基于 openGL 的虚拟教室绘制

——虚拟现实与数据可视化课程作业

姓名：张琦

学号：22012206

专业：测控技术与仪器

指导老师：孙立博老师

一． 作业要求

用 openGL 绘制一个虚拟教室场景。要求实现视角可变化以及漫游功能，教室里的物品不少于五件。

二．整体思路分析

运用 openGL 图形库函数，基于 visualc++6.0 平台绘制一个虚拟教室的步骤大概如下：

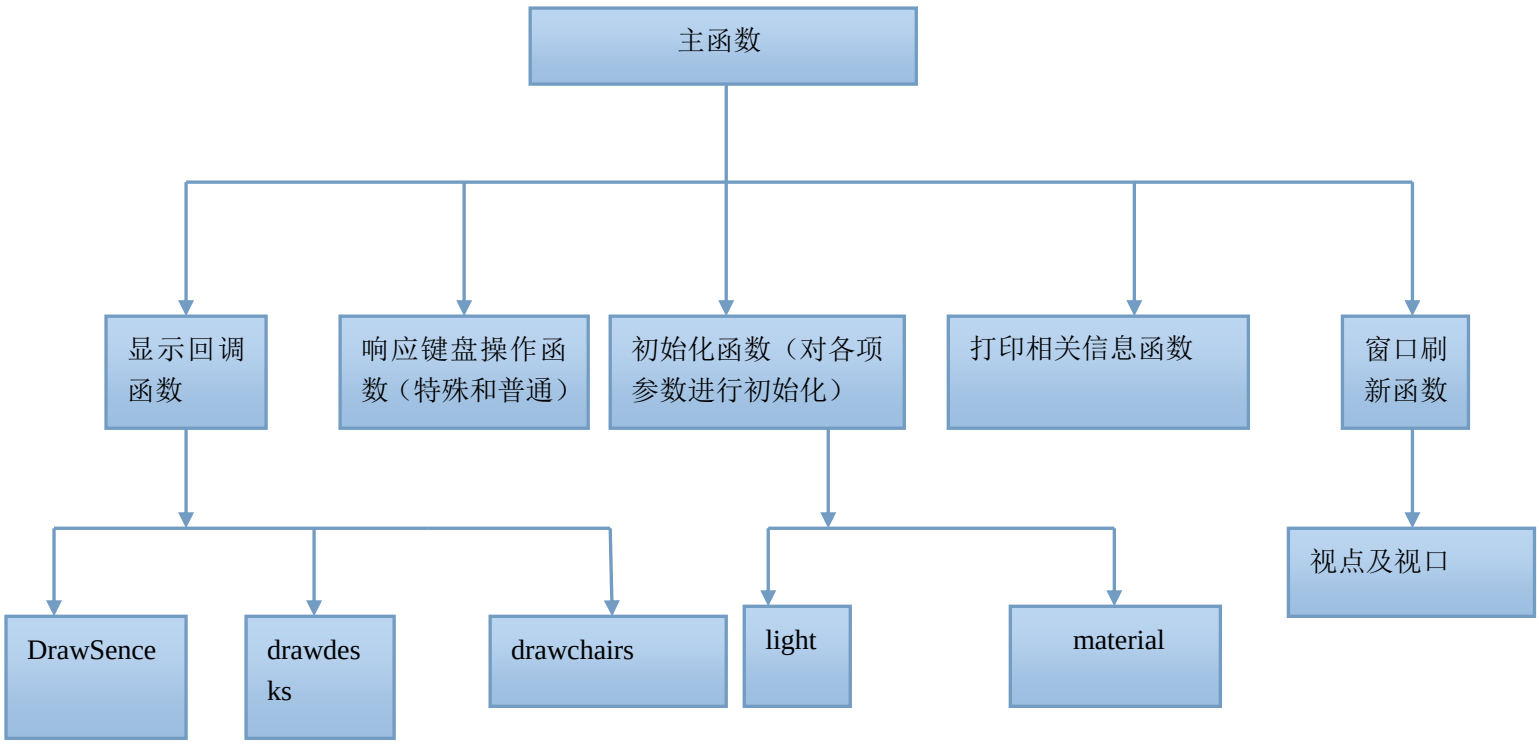
1. 首先建立起一个教室的整体框架，即画出一个空间坐标系中的长方体，当然四个面均可使用 GL_QUADS 来实现，只要定下教室的大小以及在空间坐标系中的位置，即可定下八个顶点坐标。
   这是我绘制教室这个大场景的部分代码截图：

```
//自无绘制大化板
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, texceili
glColor3f(0.3,0.3,0.3);
glBegin(GL_QUADS);
glNormal3f(0.0f, -1.0f, 0.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(-40.0f,30.0f, 30.0f);
glTexCoord2f(0.0f, 3.0f);
glVertex3f(-40.0f, 30.0f, -30.0f);
glTexCoord2f(6.0f, 3.0f);
glVertex3f(40.0f, 30.0f, -30.0f);
glTexCoord2f(6.0f, 0.0f);
glVertex3f(40.0f, 30.0f, 30.0f);
glEnd();
glDisable(GL_TEXTURE_2D);
//绘制地板
glColor3f(0.8f, 1.0f, 0.8f);
glBegin(GL_QUADS);
glNormal3f(0.0f, 1.0f, 0.0f);
glVertex3f(-40.0f,0.0f, 30.0f);
glVertex3f(-40.0f, 0.0f, -30.0f);
glVertex3f(40.0f, 0.0f, -30.0f);
glVertex3f(40.0f, 0.0f, 30.0f);
glEnd();
```

