C++从零开始编游戏

Display单元库教程

制作：ax_pokl

日期：2017-12-07

目录

前言

- 献给所有热爱游戏编程的程序员们。
- 学习Pascal语言的同学可以阅读本教程的原始版《Free Pascal从零开始编游戏》教程。
- 本教程使用Windows NT为内核的操作系统。
- 本教程的实例均通过GCC 5.3.0编译。
- 本教程使用Pascal语言Display单元库编译的的disp.dll动态链接库实现功能。类似于ege库，编译该库时需要disp.h头文件。

前言

- 本教程使用Display单元库，请先阅读头文件disp.h内所有文字说明和所有子程序重载
- 本教程所有实例永久下载地址：

http://axpokl.com/display.zip

http://axpokl.ys168.com/

- 作者ax_pokl联系方式：

E-mail：ax_pokl@sina.com，QQ：395838203。

- 由于作者水平有限，教程难免有错误和疏漏之处，敬请谅解。发现错误请联系作者，谢谢！

第一章 配置

- 第一节 Windows操作系统
- 第二节 GUN C语言编译器
- 第三节 Display单元库
- 第四节 使用Visual Studio编译

第一节 Windows操作系统

- Windows操作系统是微软公司推出的操作系统。正如其名，通过此操作系统可以建立窗口。
- 本教程所用的单元库Display使用了Windows API建立窗口并使用GDI+进行绘图，因此本教程只适用于Windows操作系统。请确保已经安装了以Windows NT为内核的操作系统。
- 在编写窗体应用程序时，可以在编译命令行中添加编译指令-mwindows防止控制台的创建。

第二节 GUN C语言编译器

- 为了编译C++语言程序，请下载并安装GUN C语言编译器：https://gcc.gnu.org/install/binaries.html
- 官方手册：https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html
- 你也可以使用任意一款你喜欢的IDE（例如notepad.exe）。
- 本教程的编译器以gcc version 5.3.0 (GCC)（g++.exe）为准。

第三节 Display单元库

- 本教程使用Display单元库实现游戏：

http://axpokl.com/display.zip

http://axpokl.ys168.com

- 编译前，请将disp.h拷贝到主程序同一个目录下，或者将disp.h拷贝到编译器的include文件夹。
- 编译程序时，请在源代码列表中加入disp.h。
- 建议编译时使用-static静态链接并使用-mwindows关闭控制台。
- Display单元库仍在不断更新，每个版本并不互相兼容，因此对于每一个程序应使用独立的Display单元库。

第三节 Display单元库

- 引入后，可以使用Display单元库的子程序：

```
#include "disp.h"//使用Display单元库
int main(){
msgbox(i2s(geterror()));//弹出错误代码
return 0;}
```

- geterror获取最后一个Windows API错误代码。
- i2s将整型转换为const char*型。
- msgbox弹出const char*型窗口文字。

第三节 Display单元库

- 本教程的文件清单：
- disp\display.pp：

Pascal语言Display单元库源代码。

- disp\display_createlib.exe：

输出导出dll所需的Pascal源代码的Pascal程序。

- disp\display_library.pas：

根据display.pp导出dll所需的Pascal源代码。

- disp\pas2c.exe：

将Pascal语言转为C语言的Pascal程序。

第三节 Display单元库

- disp\display_library_fun.h

由display_createlib.exe生成的含有函数清单的头文件。

- disp\disp.h：

由display_library_fun.h合并的用于C语言程序的头文件。

- disp\disp.def：

由display_createlib.exe生成的用于生成disp.lib的文件。

- disp\disp.dll：

由display_library.exe导出的用于C语言程序的dll文件。

第三节 Display单元库

- pas*.pas：

Free Pascal教程中pascal语言示例源代码。

- cpp*.cpp：

由pascal语言示例代码转来的C语言示例代码。

- bin*.pas.exe：

由pascal语言示例代码编译而来的示例程序。

- bin*.cpp.exe：

由C语言示例代码编译而来的示例程序。

第三节 Display单元库

- make(dll).bat：

编译display_createlib，display_library；生成disp.h和disp.dll并将必要文件复制到指定文件夹的脚本。

- make(pas2c).bat：

将pascal语言示例代码转为的C语言示例代码的脚本。

- make(pas).bat

编译pascal语言示例代码的脚本。

- make(cpp).bat

编译C语言示例代码的脚本。

第四节 使用Visual Studio编译

- 使用Visual Studio编译Display单元库时，您需要以下文件：

1.Display单元库头文件(disp.h)；

2.Display单元库静态连接库(disp.lib)；

3.Display单元库动态链接库(disp.dll)。

- 您可以使用Visual Studio的lib工具用disp.def文件创建disp.lib文件：

lib /def:disp.def /OUT:disp.lib

第四节 使用Visual Studio编译

- 创建Windows窗体应用程序项目时，请将程序主函数入口设为int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)。
- 请将disp.h，disp.lib和disp.dll复制项目的文件夹中，并分别将disp.h和disp.lib加入项目的头文件和资源文件中。
- 请删除disp.h中命名可能冲突的一些函数（例如max和min函数）。

第二章 窗口

- 第一节 建立窗口
- 第二节 设定窗口标题
- 第三节 判断窗口状态
- 第四节 获取窗口大小
- 第五节 关闭窗口

第一节 建立窗口

- 以下过程可以创建窗口：

```
void createwin();
void createwin(unsigned long c);
void createwin(unsigned long w,unsigned long h,unsigned long c);
```

- 其中w,h代表宽度和高度，如不指定w,h则默认使用屏幕一半宽高来建立窗口。c为默认颜色。
- 颜色c为四字节ABGR模式（A=透明，B=蓝，G=绿，R=红，各占一个字节）。

