**HGE读书笔记（一）--绘制，声音，和碰撞处理**

2013年04月18日 11:38:19

阅读数：3574

HGE读书笔记一：

1.库的安装和下载

从官网上<http://hge.relishgames.com/>上可以下载到最新的HGE游戏引擎，此游戏引擎是完全免费开源的。在此感谢此引擎的编写者，感谢他的无私奉献。本教程是入门的教程，可是在此也并不介绍如何配置环境，这些操作网上琳琅满目，读者可自行上网收索

2.HGE实例

***#include<hge.h>***

***#pragma comment(lib,"hge.lib")***

Q:为什么要导入这个lib?不是在配置里面添加了吗？

A:由于不可知的原因，在VC6.0中配置时，加入了lib之后，所有其他的函数都可以使用，只有hgeCreate函数总是提示“无法解析的外部符号”。而这个函数是创建HE游戏引擎的函数，可谓是巨头函数，这个函数不能用，那整个程序就Game Over了。所以在用代码导入这个lib之后，就可以使用了。

***//定义全局变量***

***HGE \* hge = NULL ;***

***hgeQuad quad ;   //用于保存绘图的顶点，纹理，和混色的函数***

Q:hgeQuad结构是什么？

A:查看HEG的官网，上面解释说这个结构是跟绘图有关的结构。如果要绘图，就需要有这个结构存在。以下是这个结构的详细定义：

struct hgeQuad

{

       hgeVertex v[4] ;

       HTEXTURE tex ;

       int blend ;

};

上面结构中的几个成员分别是hgeVertex顶点结构，用于保存顶点的信息；第二个是HTEXTURE结构的数据，此结构是纹理的句柄；blend表示混色模式。

***HEFFECT snd ;    //保存音效的句柄***

***float x = 100.0f , y = 100.0f ;***

***float dx = 0.0f , dy = 0.0f ;***

***const float speed = 90 ;***

***const float friction = 0.98f ;***

***//定义播放音效的函数***

***void boom()***

***{***

***int pan = int((x-400)/4);***

***float pitch = (dx\*dx+dy\*dy)\*0.0005f + 0.2f ;***

***hge->Effect\_PlayEx(snd,100,pan,pitch); //snd是音频资源的句柄，100表示声音大小，pan表示左右声道，pitch表示播放频率***

***}***

Q:Effect\_PlayEx函数是什么用处？

A：Effect\_PlayEx函数，用于播放指定的音效文件，同时可以根据后面的几个参数来进行声音，声道，播放频率的改变。

***//定义帧回调函数***

***bool FrameFunc()***

***{***

***float dt = hge->Timer\_GetDelta();   //获取自上次调用此函数到此次调用此函数的时间间隔***

***if(hge->Input\_GetKeyState(HGEK\_ESCAPE))***

***return true ;***

***//根据不同的按键，使用速度和延迟时间来修正位置***

***if(hge->Input\_GetKeyState(HGEK\_LEFT))***

***dx-=speed\*dt ;***

***if(hge->Input\_GetKeyState(HGEK\_RIGHT))***

***dx+=speed\*dt ;***

***if(hge->Input\_GetKeyState(HGEK\_UP))***

***dy-=speed\*dt ;***

***if(hge->Input\_GetKeyState(HGEK\_DOWN))***

***dy+=speed\*dt ;***

***//做一些运动的计算***

***dx\*=friction;***

***dy\*=friction;***

***x+=dx;***

***y+=dy;***

***if(x>784)***

***{***

***x=784-(x-784);***

***dx=-dx;***

***boom();***

***}***

***if(x<16)***

***{***

***x=16+16-x;***

***dx=-dx;***

***boom();***

***}***

***if(y>584)***

***{***

***y=584-(y-584);***

***dy=-dy;***

***boom();***

***}***

***if(y<16)***

***{***

***y=16+16-y;***

***dy=-dy;***

***boom();***

***}***

***//配置hgeQuad结构中的顶点***

***quad.v[0].x=x-16;***

***quad.v[0].y=y-16;***

***quad.v[1].x=x+16;***

***quad.v[1].y=y-16;***

***quad.v[2].x=x+16;***

***quad.v[2].y=y+16;***

***quad.v[3].x=x-16;***

***quad.v[3].y=y+16;***

***return false ;***

***}***

Q:什么是帧回调函数？有什么用处？

A：这个帧回调函数是用来表示游戏逻辑的地方，会在后面的代码中设置这个函数被调用的频率，一般以每秒多少次表示。在这个函数中，我们就可以进行游戏逻辑的编写，注意此函数在返回true的情况下，游戏的逻辑将结束，而在返回false的情况下只是代表此次的函数调用结束了，并不是游戏的逻辑结束了(以后不会再被调用，称为逻辑结束)。