2. 选择一个合适的视点，我们看物体时，我们的眼睛在什么位置，我们看向什么位置均会影响看到的效果，选取合适的“眼睛”坐标和视点坐标，利用 gluLookAt 函数，来使整个教室呈现在观察者的视野当中。键盘控制改变视角只需加入一个键盘响应函数并在其中添加视点相关变量的增加和减少即可实现。比如：gluLookAt(myEye.x, myEye.y, myEye.z, vPoint.x+30*sin(vAngle), vPoint.y,-30*cos(vAngle), 0.0f, 1.0f, 0.0f);当然这是加入了改变视角和巡游的功能。

3.具体绘制部分：绘制其他处于这个长方体中的物体，桌子，椅子，讲台，空调等，考虑纹理贴图以及设置灯光和材质。绘制桌椅均使用 openGL 中的绘制简单几何体的函数，glBegin(GL_QUADS)以及 glutSolidCube(1.0f)和一些矩阵变化平移缩放来实现。纹理贴图部分参考网上代码使用一个载入位图返回纹理编号的函数 int load_texture（char*filename）来实现。

三．程序框图（部分主要）



四．总结与评价

1）技术总结：

1.初次接触 openGL 这一图形函数库确实有很多地方需要注意。首先就是视点的问题，刚开始绘制时，在画完了四面墙以及天花板和地板之后，运行出来的窗口中只有两块重叠的白板，仔细思考以及调试程序之后才发现是视点的问题。通过查阅资料给 gluLookAt(GLdouble eyex,GLdouble eyey,GLdouble eyez,GLdouble centerx,GLdouble centery,GLdouble centerz,GLdouble upx,GLdouble upy,GLdouble upz)这个函数赋值了正确合适的参数，才在窗口中看到了一个教室的大概轮廓。但是设置对视点之后发现窗口出现了变形，查阅资料发现是因为视口比例不对，视口比例要和所绘制的教室的纵横比相同，调用 glViewport(0,0,(GLsizei) we, (GLsizei) he)函数即可实现视口变换。

2.在编写响应键盘操作的函数时，相关变量要不断修改并持续变化，一定要在响应函数中要再次调用窗口刷新函数和再显示函数

```
reshape(WinWidth,WinHeight);
glutPostRedisplay();
```

才能实现程序响应连续键盘操作的功能。

3 进行纹理贴图时，发现在已经贴过图的表面上再次贴图就会使其受遮挡，想要达到预期效果我投机取巧采用的坐标近似的办法，比如在墙壁上绘制窗户，左墙壁的 x 左边已经是-40，那么窗户的 x 左边用-39.9 来近似，这样并不会影响纹理的效果，也能使得两幅贴图同时展现出来。

2）心得体会

这次使用 openGL 图形函数库来绘制虚拟教室场景让我重新对 c++的知识有了新的体会，这是我第一次编写大规模的代码（代码已经快到 1000 行），出错的时候排查都不好排查。所以我借鉴学习别人的程序，尽量使用封装函数，函数调用函数，这也是 c++语言的特点。让自己的代码看起来更加有层次感，条理也更加清楚，我想以后再借助于 c++的平台来学习实践其他函数库以完成更加复杂功能时我会入门的更快更得心应手。

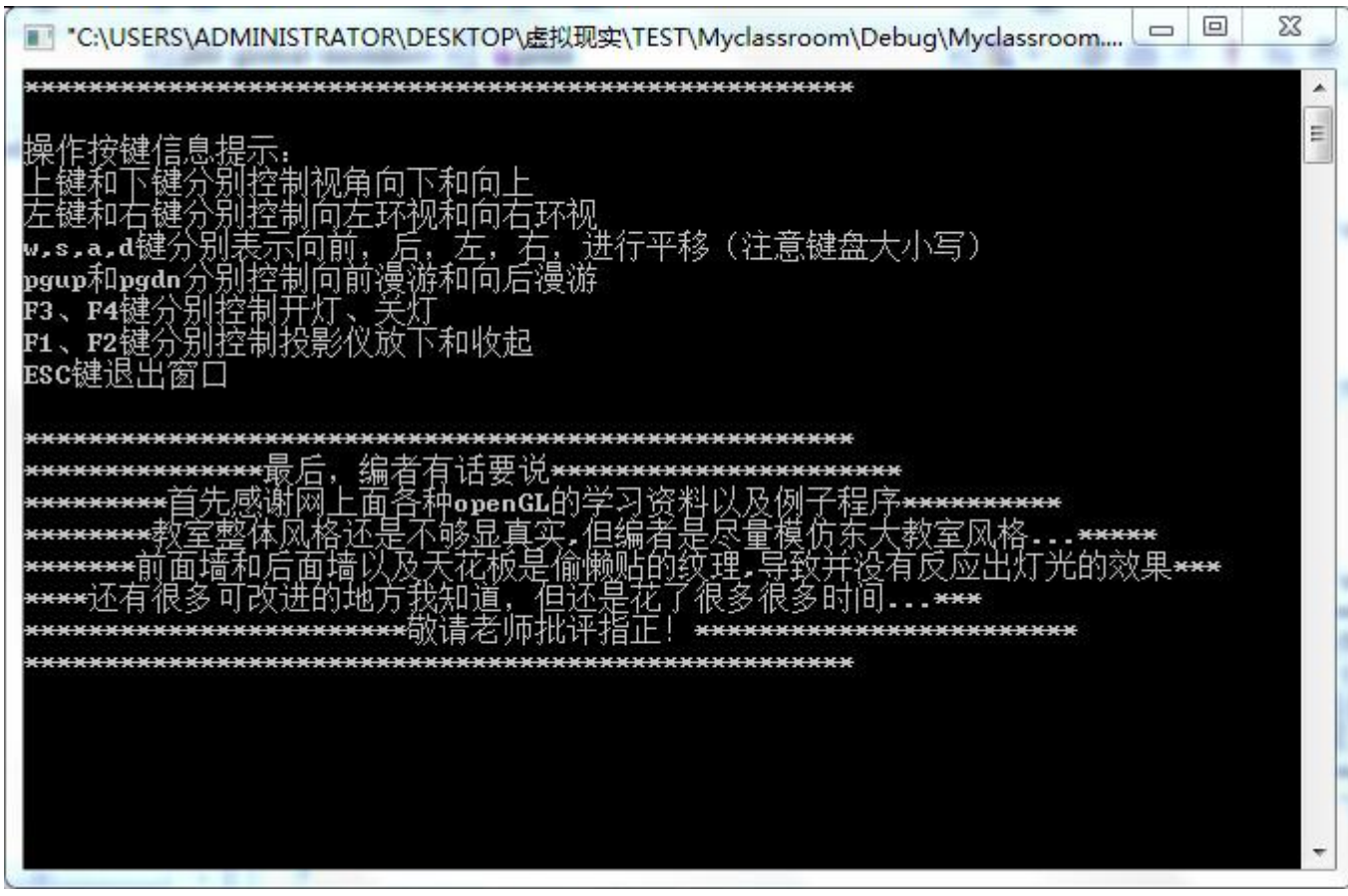
openGL 我以前并没有过接触，除了仔细消化老师上课时介绍的知识，课后我上网查阅了大量资料以及例子程序，在这个自我学习的过程中，我深刻体会到接触任何一种语言或者函数库都需要抱着一种从头从基础学起，不放过任何细节的心态，才能真正的入门。