第一节 建立窗口

- 实例：

```
#include "disp.h"//使用Display单元库
int main(){
createwin();//建立窗口
msgbox("窗口已建立");//输出窗口建立信息
return 0;}
```

第二节 设定窗口标题

- 以下子程序可以设定或获取标题：

```
void settitle(const char* s);
const char* gettitle();
```

- 在使用string类型的时候，请注意类型的转换。
- 请注意const char*和char*的区别。

第二节 设定窗口标题

- 实例：

```
#include "disp.h"//使用Display单元库
int main(){
createwin();//建立窗口
settitle("我是标题");//设定窗口标题
msgbox(gettitle());//获取并输出窗口标题
return 0;}
```

第三节 判断窗口状态

- 以下函数可以判断窗口状态：

bool iswin();

- 如果窗口存在，则该函数返回true，否则返回false。

第三节 判断窗口状态

- 实例：

```
#include "disp.h"//使用Display单元库
int main(){
if(iswin())msgbox("yes");else msgbox("no");//输出窗口状态信息
createwin();//建立窗口
if(iswin())msgbox("yes");else msgbox("no");//输出窗口状态信息
return 0;}
```

第四节 获取窗口大小

- 以下函数可以获取窗口大小：

```
unsigned long getwidth();
unsigned long getheight();
unsigned long getsize();
```

- 其中，getsize的前两字节为宽，后两字节为高。可以用HiWord(getsize())和LoWord(getsize())获取。
- 如需改变窗口大小，需要使用消息传递函数。这会在教程的第二部分进行解说。

第四节 获取窗口大小

- 此外还有以下函数可以获取屏幕大小：

long getscrwidth();

long getscrheight();

unsigned long getscrsize();

- 以及以下函数可以获取窗口位置：

long getposx();

long getposy();

第四节 获取窗口大小

- 实例：

```
#include "disp.h"//使用Display单元库
int main(){
createwin();//建立窗口
msgbox(i2s(getwidth())+" "+i2s(getheight()));//输出窗口大小信息
return 0;}
```

第五节 关闭窗口

- 以下过程可以关闭窗口：

```
void closewin();
```

- closewin过程不仅会关闭窗口，还会释放窗口句柄及设备上下文句柄（HDC）。也就是说，无法再用createbmp函数获取窗口内容，也无法使用loadbmp函数读取图片。

第五节 关闭窗口

- 实例：

```
#include "disp.h"//使用Display单元库
int main(){
if(iswin())msgbox("yes");else msgbox("no");//输出窗口状态信息
createwin();//建立窗口
if(iswin())msgbox("yes");else msgbox("no");//输出窗口状态信息
closewin();//关闭窗口
```

第五节 关闭窗口

```
if(iswin())msgbox("yes");else msgbox("no");//输出窗口状态信息
createwin();//再次建立窗口
if(iswin())msgbox("yes");else msgbox("no");//输出窗口状态信息
closewin();//再次关闭窗口
if(iswin())msgbox("yes");else msgbox("no");//输出窗口状态信息
return 0;}
```

第三章 绘图

- 第一节 刷新窗口
- 第二节 绘制图形
- 第三节 读取图片
- 第四节 绘制图片
- 第五节 绘制拉伸图片
- 第六节 绘制透明图片
- 第七节 快速画点

第一节 刷新窗口

- 以下函数可以刷新窗口：

void freshwin();

- 绘图完毕后须刷新窗口才能使绘制的内容生效（默认情况下，绘图子程序会绘制到缓冲区）。
- 使用绘图子程序会占用CPU，因此应尽量避免使用或减少使用次数，例如用图片代替图形。
- 刷新窗口后，帧率会自动更新。详情请阅读第五章第四节帧率获取。

第一节 刷新窗口

- 实例：

```
#include "disp.h"//使用Display单元库
int main(){
createwin(red);//建立红色窗口
clear(blue);//清屏蓝色
msgbox("清屏没有生效");//这里清屏不会生效
freshwin();//刷新窗口
msgbox("清屏已生效");//这里清屏生效了
return 0;}
```

第二节 绘制图形

- 以下过程可以绘制图形：

void setpixel(unsigned long x,unsigned long y,unsigned long c);

void line(unsigned long x,unsigned long y,long w,long h,unsigned long c);

void bar(unsigned long x,unsigned long y,long w,long h,unsigned long c);

void bar(unsigned long x,unsigned long y,long w,long h,unsigned long cfg,unsigned long cbg);

第二节 绘制图形

void circle(long x,long y,long r,unsigned long cfg,unsigned long cbg);

void ellipse(long x,long y,long rx,long ry,unsigned long cfg,unsigned long cbg);

void ellipse(long x,long y,long rx,long ry,double sa,double ea,unsigned long cfg,unsigned long cbg);

void clear(unsigned long c);

void clear();

- clear过程调用了bar过程。

第二节 绘制图形

- 实例：

```
#include "disp.h"//使用Display单元库
unsigned long n=0x1000;//绘制点数量
int main(){
createwin();//建立窗口
line(10,10,100,100,red);//绘制直线
bar(110,10,100,100,blue);//绘制矩形
circle(60,160,50,green);//绘制圆形
ellipse(185,160,25,50,pink);//绘制椭圆
```

第二节 绘制图形

```
while(n>0)
{
setpixel(random(getwidth()),random(getheight()),rando
m(0xffffff));//随机画点
n=n-1;
}
freshwin();//刷新窗口
msgbox("绘制完成");//输出绘制完成信息
return 0;}
```

第三节 读取图片

- 读取图片之前，请先创建pbitmap类型变量。
- pbitmap和bitmap类型结构：

```
struct bitmap {
  unsigned long handle;
  unsigned long dc;
  unsigned long width;
  unsigned long height;
  unsigned long color;};
typedef bitmap* pbitmap;
```