Q:这里的计算问题是什么进行计算的？

A:通过分析代码发现了这里的计算是以圆球图案的中心来进行计算的，以全局变量x,y表示圆球图案中心的坐标。然后使用dx,和dy表示在一次帧回调函数调用中，圆球图案中心移动的距离。当明白了这点之后，剩下的计算部分就比较简单了。以dx,dy表示圆球中心坐标的改变，正的代表向右或下进行移动，负的代表向左和右进行移动。dt表示离上次调用的时间间隔，根据设置的FPS值，计算出这个值应该是1/100=0.01秒，调试发现dt值确实等于这个。而在我们按下了一个或者多个按键的时候，就可以通过speed\*dt来修正dx，dy的值。注意这里的一个比较巧妙的设计，作者在定义全句变量的时候定义了一个名为friction(摩擦力)的全局变量，由于这个变量小于1，所以使用dx\*=friction,dy\*=friction，来进行计算的话，在dx，dy保持不变的情况下就会慢慢的变小，直至变为0，这就导致了游戏中圆球团在后来慢慢的变成的静止不动的效果。至于碰撞的处理就根据游戏界面的大小简单的使用返回距离来表示碰撞。

**//定义重绘函数，每次绘图时，调用此函数**

**bool ReDraw()**

**{**

**hge->Gfx\_BeginScene();  //开始绘图**

**hge->Gfx\_Clear(0);                    //清空背景**

**hge->Gfx\_RenderQuad(&quad); //重绘图片**

**hge->Gfx\_EndScene();    //结束绘图**

**return false ;**

**}**

Q:重绘函数的作用是什么？

A：重绘函数是在窗口需要重绘的时候进行调用的。和帧回调函数差不多，也是一种回调函数。在这个函数里面通过调用Gfx\_BeginScene来开始绘制，使用Gfx\_RenderQuad()函数来绘制一个hgeQuad结构中的物体。Gfx\_Clear(0)函数可以清空游戏的背景，然后进行重绘。Gfx\_EndScene函数会结束绘制，并且更新窗口。

Q:Gfx\_Clear是带参数的函数，它有哪些参数？分别表示什么？

A：Gfx\_Clear函数中带一个DWORD的参数，其中表示的是用指定的ARGB颜色来清除窗口，如果参数为0就表示用黑色来清除窗口。

**//定义WinMain函数**

**int WINAPI WinMain(HINSTANCE,HINSTANCE,LPSTR,int)**

**{**

**hge = hgeCreate(HGE\_VERSION); //创建HGE**

**//设置状态属性**

**hge->System\_SetState(HGE\_FRAMEFUNC,FrameFunc);              //设置帧回调函数**

**hge->System\_SetState(HGE\_WINDOWED,true);                    //设置是否全屏**

**hge->System\_SetState(HGE\_USESOUND,true);                     //是否使用音乐**

Tip:注意这里，如果我们要加载音效的话，这里就需要设置为true

**hge->System\_SetState(HGE\_TITLE,"My First HGE Window -by XJ"); //设置标题栏**

**hge->System\_SetState(HGE\_LOGFILE,"log.txt");          //设置日志文件**

**hge->System\_SetState(HGE\_RENDERFUNC,ReDraw);          //设置重绘函数**

**hge->System\_SetState(HGE\_FPS,100);                                  //设置每秒FrameFunc调用的次数**

**hge->System\_SetState(HGE\_SCREENWIDTH,800);               //设置屏幕宽度**

**hge->System\_SetState(HGE\_SCREENHEIGHT,600);                    //设置屏幕高度**

**hge->System\_SetState(HGE\_SCREENBPP,32);                      //设置像素位数为32bit/pix**

**if(hge->System\_Initiate()) //初始化hge**

**{**

**snd = hge->Effect\_Load("menu.wav");                                 //加载音效**

**quad.tex =hge->Texture\_Load("particles.png");              //加载纹理图片进行贴图**

Q:为什么要在这里加载？在其他地方可以不？

A:经过试验表明，如果在没有调用System\_Initiate函数之前调用资源加载的函数，调用会失败，这里需要在以后注意，只有在初始化之后，我们才能够调用加载资源的函数。