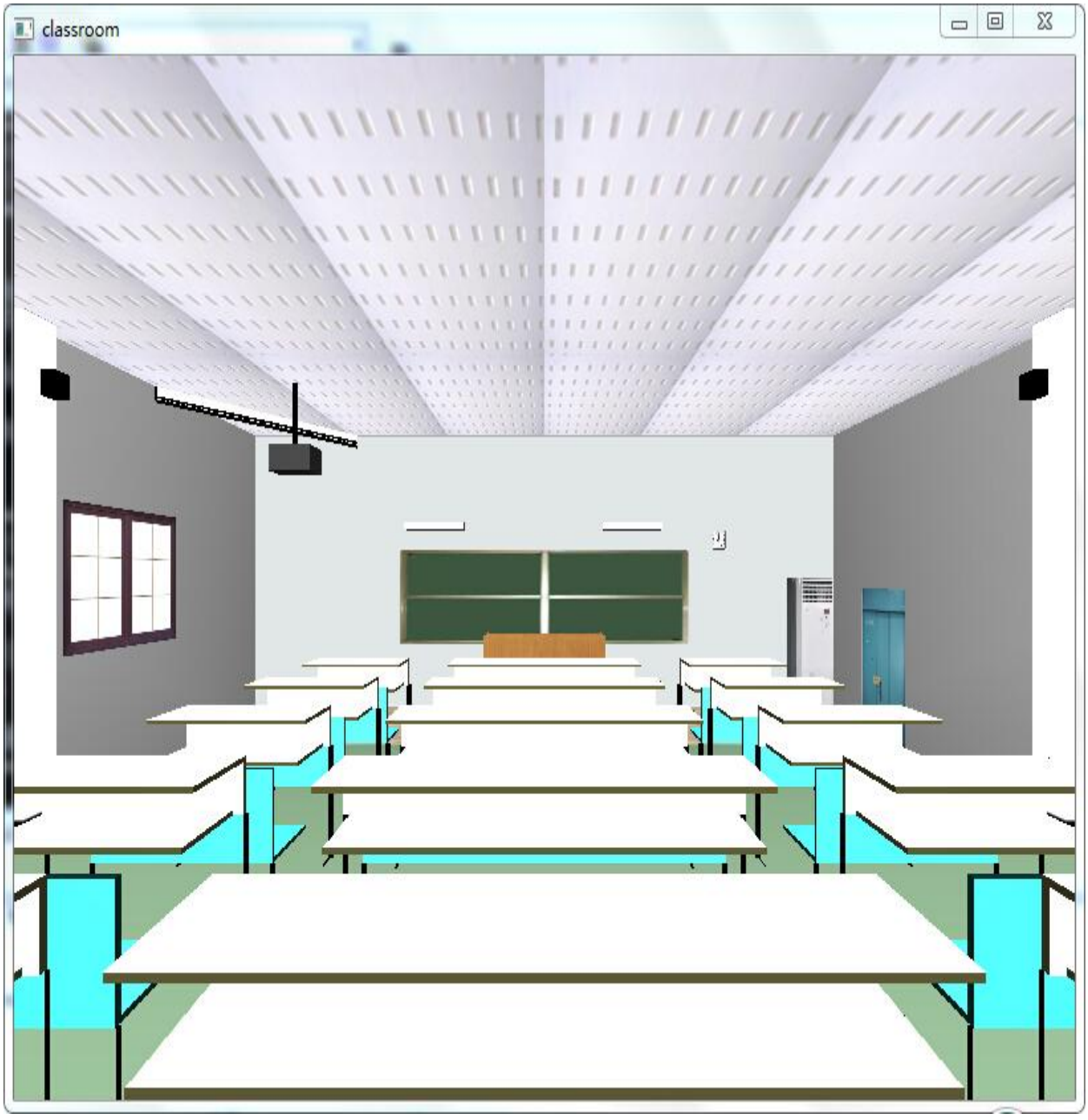
五． 完成情况

1、 画出了教室里的基本物体：黑板、讲台、投影仪、空调、门窗、音响、五排三列桌子、五排三列凳子，挂钟，挂灯等。整体风格参照东大教室风格。
2、 添加了纹理和灯光效果，门窗，空调，挂钟，黑板均使用纹理贴图。使得整个教室更加形象生动具有真实感。
3、 灯光具有键盘可控性，对比开灯关灯效果。投影仪和收起放下效果，以及一些其他实现转换视角和巡游的按键操作，可实现 360 度环视，左右平移，前进后退，并且对巡游范围进行了限制，只能在教室这个大场景中进行巡游。
4、 在输出窗口添加了按键提示信息，使界面更加友好。