第三节 读取图片

- 以下函数可以读取图片：

pbitmap loadbmp(const char* s);

pbitmap loadbmp(const char* s,unsigned long c);

- s为文件名，c为图片背景颜色（默认透明色）。
- 函数返回pbitmap类型的图片。
- 支持的格式：BMP，PNG，JPG，GIF和TIF。
- 在读取图片之前必须先创建窗口（因为创建图片时需要创建和窗口兼容的设备上下文句柄）。

第三节 读取图片

- 实例：

```
#include "disp.h"//使用Display单元库
pbitmap img;
int main(){
createwin();//建立窗口
img=loadbmp("display.png");
msgbox(i2s(img->width)+" "+i2s(img->height));//输出图片信息
return 0;}
```

第四节 绘制图片

- 以下过程可以绘制图片：

void drawbmp(pbitmap b,unsigned long xd,unsigned long yd);

void drawbmp(pbitmap bs,pbitmap bd,unsigned long xd,unsigned long yd);

xd,yd为目标坐标，b,bs为需要绘制的图片，bd为绘制的目标。未指定bd时绘制到窗口。

- bar,line等绘图过程也可以绘制到图片，只需在第一个参数加入pbitmap的图片变量即可。

第四节 绘制图片

- 实例：

```
#include "disp.h"//使用Display单元库
pbitmap img1,img2;
int main(){
createwin(800,600);//建立窗口
img1=loadbmp("display.png");//读取图片
img2=loadbmp("display.png");//读取图片
drawbmp(img1,(getwidth()-img1->width)/2,(getheight()-
img1->height)/2);//绘制图片1
```

第四节 绘制图片

```
freshwin();//刷新窗口
msgbox("图片1");//输出绘制完成信息
bar(img2,img2->width/4,img2->height/4,img2->width/2,img2->height/2,transparent,blue);//绘制矩形到图片2中间
drawbmp(img2,(getwidth()-img2->width)/2,(getheight()-img2->height)/2);//绘制图片2
freshwin();//刷新窗口
msgbox("图片2");//输出绘制完成信息
return 0;}
```

第五节 绘制拉伸图片

- 以下过程可以绘制拉伸的图片：

procedure void drawbmp(pbitmap b,unsigned long xs,unsigned long ys,unsigned long ws,unsigned long hs,unsigned long xd,unsigned long yd,unsigned long wd,unsigned long hd);

- wd,hd为目标大小，不能为负数（不能反射）。
- xs,ys为需要绘制的图片左上角的位置，ws,hs为需要绘制的图片从xs,ys开始的大小。
- ws,hs必须比原始图片小，否则绘图将会失败。

第五节 绘制拉伸图片

- 实例：

```
#include "disp.h"//使用Display单元库
pbitmap img;
int main(){
createwin(800,600);//建立窗口
img=loadbmp("display.png");//读取图片
drawbmp(img,(getwidth()-img->width*2)/2,(getheight()-img->height*2)/2,img->width*2,img->height*2);//绘制拉伸图片
```

第五节 绘制拉伸图片

```
freshwin();//刷新窗口
msgbox("绘制拉伸完成");//输出绘制完成信息
drawbmp(img,img->width/4,img->height/4,img->width/2,img->height/2,(getwidth()-img->width*2)/2,(getheight()-img->height*2)/2,img->width*2,img->height*2);//绘制剪切拉伸图片
freshwin();//刷新窗口
msgbox("绘制剪切完成");//输出绘制完成信息
return 0;}
```

第六节 绘制透明图片

- 以下过程可以绘制透明和半透明的图片：

void drawbmp(pbitmap b,unsigned long c);

- c为透明颜色。如未指定c，则会使用b的背景颜色作为透明色。
- 该过程只会绘制图片中不是透明色的部分。
- 当c的Alpha部分不为0时（getalpha(c)<>0），绘图过程会以半透明的形式绘制到目标上（会创建临时位图并消耗资源，慎用）。

第六节 绘制透明图片

- 实例：

```
#include "disp.h"//使用Display单元库
int main(){
createwin(800,600,red);//建立窗口
drawbmp(loadbmp("display.png"),0x7fffffff);//绘制透明色半透明图片
freshwin();//刷新窗口
msgbox("绘制透明完成");//输出绘制完成信息
return 0;}
```

第七节 快速画点

- 读取图片之前，请先创建pbitbuf类型变量。
- pbitbuf和bitbuf类型结构：

```
struct bitbuf {
  BITMAPINFO bmi;
  unsigned long len;
  void* buf;
  pbitmap bmp;
};
typedef bitbuf* pbitbuf;
```

第七节 快速画点

- 一个位图缓存必须且只能和一个位图绑定。
- 请使用以下函数创建位图缓存：

pbitbuf createbb(pbitmap b);

- 您可以使用以下函数直接获取窗口位图：

pbitmap getwin();

- 绘制完毕后，请释放位图缓存：

void releasebb(pbitbuf bb);

第七节 快速画点

- 以下函数可以快速取点和画点：

unsigned long getbbpixel(pbitbuf bb,unsigned long x,unsigned long y);

void setbbpixel(pbitbuf bb,unsigned long x,unsigned long y,unsigned long c);

- 请使用以下过程绘制缓存到位图：

void setbb(pbitbuf bb);

- 您也可以读取位图到缓存：

void getbb(pbitbuf bb);

第七节 快速画点

- 实例：

```
#include "disp.h"//使用Display单元库
unsigned long n=0x100000;//绘制点数量
pbitbuf bb;//位图缓存
int main(){
createwin(800,600,red);//建立窗口
bb=createbb(getwin());//创建位图缓存
while(n>0)
```

第七节 快速画点

```
{
setbbpixel(bb,random(getwidth()),random(getheight()),random(0xffffff));//随机画点到缓存
n=n-1;
}
setbb(bb);//绘制缓存到位图
freshwin();//刷新窗口
msgbox("快速画点完成");//输出绘制完成信息
return 0;}
```

