制一条 n 个节点的贝塞尔曲线， 需要传入 n*3-2 个点。 数组中点的顺序应为:起点、 起点控制点、 第 2 个点的第一个控制点、 第二个节点、 第 2 个点的第二个控制点、 ……、 终点的控制点、 终点。

```
void polyLine(const POINT *lppt, int cPoints);
```

说明:
顺次连接传入的数组中所有的点。

图形

```
void chrod(int nLeftRect, int nTopRect, int nRightRect, int nBottomRect, \
        int nXRadial1, int nYRadial1, int nXRadial2, int nYRadial2);
```

说明:
绘制一块弓形，参数与 Arc 相同，建议实践测试效果。

```
void ellipse(int nLeftRect, int nTopRect, int nRightRect, int nBottomRect);
```

说明:
绘制一个椭圆，参数为外切椭圆的矩形的左上角、 右下角坐标。

```
void pie(int nLeftRect, int nTopRect, int nRightRect, int nBottomRect, \
        int nXRadial1, int nYRadial1, int nXRadial2, int nYRadial2);
```

说明:
绘制一块扇形，参数与 Arc 相同， 建议实践测试效果。

```
void polygon(const POINT *lpPoints, int nCount);
```

说明:
将输入数组中的所有点顺序连接， 绘制一个多边形。

```
void rectangle(int nLeftRect, int nTopRect, int nRightRect, int nBottomRect);
```

说明:
绘制一个矩形，参数为矩形左上角、 右下角坐标。

```
void roundrect(int nLeftRect, int nTopRect, int nRightRect, int nBottomRect, \
        int nWidth, int nHeight);
```

说明:
绘制一个圆角矩形。最后两个参数为圆角部分的宽、高

画笔

画笔对应所有图形边框的绘制。
画笔颜色可以使用头文件中预定的颜色，也可以使用 RGB(r,g,b)宏或 createColor 函数自行设定颜色，每种颜色分量的范围为 0-255。 setPenColor 可以接受 EMPTY 作为参数， 将线条颜色设置为透明。
参见 acllib.h 中 ACL_Pen_Style。

设置当前画笔格式。

```
    void setPenColor(Color color);
    void setPenSize(int width);
    void setPenStyle(ACL_Pen_Style style);
```

获取当前画笔格式。

```
int GetPenSize(void);
Color GetPenColor(void);
```

画刷

画刷对应所有图形填充颜色的绘制。 画刷颜色也可以使用 setBrushColor(EMPTY)设置为透明。 参见 acllib.h 中 ACL_Brush_Style。

设置画刷格式

```
void setBrushColor(Color color);
void setBrushStyle(ACL_Brush_Style style);
```

获取当前画刷格式:

```
Color GetBrushColor(void);
ACL_Brush_Style GetBrushStyle(void);
```

文字

设置文字的前景颜色、背景颜色、尺寸、字体。

```
void setTextColor(Color color);
void setTextBkColor(Color color);
void setTextSize(int size);
void setTextFont(const char *pFontName);
```

获取文字的前景颜色、背景颜色、尺寸。

```
Color getTextColor();
Color GetTextBkColor();
int GetTextSize();
```

获取字符串所占矩形位置的高度、宽度。

```
int getTextWidth(string text);
int getTextHeight(string text);
```

```
void DisplayText(int x, int y, string textstring);
```

说明:

将 pStr 所指向的文字输出到屏幕坐标 x、 y 处。注意，此函数不会移动画笔。

```
void DrawTextString(string text);
```

说明:

从画笔当前位置出发，绘制字符串；绘制的过程中移动画笔。

```
void drawTextBox(int nLeftRect, int nTopRect, int nRightRect, int nBottomRect,
string text, unsigned int format);
```

说明:
在一个矩形中绘制字符串。
参数:
int nLeftRect, int nTopRect, int nRightRect, int nBottomRect,：矩形的四个顶点，和上文相同;
string text：要输出的字符串;
unsigned int format：指定输出的格式：标签如下:
DT_CENTER：指定文本水平居中显示。
DT_VCENTER：指定文本垂直居中显示。该标记仅仅在单行文本输出时有效，所以它必须与 DT_SINGLELINE 结合使用。
DT_SINGLELINE：单行显示文本，回车和换行符都不断行。

光标

```
void setCaretSize(int w, int h);
void setCaretPos(int x, int y);
void showCaret();
void hideCaret();
```

说明:
这里的光标指的是输入文字时，闪烁指示当前输入位置的光标。 输入文字时，光标并不会自动移动，需要手动调整光标位置。 光标默认位置在窗口右下角，输入法的输入窗口会出现在光标所在位置。