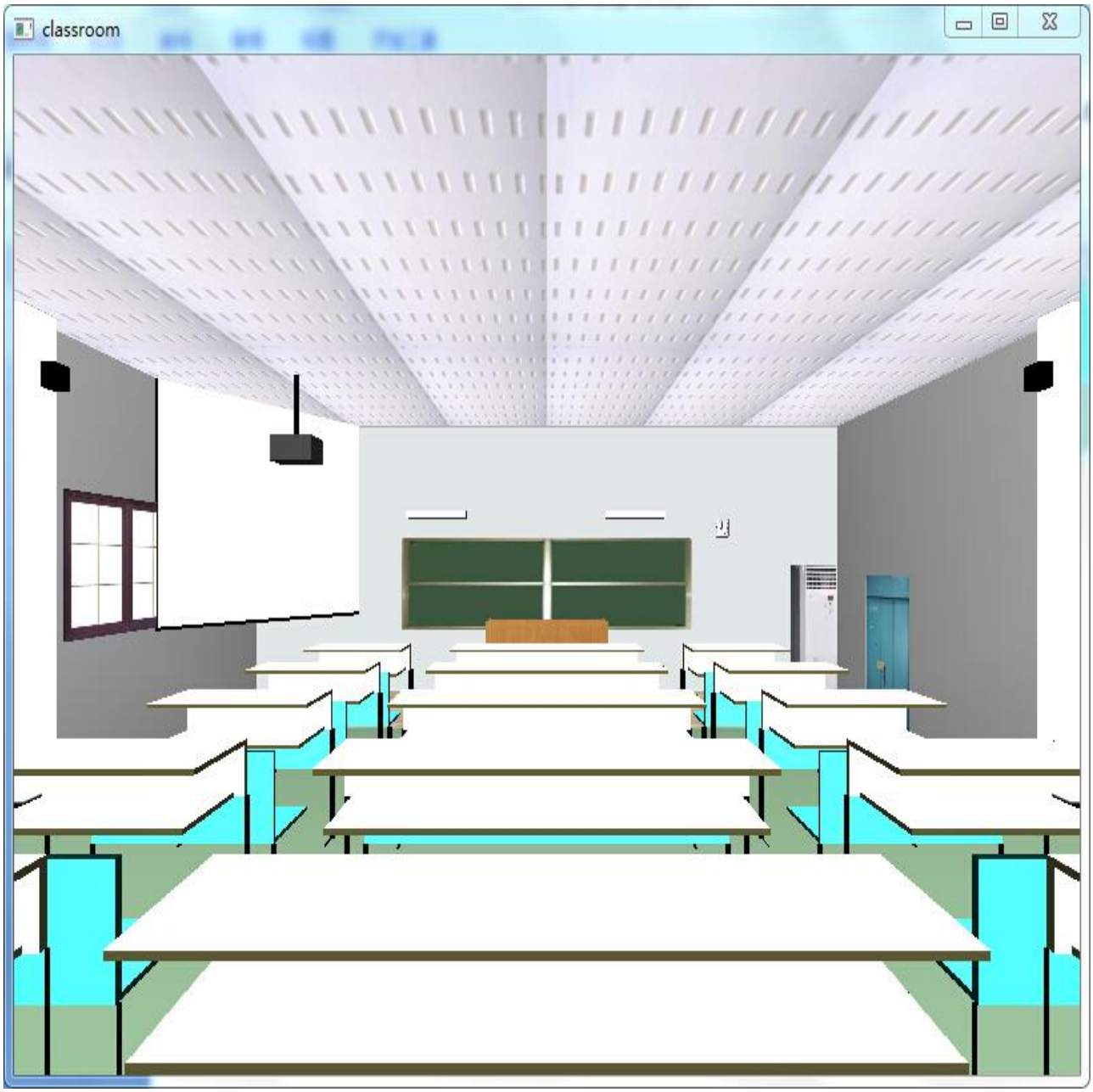
六. 结果展示

初始窗口截图：

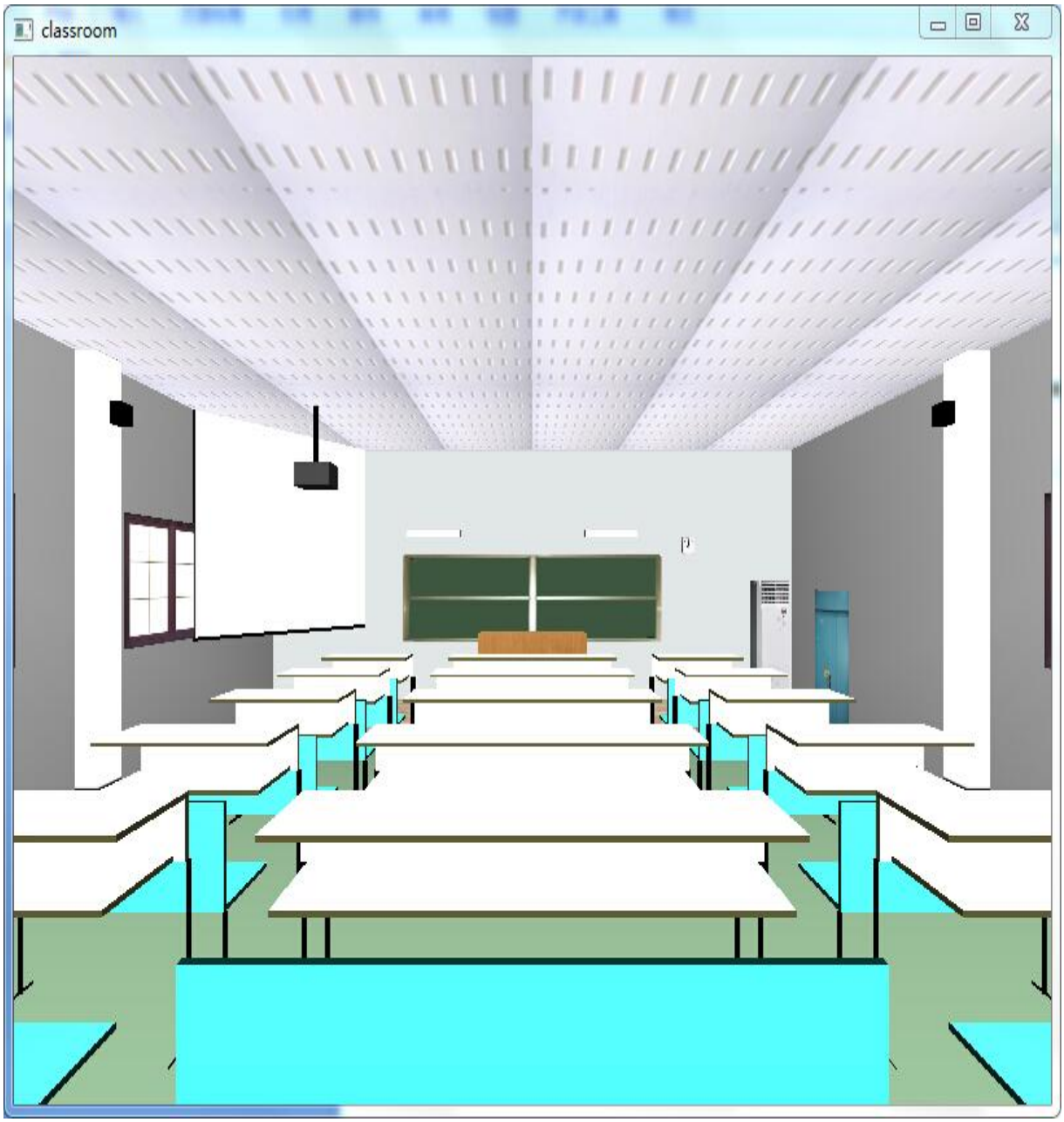




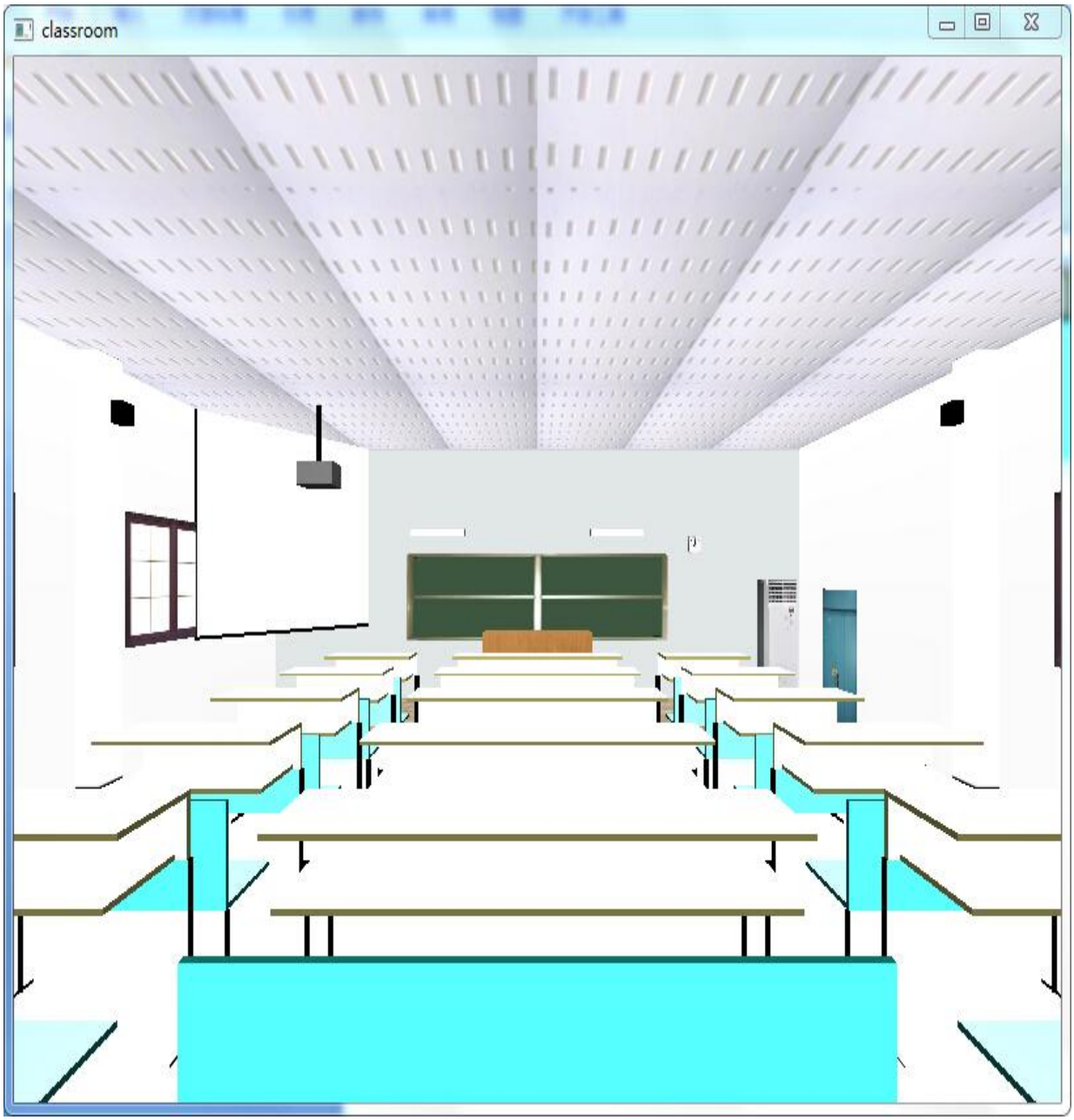
2.投影仪放下：



3.开灯关灯对比
未开灯时：

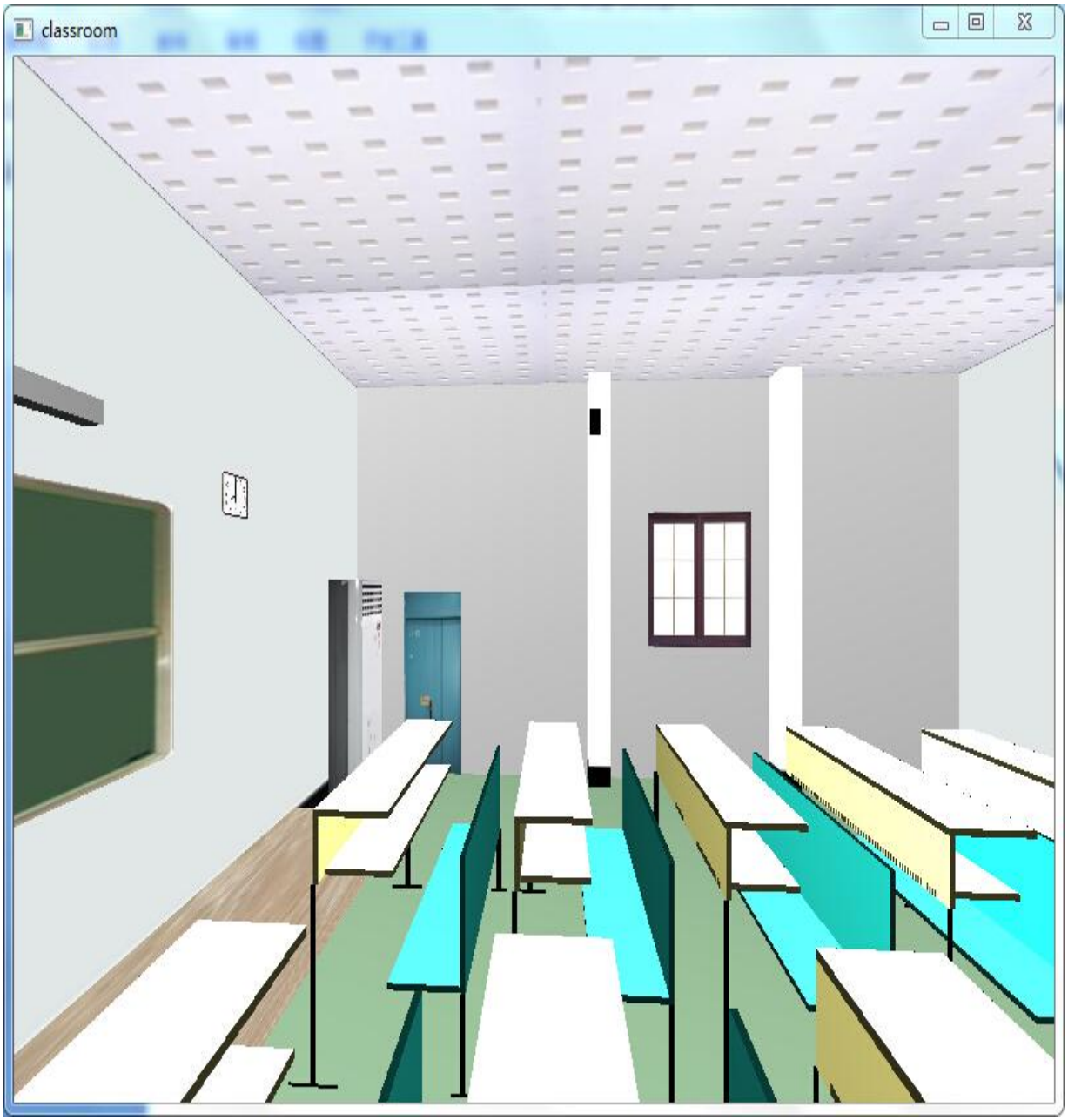


开灯时：

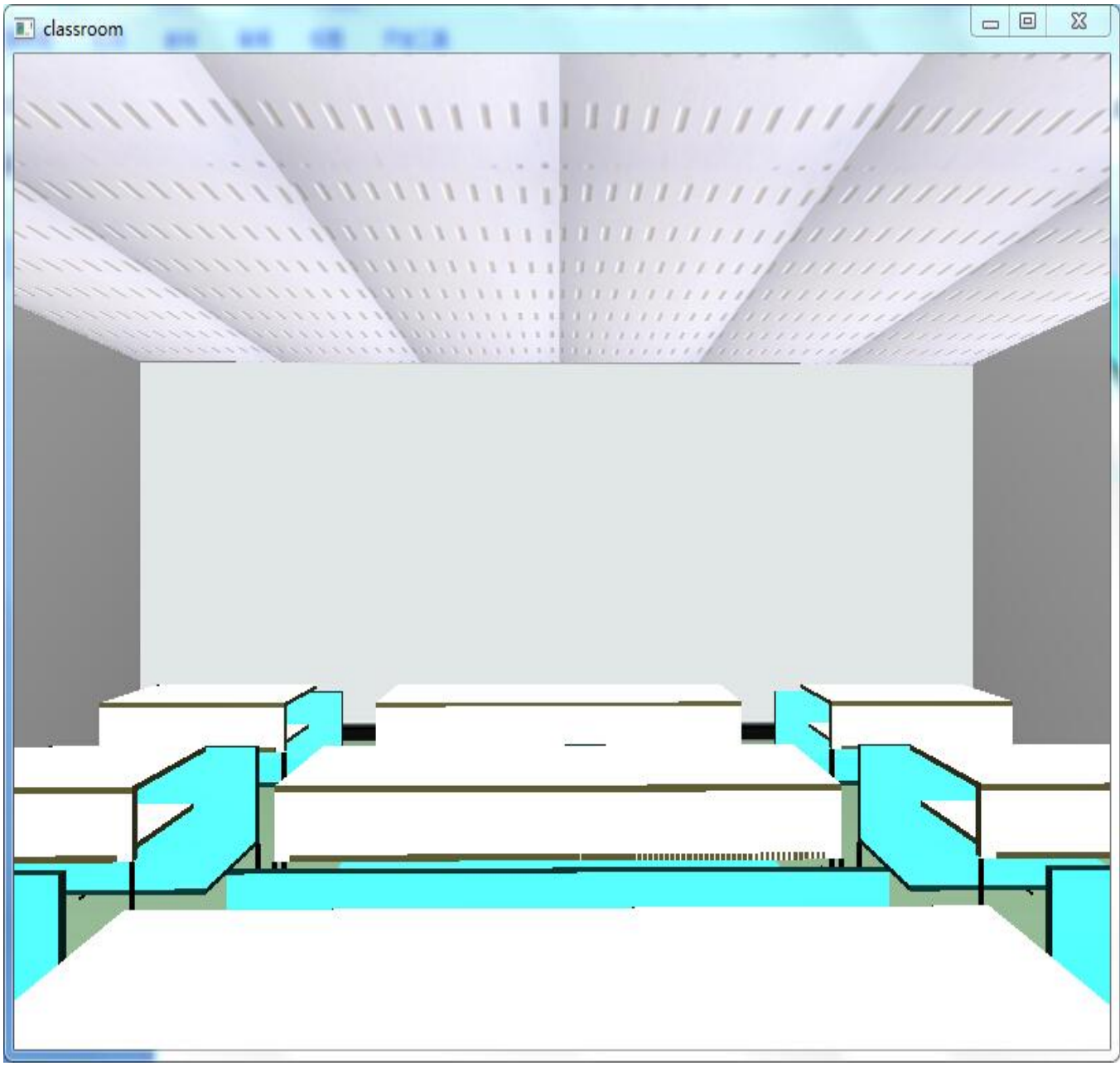


4.转化视角：

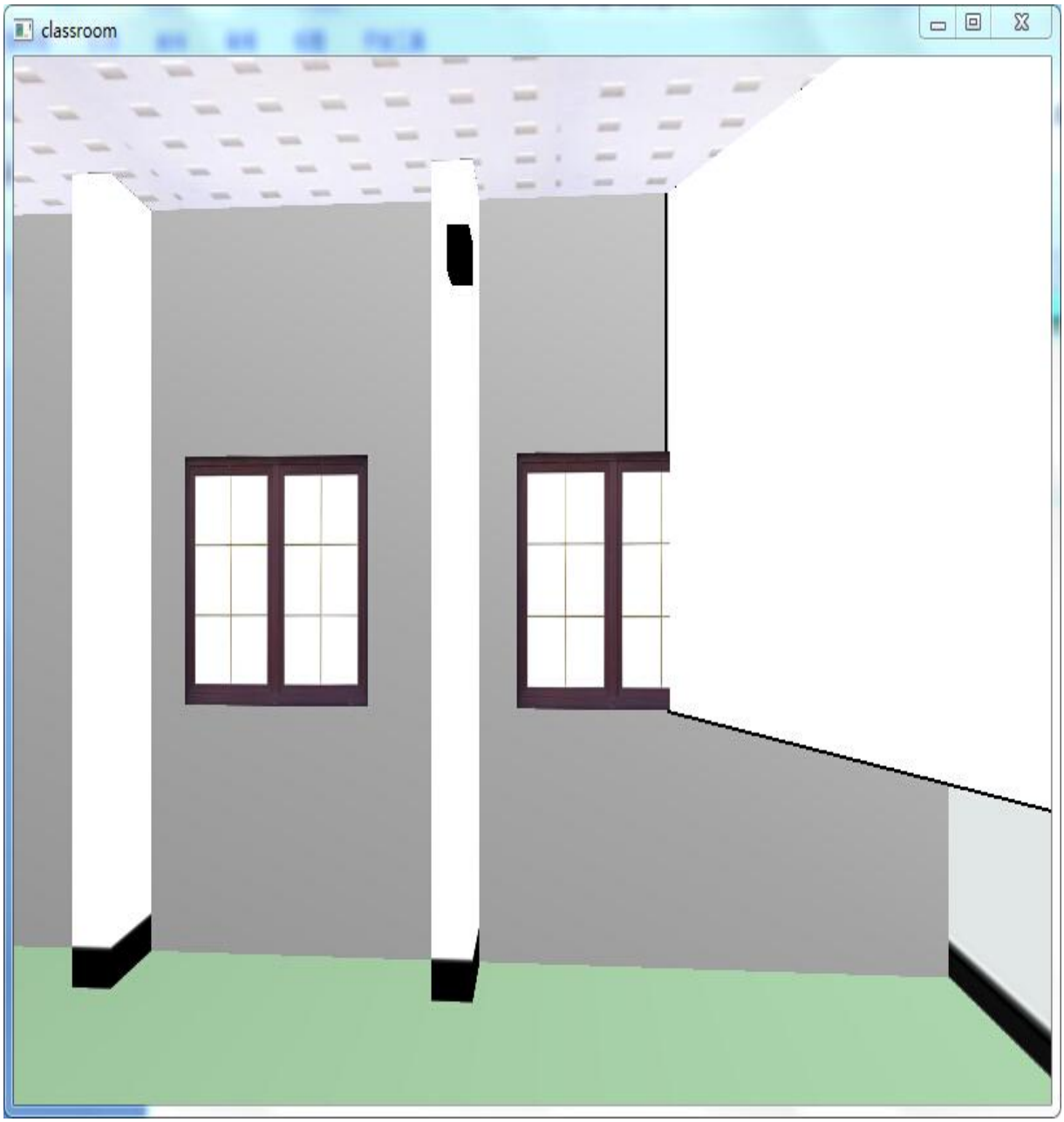
面向右边：



面向后面墙：



面朝左边墙：



其余细节部分请双击文件夹内 exe.

七．参考资料

1.openGL 编程指南，（美）Dave Shreiner 等著，王锐等，译/2014-10-01,机械工业出版社

2.http://wenku.baidu.com/link?url=sdsbS30L4fEv266ghka4gFlQN912BK3adb57-Qm-raFfmK7SNjQihSHOffAz8IoKnSVVhsN1rFX1nfThTNfm9JcwmaG8c4PEIwx1ixHJRZC

八．代码附录

```