第四章 文字

- 第一节 输出文字
- 第二节 设定字体大小
- 第三节 设定字体

第一节 输出文字

- 以下过程可以输出文字：

void drawtext(const char* s,unsigned long cfg,unsigned long cbg);

void drawtext(const char* s,unsigned long c);

void drawtext(const char* s);

- s为需要输出的字符串。
- cfg,cbg分别为文字的颜色和背景色。如果只指定c，则背景色为透明（不绘制背景色）。不指定c时，默认的文字颜色是窗体的前景色。

第一节 输出文字

- 可以使用XY系列过程将文字输出到指定位置：

```
void drawtextxy(const char* s,unsigned long
x,unsigned long y);
```

- 或者ln系列过程按行效果输出：

```
void drawtextln(const char* s);
```

- 使用w系列过程可输出定宽文本：

```
void drawtextw(const char* s);
```

- 定宽字符宽度取决于字体，请先定义字体大小。
- 部分过程也可将文字输出到指定图片。

第一节 输出文字

- 实例：

```
#include "disp.h"//使用Display单元库
int main(){
createwin();//建立窗口
drawtextln("ax_pokl output text.");//输出文本并换行
drawtextw("ax_pokl""s text is tight");//输出定宽文本（宽度默认为0）
drawtextxy("ax_pokl output text anywhere",50,50);//指定位置输出文本
```

第一节 输出文字

```
drawtextxy("",0,80);//强制改变输出位置
drawtext("and it""s colorful ",orange);//输出带颜色文本
drawtext("with backgroud color",red,blue);//输出带背景颜色文本
freshwin();//刷新窗口
msgbox("绘制完成");//输出绘制完成信息
return 0;}
```

第二节 设定字体大小

- 以下过程可以设定字体大小：

void setfontwidth(unsigned long w);

void setfontheight(unsigned long h);

void setfontsize(unsigned long w,unsigned long h);

- w,h为宽和高，设为0时有特殊含义：
- h为0时，将使用系统默认的高度。
- w为0时，宽度将匹配高度。
- 设为0时，虽然显示的字体有宽和高，但其宽或高仍旧为0。

第二节 设定字体大小

- 实例：

```
#include "disp.h"//使用Display单元库
unsigned long pyi;
int main(){
createwin();//建立窗口
setfontsize(5,10);//宽5,高10
drawtextxy("5,10",0,0,white,red);
setfontheight(20);//高20,宽不变
drawtextxy("-,20",0,20,white,red);
```

第二节 设定字体大小

```
setfontsize(0,0);//默认大小
drawtextxy("匹配,默认",0,40,white,red);
setfontsize(0,20);//高20,宽匹配
drawtextxy("匹配,20",0,60,white,red);
for(pyi=0;pyi<=4;pyi++)line(0,pyi*20,long(getwidth),0,or
ange);
freshwin();//刷新窗口
msgbox("绘制完成");//输出绘制完成信息
return 0;}
```

第三节 设定字体

- 以下过程可以设定字体：

```
void setfontweight(unsigned long wg);
void setfontItalic(unsigned long lt);
void setfontunderline(unsigned long ud);
void setfontstrikeout(unsigned long sk);
void setfontname(const char* s);
```

- 粗细wg默认为0，标准为400，粗体为700。
- 斜体lt，下划线ud，删除线sk为0或1。
- 用setfont(pbitmap b);可将当前字体选入图片。

第三节 设定字体

- 实例：

```
#include "disp.h"//使用Display单元库
int main(){
createwin();//建立窗口
setfontname("comic sans ms");//字体名称
drawtextxy("comic sans ms",0,0);
setfontweight(700);//粗体
drawtextxy("weight",0,20);setfontweight(0);
```

第三节 设定字体

```
setfontItalic(1);//斜体
drawtextxy("Italic",0,40);setfontItalic(0);
setfontunderline(1);//下划线
drawtextxy("unterline",0,60);setfontunderline(0);
setfontstrikeout(1);//删除线
drawtextxy("strikeout",0,80);setfontstrikeout(0);
freshwin();//刷新窗口
msgbox("绘制完成");//输出绘制完成信息
return 0;}
```

第五章 消息

- 第一节 获取消息
- 第二节 处理消息
- 第三节 获取时间
- 第四节 获取帧率
- 第五节 控制帧率

第一节 获取消息

- 以下函数用以判断或获取消息：

bool isnextmsg();

unsigned long getnextmsg();

unsigned long waitnextmsg();

- 窗口线程中的消息会由线程自动发送给窗口，然后同时自动刷新消息缓存。
- 对于非线程消息，窗口线程不会发送。
- isnextmsg会返回队列中是否有新消息。getnextmsg和waitnexgmsg会返回消息号。

第一节 获取消息

- 在消息缓存中被放入新消息之前，waitnextmsg不会返回。isnextmsg和getnextmsg会立即返回，无论是否有新消息。
- 使用getnextmsg前请务必先使用isnextmsg或waitnextmsg刷新窗口消息缓存计数，否则将取不到下一条消息。
- getnextmsg会返总是返回刷新消息缓存计数后的当前计数的消息的消息号。

第一节 获取消息

- 实例：

```
#include "disp.h"//使用Display单元库
int main(){
createwin();//建立窗口
do{//第一种消息循环
if(isnextmsg())settitle(i2s(getnextmsg()));//如果有新消息则输出消息号到标题栏
else delay();//否则等待1毫秒
}while(!(!(iswin())||(iskey(27))));//直到关闭窗口或按ESC
```