**quad.blend = BLEND\_ALPHAADD | BLEND\_COLORMUL | BLEND\_ZWRITE ; //设置混色模式**

Q:混色到底要如何使用？混色的参数要如何设置才能达到自己想要的效果？

A:？？？

**if(!snd || !quad.tex)**

**{**

**// If one of the data files is not found, display**

**// an error message and shutdown.**

**MessageBox(NULL, "Can't load MENU.WAV or PARTICLES.PNG", "Error", MB\_OK | MB\_ICONERROR | MB\_APPLMODAL);**

**hge->System\_Shutdown();**

**hge->Release();**

**return 0;**

**}**

**for(int i = 0 ; i < 4 ; i ++)**

**{**

**quad.v[i].z = 0.5f ;  //设置Z序**

**quad.v[i].col = 0xFFFFA000 ;  //设置顶点颜色**

**}**

Q:什么是顶点的Z序？

           A:Z序就是从屏幕里到外的一个轴，一般用来显示立体的效果时需要使用它。而在上面由于同一的都是0.5所以没有看出什么效果，如果修改下，将这四个顶点的Z序改变成不一的，看看情况如何？通过实验发现，不管将Z序的值设置为多少，对于游戏的显示来说，没有任何改变。查询官网上的资料，说了如果Z-BUFFER没有启动的话，Z-order的值会被忽略掉，这里可能就是因为没有启用Z-BUFFER。

       Q:如何使用ARGB颜色？前面的A值表示什么？

       A：前面的A值表示的是颜色的透明度是多少。这个透明度是用来在混合的时候发生作用的，如果不进行混合，那么指定任何值都没有关系。但是一旦进行混合，我们就需要注意这个A值所代表的意思。白色（FF）表示的是当前这个颜色没有透明度，而黑色（00）表示的是当前这个衍射全透明，而介于两者之间的就是在混合的时候，将这种颜色设置为半透明状态。而在进行混合时A值起到什么作用一般根据混合的模式不同，A的作用也会不同。

**//设置纹理的位置**

**quad.v[0].tx=96.0/128.0 ; quad.v[0].ty=64.0/128.0  ;**

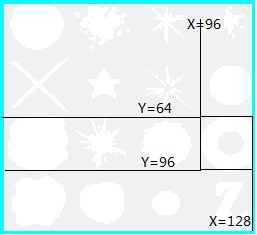
**quad.v[1].tx=128.0/128.0; quad.v[1].ty=64.0/128.0  ;**

**quad.v[2].tx=128.0/128.0; quad.v[2].ty=96.0/128.0  ;**

**quad.v[3].tx=96.0/128.0 ; quad.v[3].ty=96.0/128.0  ;**

Q:纹理的坐标是什么意思？为什么要这么计算？

       A:在hgeVertex结构中含有两个float参数，表示的是纹理的坐标，范围在0.0-1.0之间。要想了解如何定位这个纹理坐标我们来看下资源文件中的particles.png图片。



从图片中我们可以看出在资源文件中存在着很多的图案，为什么此实例中没有看到其他的图案了？答案就在于顶点结构中的两个纹理坐标。

我们知道hgeQuad中加载了一个纹理，同时保存了4点的hgeVertex结构。接下来我们看看hgeVertex结构

struct hgeVertex

{

       float x , y  ;    //顶点坐标

       float z ;          //Z序坐标

       DWORD col ;   //顶点颜色

       float tx , ty ;    //顶点在纹理上的坐标

};

注意看最后两个参数，tx,ty。它们表示的是所定义的顶点在纹理中的坐标。即这个顶点与纹理图片中的哪个点相对应，而第一，二个参数的意思是这个顶点在显示区域的位置坐标。比如上面代码中设置的纹理坐标，从图片中分析可以知道v[0]与纹理中的(96,64)绑定在一起，v[1]与纹理中的(128,64)绑定在一起，v[2]与纹理中的(128,96)绑定在一起，v[3]与纹理中的(96,96)绑定在一起。由于纹理坐标的范围是在0.0-1.0之间，所以这些坐标点分别除上了像图片的宽度或者高度128，来计算比例。这样四个点就分别对应于纹理中的包含圆球图案的矩形的四个点，也就是说在调用Gfx\_RenderQuad的时候，就会根据顶点结构中的x,y坐标绘制这个4个顶点，然后在贴上这四个顶点所绑定的图案，这就导致了为什么在程序中引入的是整张图片，而在游戏中只出现了一个圆球图案的情况。细心的读者可能注意到了，在定义纹理坐标时，这四个顶点的顺序不是任意的，而是按照顺时针旋转的方向来定义这四个顶点的，这里需要注意。