#include<gl/glut.h>
#include<windows.h>
#include<math.h>
#include <stdio.h>
#include <stdlib.h>
/****************************** 定 义 程 序 中 所 用 的 常 数 变 量
```

```
**************************************/
    GLfloat light_position1[]={0,28,-20,1.0};
    GLfloat model_ambient[]={0.05f,0.05f,0.05f,1.0f};
    GLfloat mat_specular[]={0.8,1.0,1.0,1.0};
    GLfloat mat_shininess[]={5.0};
    GLfloat mat_ambient[]={0.1,0.1,0.1,1};
    GLfloat white_light[]={1.0,1.0,1.0,1.0};
    GLfloat light[]={1.0,1.0,1.0,1};
    GLfloat light_position0[]={0,28,20,1.0};
    GLfloat  no_mat[]= {0.0f, 0.0f, 0.0f, 1.0f};
    GLfloat  mat_diffuse1[]= {0.1f, 0.5f, 0.8f, 1.0f};
    GLfloat  no_shininess[]= {0.0f};
    GLfloat sound[]={0.9,0.9,0.9,1};
    GLint    WinWidth;
    GLint    WinHeight;

    /************************      定      义      视      点      结      构
**********************************************************/
    typedef struct EyePoint
    {
        GLfloat  x;
        GLfloat y;
        GLfloat z;
    }EyePoint;
    EyePoint     myEye;
    EyePoint       vPoint;
    GLfloat pro_up_down=29.0f;
    GLfloat vAngle=0;
    /*************************** 载 入 位 图 作 为 纹 理 的 相 关 函 数