第一节 获取消息

```
while(iswin())//如果窗口开着则循环
{//第二种消息循环
waitnextmsg();//等待新消息
settitle(i2s(getnextmsg()));//输出消息号到标题栏
if(iskey(27))closewin();//如果是按ESC则关闭窗口
}//直到关闭窗口
return 0;}
```

第二节 处理消息

- 获取消息后，可以使用以下函数进行处理

bool ismsg(unsigned long um);

long long getmsg(unsigned long um);

long long waitmsg(unsigned long um);

- ismsg用以判断当前消息是否指定消息。
- getmsg可获取消息的参数。如果当前消息不是指定消息，则函数返回0。
- waitmsg会等待指定消息并返回消息的参数。

第二节 处理消息

- 以下函数可以判断特定类型消息：

```
bool iskey();
bool iskey(unsigned long k);
bool ismouse();
bool ismouse(unsigned long m);
bool ismouseleft();
bool ismousemove();
bool isdropfile();
```

- 部分以上函数也有get和wait的版本。

第二节 处理消息

- 以下函数可获取鼠标的位置：

```
long getmouseabsx();
long getmouseabsy();
long getmousewinx();
long getmousewiny();
long getmouseposx();
long getmouseposy();
```

- abs，win和pos分别为鼠标的绝对坐标，窗口坐标和绘图区坐标。

第二节 处理消息

- 实例：

```
#include "disp.h"//使用Display单元库
int main(){
createwin();//建立窗口
do{
settitle(i2s(getmouseposx())+"
"+i2s(getmouseposy()));//输出鼠标位置到标题栏
waitnextmsg();
if(iskey())msgbox(i2s(getkey()));//如果是按键则输出按
键号
```

第二节 处理消息

```
if(ismouse())msgbox(i2s(getmouse()));//如果按鼠标则输出鼠标号
if(ismousewheel())msgbox(i2s(getmousewheel()));//如果鼠标滚轮则输出滚轮号
if(isdropfile())msgbox(getdropfile());//如果是拖拽文件则输出文件名
}while(!(!(iswin())||(iskey(27))));//直到关闭窗口或按ESC键
return 0;}
```

第三节 获取时间

- 以下函数可以获取时间：

double gettimer();

unsigned long gettime();

- gettimer返回从窗口建立开始到现在的时间。
- gettime返回整型时间，以毫秒计。

第三节 获取时间

- 实例：

```
#include "disp.h"//使用Display单元库
int main(){
createwin();//建立窗口
do{
settitle(i2s(gettime()));//输出时间到标题栏
isnextmsg();delay();////等待新消息并延迟1毫秒
}while(!(!(iswin())||iskey()));//直到窗口关闭或按键
return 0;}
```

第四节 获取帧率

- 以下函数可以获取帧率：

unsigned long getfpsl();

double getfpsr();

unsigned long getfps();

- getfpsl返回从一秒前开始到当前的帧数（刷新次数，即调用freshwin的次数）。
- getfpsr返回getfpsl*一秒前开始第一帧到当前帧（最后刷新）的时间（时间小于1秒）。
- getfps返回getfpsr取整的结果。

第四节 获取帧率

- 实例：

```
#include "disp.h"//使用Display单元库
int main(){
createwin();//建立窗口
do{
isnextmsg();//等待新消息
clear();//清屏
drawtextxy("",0,0);//设置文本输出位置
```

第四节 获取帧率

```
drawtextln(i2s(getfpsl()));//输出瞬时刷新率
drawtextln(i2s(getfps()));//输出平均刷新率
freshwin();//刷新窗口
delay();//延迟1毫秒
}while(!(!(iswin())||iskey()));//直到窗口关闭或按键
return 0;}
```

第五节 处理帧率

- 以下过程可以延迟时间：

void delay(unsigned long t);

void delay();

- t为unsigned long时以毫秒计。
- 最短延迟时间视系统状态而定，这可能是1000/60毫秒或者1毫秒（1系统tick）。

第五节 处理帧率

- 实例：

```
#include "disp.h"//使用Display单元库
double frame=120.0;//帧率
double frametime=0.0;//当前帧时间
int main(){
createwin();//建立窗口
do{
if(isnextmsg())//如果有新消息
{
```

第五节 处理帧率

```
if((frame>10)&&iskey(37))frame=frame-1;//如果按左则减小帧率
if((frame<480)&&iskey(39))frame=frame+1;//如果按右则增加帧率
}
if(gettimer()>frametime+1/frame)//如果当前时间已超过一帧时间
{
while(gettimer()>frametime+1/frame)frametime=frametime+1/frame;//增加帧数（包括跳帧）
```

第五节 处理帧率

```
clear();//清屏
drawtextlnxy(i2s(getfpsl()),0,0);//输出瞬时刷新率
drawtextln(i2s(getfps()));//输出平均刷新率
drawtextln(i2s(long(round(frame))));//输出瞬时刷新率
freshwin();//刷新窗口
}
delay();//延迟1毫秒
}while(!(!(iswin())||iskey(27)));//直到窗口关闭或按ESC
return 0;}
```

第六章 音频

- 第一节 读取音频
- 第二节 播放音频
- 第三节 设定音量
- 第四节 暂停音频
- 第五节 跳转音频
- 第六节 音频播放器

第一节 读取音频

- 以下函数可以读取音频，请在读取之前先创建unsinged long类型变量id：

unsigned long loadaudio(const char* s);

- 之后对音频的操作需要这个id变量。
- 可以同时读取多个音频，用id区分。
- 支持的音频格式有wav，mp3，wmv等系统内生支持的格式，和Windows media player相同。

第一节 读取音频

- 实例：

```
#include "disp.h"//使用Display单元库
unsigned long audio;
int main(){
audio=loadaudio("display.mp3");//读取音频
msgbox(i2s(audio));//输出音频号
return 0;}
```

第二节 播放音频

- 以下过程可以播放音频：

```
void playaudio(unsigned long id,bool b);
void playaudio(unsigned long id);
```