**hge->System\_Start(); //开始帧回调函数的执行**

Q:   为什么调用System\_Start之后就将资源释放掉？

A:   在官网有这样一段话，“Now we're ready to start the game loop with the System\_Start function. When the frame function returns **true** and the game loop is finished we should free the loaded texture and the sound effect:”

这句话的意思是说我们调用System\_Start来开始游戏的循环。循环包括图形图像的绘制，音效的播放，游戏的逻辑和按键的判断等等。当帧回调函数返回true的是偶，System\_Start才会返回。所以System\_Start返回了就代表游戏结束了，我们需要将游戏中使用的全部资源都释放掉。

**hge->Texture\_Free(quad.tex);  //释放纹理资源**

**hge->Effect\_Free(snd);          //释放音效资源**

**}**

**else**

**{**

**MessageBox(NULL,hge->System\_GetErrorMessage(),"Error",MB\_OK | MB\_ICONERROR);**

**}**

**hge->System\_Shutdown();**

Tip:注意先要关闭，然后在释放。

**hge->Release();**

**return 0 ;**

**}**

Q:关于Z序Z-BUFFER的开启有什么作用？如何使用？

A：这个问题，期待官网上后续的教程可以有效的演示如何进行操作。

以下是程序的完整代码：

#include<hge.h>

#pragma comment(lib,"hge.lib")

//定义全局变量

HGE \* hge = NULL ;

hgeQuad quad ;   //用于保存绘图的顶点，纹理，和混色的函数

HEFFECT snd ;     //保存音效的句柄

float x = 100.0f , y = 100.0f ;

float dx = 0.0f , dy = 0.0f ;

const float speed = 90 ;

const float friction = 0.98f ;

//定义播放音效的函数

void boom()

{

       int pan = int((x-400)/4);

       float pitch = (dx\*dx+dy\*dy)\*0.0005f + 0.2f ;

       hge->Effect\_PlayEx(snd,100,pan,pitch); //snd是音频资源的句柄，100表示声音大小，pan表示左右声道，pitch表示播放频率

}

//定义帧回调函数

bool FrameFunc()

{

       float dt = hge->Timer\_GetDelta();   //获取自上次调用此函数到此次调用此函数的时间间隔

       if(hge->Input\_GetKeyState(HGEK\_ESCAPE))

              return true ;

       //根据不同的按键，使用速度和延迟时间来修正位置

       if(hge->Input\_GetKeyState(HGEK\_LEFT))

              dx-=speed\*dt ;

       if(hge->Input\_GetKeyState(HGEK\_RIGHT))

              dx+=speed\*dt ;

       if(hge->Input\_GetKeyState(HGEK\_UP))

              dy-=speed\*dt ;

       if(hge->Input\_GetKeyState(HGEK\_DOWN))

              dy+=speed\*dt ;

       //做一些运动的计算

       dx\*=friction;

       dy\*=friction;

       x+=dx;

       y+=dy;

       if(x>784)

       {

              x=784-(x-784);

              dx=-dx;

              boom();

       }

       if(x<16)

       {

              x=16+16-x;

              dx=-dx;

              boom();

       }

       if(y>584)

       {

              y=584-(y-584);

              dy=-dy;

              boom();

       }

       if(y<16)

       {

              y=16+16-y;

              dy=-dy;

              boom();

       }

       //配置hgeQuad结构中的顶点

       quad.v[0].x=x-16;

       quad.v[0].y=y-16;

       quad.v[1].x=x+16;

       quad.v[1].y=y-16;

       quad.v[2].x=x+16;

       quad.v[2].y=y+16;

       quad.v[3].x=x-16;

       quad.v[3].y=y+16;

       return false ;

}

//定义重绘函数，每次绘图时，调用此函数

bool ReDraw()

{

       hge->Gfx\_BeginScene();  //开始绘图

       hge->Gfx\_Clear(0);                     //清空背景

       hge->Gfx\_RenderQuad(&quad); //重绘图片

       hge->Gfx\_EndScene();    //结束绘图

       return false ;

}

//定义WinMain函数

int WINAPI WinMain(HINSTANCE,HINSTANCE,LPSTR,int)