图片:

```
void loadImage(string pImageFileName, IMAGE pImage);
```

说明:
从文件中加载文件，仅支持 BMP、 JPEG、 GIF 格式。 第一个参数为文件名，第二个参数为指向一个 Image 结构的指针。

```
void freeImage(IMAGE pImage);
```

说明:
释放文件指针所在位置。

```
void putImage(IMAGE pImage, int x, int y);
```

将图像绘制到点(x,y)处。

```
void putImageScale(IMAGE pImage, int x, int y, int width, int height);
```

将图像绘制到点(x,y)处， 并且改变图像的宽、 高。

```
void putImageTransparent(IMAGE pImage, int x, int y, int width, int height, Color bkColor);
```

将图像绘制到点(x,y)处， 可以改变图像的宽、 高， 最后一个参数中给出的颜色将被过滤为透明色

绘图状态的保存和恢复

```
int SaveGraphicsState();
void RestoreGraphicsState(int id);
```

这两函数保存下来全局的绘图状态参数，如画笔位置、大小、颜色、笔刷颜色、类型、字符大小、颜色等等。全局绘图状态参数会被保存在栈中，栈的容量为 10（宏定义可修改）。

保存绘图状态的时候会返回一个绘图状态在栈中的位置， 当恢复状态时可使用参数 id 在多种状态中选择。如果传入的 id 参数为-1，会自动恢复上一次保存的状态。所有状态恢复后即从栈中弹出。

声音

直接播放声音:

提供了一个很简单的 api 函数，只要输入文件名就能直接播放声音。加载可能稍慢，但播放为异步过程，不影响程序进行。

```
void Sound(const char * SoundName);
```

加载声音、播放和删除

这些是 acllib 库函数的原有 api。

```
int loadSound(const char *fileName, int *pSound);
```

从文件中加载声音文件，第一个参数为文件名，第二个参数为指向一个 ACL_Sound 结构的指针。此处传入指针是为了获取声音文件所在地址，可以同样使用返回值获取。

```
void playSound(int sid, int repeat);
```

播放声音， 第二个参数非零时， 音乐将循环播放。否则只播放一遍。 多次调用该函数，可以同时播放多个声音。

```
void stopSound(int sid);
```

停止播放给定的声音。

事件回调函数

```
void registerKeyboardEvent(KeyboardEventCallback callback);
void registerCharEvent(CharEventCallback callback);
void registerMouseEvent(MouseEventCallback callback);
void registerTimerEvent(TimerEventCallback callback);
```

注册鼠标键盘回调函数。

```
void cancelKeyboardEvent();
void cancelCharEvent();
void cancelMouseEvent();
void cancelTimerEvent();
```

取消回调函数。

```
void startTimer(int timerID, int timeinterval);
void cancelTimer(int timerID);
```

定时器时间间隔单位为毫秒。设置定时器后，每经过 timeinterval 毫秒时间，便会触发 timer 事件。可以使用多个 timer，ID 建议使用从 0 开始的整数。当 timeinterval 设置为 0 时，定时器会在程序空闲时（没有键盘、鼠标输入，且上次 timer 事件已经处理）触发。timer 的时间精度在 10ms 数量级，当设置的时间间隔小于 10ms 时，实际间隔会在 10ms 以上

其他

消息框

```
int msgBox(const char title[], const char text[], int flag);
```

说明:

打开一个对话框，返回值为点选按钮的类型。

参数:

const char title[]：标题

const char text[]：对话框的内容

int flag 标识符:

MB_OK 默认值。有一个确认按钮在里面。

MB_YESNO 有是和否在里面。

MB_ABORTRETRYIGNORE 有 Abort（放弃），Retry（重试）和 Ignore（跳过）

MB_YESNOCANCEL 消息框含有三个按钮：Yes，No 和 Cancel

MB_RETRYCANCEL 有 Retry（重试）和 Cancel（取消）

MB_OKCANCEL 消息框含有两个按钮：OK 和 Cancel

返回值:

IDOK（1）	OK
IDCANCEL（2）	CANCEL
IDABORT（3）	ABORT
IDRETRY（4）	RETRY
IDIGNORE（5）	IGNORE
IDYES（6）	YES
IDNO（7）	NO

获取颜色:

```
//create color: from 0-255
    Color createColor(int red, int green, int blue);
```

传入参数为 RGB 值，范围为 0-255

暂停和退出

```
void Pause(double seconds);
```

暂停绘图多少秒钟。

void ExitGraphics(void);
退出程序。