- b为true时，音频将重复播放。
- 不指定b时默认为false（单曲播放）。
- 实例：

```
#include "disp.h"//使用Display单元库
unsigned long audio1,audio2;
```

第二节 播放音频

```
int main(){
audio1=loadaudio("display.mp3");//读取音频1
audio2=loadaudio("display.mp3");//读取音频2
playaudio(audio1);//播放音频1
msgbox("正在播放音频");
playaudio(audio2,true);//重复播放音频2
msgbox("正在重复播放音频");
return 0;}
```

第三节 设定音量

- 播放过程中可获取或设定音频音量：

unsigned long getaudiovol(unsigned long id);

void setaudiovol(unsigned long id,unsigned long v);

- 音量v为unsigned long类型，范围为0到1000。
- 每个音频可以设定不同的音量。
- 音频必须在开始播放以后才能设定音量。

第三节 设定音量

- 实例：

```
#include "disp.h"//使用Display单元库
unsigned long audio;
int main(){
audio=loadaudio("display.mp3");//读取音频
playaudio(audio);//播放音频
setaudiovol(audio,200);//设定音量
msgbox("正在播放音频，音量200");
return 0;}
```

第四节 暂停音频

- 以下过程可以实现音频的暂停，继续，停止：

```
void pauseaudio(unsigned long id);
void resumeaudio(unsigned long id);
void stopaudio(unsigned long id);
void releaseaudio(unsigned long id);
```

- pause和resume可以暂停，继续音频的播放。
- 用stop停止播放音频后，可以用play重新播放。
- 如需彻底将音频从内存中释放，请用release。

第四节 暂停音频

- 实例：

```
#include "disp.h"//使用Display单元库
unsigned long audio;
int main(){
audio=loadaudio("display.mp3");//读取音频
playaudio(audio);//播放音频
msgbox("正在播放音频，按确定暂停");
pauseaudio(audio);
msgbox("音频已暂停，按确定继续");
```

第四节 暂停音频

```
resumeaudio(audio);
msgbox("音频已继续重复播放");
stopaudio(audio);
msgbox("音频已停止播放");
playaudio(audio);
msgbox("音频已重新开始播放");
releaseaudio(audio);
msgbox("音频已释放");
return 0;}
```

第五节 跳转音频

- 以下过程可以实现音频的跳转，以及获取音频的播放位置和长度：

unsigned long getaudiolen(unsigned long id);

unsigned long getaudiopos(unsigned long id);

void setaudiopos(unsigned long id,unsigned long pos,bool b);

void setaudiopos(unsigned long id,unsigned long pos);

- 位置和长度为unsinged long，以毫秒计。
- 如需从指定位置重复播放，请设b为true。

第五节 跳转音频

- 实例：

```
#include "disp.h"//使用Display单元库
unsigned long audio;
int main(){
audio=loadaudio("display.mp3");//读取音频
playaudio(audio);setaudiopos(audio,getaudiolen(audio)/2);//从中间开始播放音频
msgbox("正在播放音频，从中间开始");
return 0;}
```

第七章 应用

- 第一节 音频播放器
- 第二节 俄罗斯方块

第一节 音频播放器

- 使用前六章知识，编写简易音频播放器：

```
#include "disp.h"//使用Display单元库
unsigned long w=600;//窗口宽
unsigned long h=100;//窗口高
double frame=120.0;//帧率
double frametime=0.0;//当前帧时间
unsigned long audio;//音频句柄
unsigned long pos;//音频窗口位置
bool play=false;//音频播放状态
```

第一节 音频播放器

```
int main(){
createwin(w,h,blue);//建立蓝色窗口
settitle("display.mp3");//设定标题
audio=loadaudio("display.mp3");//读取音频
playaudio(audio);//播放音频
do{//开始消息循环
if(isnextmsg())//如果有新消息
{
if(isdropfile())//如果有拖拽文件
```

第一节 音频播放器

```
{
settitle(getdropfile());//设定标题为拖拽文件名
stopaudio(audio);//停止正在播放的音频
audio=loadaudio(getdropfile());//读取音频
playaudio(audio);//播放音频
play=true;//设定音频状态
}
if(iskey(37))//如果按左
setaudiopos(audio,max(getaudiopos(audio)-1000,0));//
倒退1秒
```

第一节 音频播放器

if(iskey(39))//如果按右

setaudiopos(audio,min(getaudiopos(audio)+1000,getaudiolen(audio)));//前进1秒

if(iskey(40))//如果按下

setaudiovol(audio,max(getaudiovol(audio)-100,0));//减小100音量

if(iskey(38))//如果按上

setaudiovol(audio,min(getaudiovol(audio)+100,1000));//增大100音量

if(ismouseleft())//如果鼠标左键

第一节 音频播放器

```
setaudiopos(audio,round(getmouseposx()/double(w)*getaudiolen(audio)));//跳转音频
if(ismouseright()||iskey(32))//如果鼠标右键或按空格
{
if(play)pauseaudio(audio);//如果正在播放则暂停
else resumeaudio(audio);//否则继续播放
play=not(play);//更改音频状态
}
}
```

第一节 音频播放器

if(getaudiopos(audio)==getaudiolen(audio))//如果已播放完毕

setaudiopos(audio,0);//重头播放

if(gettimer()>frametime+1/frame)//如果当前时间已超过一帧时间

{

while(gettimer()>frametime+1/frame)frametime=frametime+1/frame;//增加帧数（包括跳帧）

if(getaudiolen(audio)==0)pos=0;//如果音频长度为0（没有音频）则设音频窗口位置为0

第一节 音频播放器

```
else pos=round(double(w)*getaudiopos(audio)/
getaudiolen(audio));//否则设定音频窗口位置
clear();bar(pbitmap(0),0,0,pos,100,transparent,yellow);//
//绘制状态
drawtextlnxy(i2s(getaudiopos(audio))+" /
"+i2s(getaudiolen(audio)),0,0,yellow,blue);//输出状态
freshwin();}//刷新窗口
delay();//延迟1毫秒
}while(!(!(iswin())||(iskey(27))));//直到关闭窗口或按ESC
return 0;}
```