{

       hge = hgeCreate(HGE\_VERSION); //创建HGE

       //设置状态属性

       hge->System\_SetState(HGE\_FRAMEFUNC,FrameFunc);         //设置帧回调函数

       hge->System\_SetState(HGE\_WINDOWED,true);                    //设置是否全屏

       hge->System\_SetState(HGE\_USESOUND,true);               //是否使用音乐

       hge->System\_SetState(HGE\_TITLE,"My First HGE Window -by XJ"); //设置标题栏

       hge->System\_SetState(HGE\_LOGFILE,"log.txt");            //设置日志文件

       hge->System\_SetState(HGE\_RENDERFUNC,ReDraw);           //设置重绘函数

       hge->System\_SetState(HGE\_FPS,100);                                   //设置每秒FrameFunc调用的次数

       hge->System\_SetState(HGE\_SCREENWIDTH,800);                //设置屏幕宽度

       hge->System\_SetState(HGE\_SCREENHEIGHT,600);                     //设置屏幕高度

       hge->System\_SetState(HGE\_SCREENBPP,32);                       //设置像素位数为32bit/pix

       if(hge->System\_Initiate()) //初始化hge

       {

              snd = hge->Effect\_Load("menu.wav");                             //加载音效

              quad.tex =hge->Texture\_Load("particles.png");          //加载纹理图片进行贴图

              quad.blend = BLEND\_ALPHAADD | BLEND\_COLORMUL | BLEND\_ZWRITE ; //设置混色模式

              if(!snd || !quad.tex)

              {

                            // If one of the data files is not found, display

                            // an error message and shutdown.

                     MessageBox(NULL, "Can't load MENU.WAV or PARTICLES.PNG", "Error", MB\_OK | MB\_ICONERROR | MB\_APPLMODAL);

                     hge->System\_Shutdown();

                     hge->Release();

                     return 0;

              }

              for(int i = 0 ; i < 4 ; i ++)

              {

                     quad.v[i].z = 0.5f ;  //设置Z序

                     quad.v[i].col = 0xFFFFA000 ;  //设置顶点颜色

              }

              //设置纹理坐标的位置

              quad.v[0].tx=96.0/128.0 ; quad.v[0].ty=64.0/128.0  ;

              quad.v[1].tx=128.0/128.0; quad.v[1].ty=64.0/128.0  ;

              quad.v[2].tx=128.0/128.0; quad.v[2].ty=96.0/128.0  ;

              quad.v[3].tx=96.0/128.0 ; quad.v[3].ty=96.0/128.0  ;

              hge->System\_Start(); //开始帧回调函数的执行

              hge->Texture\_Free(quad.tex);  //释放纹理资源

              hge->Effect\_Free(snd);           //释放音效资源

       }

       else

       {

              MessageBox(NULL,hge->System\_GetErrorMessage(),"Error",MB\_OK | MB\_ICONERROR);

       }

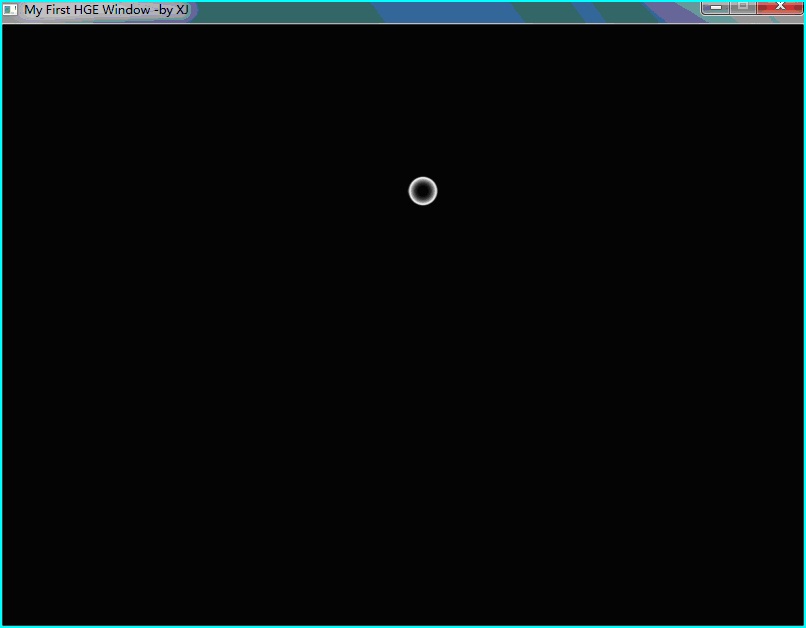
       hge->System\_Shutdown();

       hge->Release();

       return 0 ;

}

附上贴图：



https://blog.csdn.net/i\_dovelemon/article/details/8818037