第二节 俄罗斯方块

- 一款简易的俄罗斯方块小游戏：

```
//可使用-mwindows编译指令关闭控制台
#include "disp.h"//使用Display单元库
const unsigned long w=10;//场地宽
const unsigned long h=20;//场地高
unsigned long sz=30;//方块大小
double frame=120.0;//帧率
double frametime=0.0;//当前帧时间
double downtime=0.0;//下落时间
```

第二节 俄罗斯方块

```
char i,j;//场地行列计数
char x,y,r,k;//当前方块状态
char bd[w-1+1][h-1+1];//场地方块
const unsigned long bdc[7+1]=
{0x1f1f1f,0x7f7f7f,0x7f7fff,0x7fff7f,0xff7f7f,0x7fffff,0xffff7f,0xff7fff};//方块颜色
const unsigned long bdk[7+1][3+1][3+1][3+1];//方块类型（已省略）
```

第二节 俄罗斯方块

```
void drawblock(char i,char j,char k)//画方块
{ bar(i*sz,(h-j-1)*sz,sz,sz,bdc[k]);}
void newblock();//新方块
void restart()//重新开始
{
for(i=0;i<=w-1;i++)for(j=0;j<=h-1;j++)bd[i][j]=0;//清空场地
newblock();//新方块
}
```

第二节 俄罗斯方块

```
bool eraseline()//消行
{bool eraseline_r;
for(j=0;j<=h-1;j++)//从最底行开始
{
eraseline_r=true;//是满行
for(i=0;i<=w-1;i++)if(bd[i][j]==0)eraseline_r=false;//如果有洞则不是
if(eraseline_r)break;//如果是满行则跳出
}
```

第二节 俄罗斯方块

```
if(eraseline_r)//如果是满行（消行）
while(j<(h-1))//从此行开始（往上）
{
for(i=0;i<=w-1;i++)//遍历该行
bd[i][j]=bd[i][j+1];//上方方块掉落
j=j+1;//继续上一行
}
return eraseline_r;}
```

第二节 俄罗斯方块

```
void fixblock()//固定方块（落底）
{
for(i=0;i<=3;i++)for(j=0;j<=3;j++)//遍历方块行列
if(bdk[k][r][j][i]>0)bd[i+x][j+y]=k;newblock();//如果是格子非空则画到场地
while(eraseline());//消行
}
```

第二节 俄罗斯方块

```
bool overlay()//判断重叠
{bool overlay_r;
overlay_r=false;//设非重叠
for(i=0;i<=3;i++)for(j=0;j<=3;j++)//遍历方块行列
if((bdk[k][r][j][i]>0))//如果格子非空
if((i+x<0)||(i+x>=w)||(j+y<0)||(j+y>=h))overlay_r=true;//
如果超出场地则重叠
else if((bd[i+x][j+y]>0))overlay_r=true;//如果没超出场地
但场地非空也重叠
return overlay_r;}
```

第二节 俄罗斯方块

```
void newblock()//新方块
{
x=3;//新方块行
y=16;//新方块列
r=0;//新方块旋转
k=random(7)+1;//新方块类型
if(overlay())restart();//如果重叠则重来
}
```

第二节 俄罗斯方块

```
bool rotate(char d)//旋转
{bool rotate_r;
r=r+1;if(r>3)r=0;rotate_r=not(overlay());//尝试旋转
if(!(rotate_r))r=r-1;if(r<0)r=3;//如果不能旋转则转回来
return rotate_r;}
```

第二节 俄罗斯方块

```
bool move(char dx,char dy)//移动
{bool move_r;
x=x+dx;y=y+dy;move_r=not(overlay());//尝试移动
if(!(move_r)){x=x-dx;y=y-dy;};//如果不能移动则移回来
if(!(move_r)&&(dy<0))fixblock();//如果不能移动且下落则固定
if(dy<0)downtime=gettimer();//如果下落则重置下落时间
return move_r;}
```

第二节 俄罗斯方块

```
int main(){//主程序
randomize();//初始化随机种子
createwin(w*sz,h*sz);//建立窗口
settitle("俄罗斯方块");//设定标题
restart();//重新开始
do{//开始消息循环
if(isnextmsg())//如果有新消息
{
```

第二节 俄罗斯方块

```
if(iskey(37))move(-1,0);//如果按左则左移
if(iskey(39))move(+1,0);//如果按右则右移
if(iskey(40))move(0,-1);//如果按下则下落
if(iskey(38))rotate(1);//如果按上则旋转
if(iskey(32))while(move(0,-1));//如果按空格则下底
}
if(gettimer()>downtime+1)move(0,-1);//如果超过1秒则下落
```

第二节 俄罗斯方块

```
if(gettimer()>frametime+1/frame)//如果当前时间已超过一帧时间
{
while(gettimer()>frametime+1/frame)frametime=frametime+1/frame;//增加帧数（包括跳帧）
clear();
for(i=0;i<=w-1;i++)for(j=0;j<=h-1;j++)drawblock(i,j,bd[i][j]);//画场地
for(i=0;i<=3;i++)for(j=0;j<=3;j++)if(bdk[k][r][j][i]>0)drawblock(i+x,j+y,k);//画当前方块
```

第二节 俄罗斯方块

```
freshwin();//刷新窗口
}
delay();//延迟1毫秒
}while(!(!(iswin())||(iskey(27))));//直到关闭窗口或按ESC键
return 0;}
```

附录

- Display单元库重载表

Display单元库重载表

- 详情请参阅display.pp，以下列出部分重载：

```
const char* i2s(long i);
unsigned long newthread(void* th);
void msgbox(const char* s,const char* title);
void msgbox(const char* s);
void delay(unsigned long t);
double getfpsr();
unsigned long getfps();
unsigned long geterror();
```

Display单元库重载表

```
void createwin(unsigned long w,unsigned long h,unsigned long c);
void createwin(unsigned long w,unsigned long h);
void createwin(unsigned long c);
void createwin();
void freshwin();
void closewin();
bool iswin();
void setdrawprocedure(void* th);
```

Display单元库重载表

```
void settitle(const char* s);
const char* gettitle();
double gettimer();
unsigned long gettime();
unsigned long getwidth();
unsigned long getheight();
unsigned long getsize();
long getposx();
long getposy();
```

Display单元库重载表

unsigned long getbgcolor();

void setbgcolor(unsigned long c);

unsigned long getfgcolor();

void setfgcolor(unsigned long c);

unsigned char getblue(unsigned long c);

unsigned char getgreen(unsigned long c);

unsigned char getred(unsigned long c);

unsigned char getalpha(unsigned long c);

unsigned long getrgb(unsigned char r,unsigned char g,unsigned char b);

Display单元库重载表

```
void setfont(pbitmap b);
void setfontwidth(unsigned long w);
void setfontheight(unsigned long h);
void setfontsize(unsigned long w,unsigned long h);
void setfontweight(unsigned long wg);
void setfontItalic(unsigned long It);
void setfontunderline(unsigned long ud);
void setfontstrikeout(unsigned long sk);
void setfontname(const char* s);
```

Display单元库重载表

void drawtextxy(pbitmap b,const char* s,unsigned long x,unsigned long y,unsigned long cfg,unsigned long cbg);

void drawtextxy(const char* s,unsigned long x,unsigned long y,unsigned long c);

void drawtextxy(const char* s,unsigned long x,unsigned long y);

void drawtext(const char* s);

void drawtextln(const char* s);

void drawtextw(const char* s);

Display单元库重载表

unsigned long getpixel(unsigned long x,unsigned long y);

void setpixel(unsigned long x,unsigned long y,unsigned long c);

void line(unsigned long x,unsigned long y,long w,long h,unsigned long c);

void bar(unsigned long x,unsigned long y,long w,long h,unsigned long cfg,unsigned long cbg);

void clear(unsigned long c);

Display单元库重载表

pbitmap createbmp(unsigned long w,unsigned long h);

pbitmap loadbmp(const char* s);

void releasebmp(pbitmap b);

void drawbmp(pbitmap bs,pbitmap bd,unsigned long xs,unsigned long ys,unsigned long ws,unsigned long hs,unsigned long xd,unsigned long yd,unsigned long wd,unsigned long hd,unsigned long c);

void drawbmp(pbitmap b,unsigned long xs,unsigned long ys,unsigned long xd,unsigned long yd,unsigned long w,unsigned long h);

Display单元库重载表

void drawbmp(pbitmap b,unsigned long xd,unsigned long yd,unsigned long wd,unsigned long hd);

void drawbmp(pbitmap b,unsigned long xd,unsigned long yd,unsigned long c);

void drawbmp(pbitmap b,unsigned long c);

void drawbmp(pbitmap b);

void drawbmp(unsigned long xd,unsigned long yd,unsigned long wd,unsigned long hd);

void drawbmp(unsigned long xd,unsigned long yd);

void drawbmp();

Display单元库重载表

```
bool isnextmsg();
unsigned long getnextmsg();
unsigned long waitnextmsg();
bool iskey();
bool iskey(unsigned long k);
bool ismouse();
bool ismouse(unsigned long m);
bool ismouseleft();
bool ismousemiddle();
```

Display单元库重载表

```
bool ismouseright();
bool ismousewheel();
long getmousewheel();
bool ismousemove();
unsigned long getmousemove();
bool isdropfile();
const char* getdropfile();
long getmouseposx();
long getmouseposy();
```

Display单元库重载表

```
unsigned long loadaudio(const char* s);
void playaudio(unsigned long id,const char* s,bool b);
void playaudio(unsigned long id,const char* s);
void playaudio(unsigned long id,bool b);
void playaudio(unsigned long id);
void pauseaudio(unsigned long id);
void resumeaudio(unsigned long id);
```

Display单元库重载表

void stopaudio(unsigned long id);

void releaseaudio(unsigned long id);

unsigned long getaudiovol(unsigned long id);

void setaudiovol(unsigned long id,unsigned long v);

unsigned long getaudiopos(unsigned long id);

void setaudiopos(unsigned long id,unsigned long pos,bool b);

void setaudiopos(unsigned long id,unsigned long pos);

unsigned long getaudiolen(unsigned long id);

Display单元库重载表

- disp.h头文件中已默认包含了windows.h，因此你也可以在程序中直接使用Windows API。
- 你也可以包含其它的头文件，如stdio.h和stdlib.h。使用的时候注意命名空间以及类型，变量或函数名称的冲突。
- disp.h中使用了mystring类对const char*类型进行了类型转换和操作符重载。如果想让程序加快运行速度，或者转换中出现问题，可以自行修改头文件中的代码。

后记

- 使用简单的办法用Pasca语言开发窗体应用软件和游戏是我的愿望，因此我便编写了Display单元库。现在，这个愿望已经实现了。
- 由于Pascal语言日渐衰落，而学习C语言程序的人很多。受到了ege库的启发，让我有了开发针对C语言的窗体应用软件库的想法。
- 从开始只有几十个子程序，经过五年的修改，Display单元库已知在不断的完善和扩充。对于库的错误和建议，也请各位大神多多指点。