***********************************/
    #define BMP_Header_Length 54
    void grab(void)

    {

        FILE* pDummyFile; FILE* pWritingFile;
        GLubyte* pPixelData;
        GLubyte BMP_Header[BMP_Header_Length];
        GLint       i, j;
        GLint       PixelDataLength;
    //  计算像素数据的实际长度

    i = WinWidth * 3;   //  得到每一行的像素数据长度
    while( i%4 != 0 )          //  补充数据，直到 i 是的倍数
```

```
        ++i;                                    // 本来还有更快的算法，
        PixelDataLength = i * WinHeight;// 但这里仅追求直观，对速度没有太高要
求
          pPixelData = (GLubyte*)malloc(PixelDataLength);// 分配内存和打开文件
          if( pPixelData == 0 )
              exit(0);
        pDummyFile = fopen("dummy.bmp", "rb");
        if( pDummyFile == 0 )
        exit(0);
        pWritingFile = fopen("grab.bmp", "wb");
        if( pWritingFile == 0 )
        exit(0);
        glPixelStorei(GL_UNPACK_ALIGNMENT, 4);
        glReadPixels(0, 0, WinWidth, WinHeight,GL_BGR_EXT, GL_UNSIGNED_BYTE,
pPixelData);
    // 把 dummy.bmp 的文件头复制为新文件的文件头

        fread(BMP_Header, sizeof(BMP_Header), 1, pDummyFile);
        fwrite(BMP_Header, sizeof(BMP_Header), 1, pWritingFile);
        fseek(pWritingFile, 0x0012, SEEK_SET);
        i = WinWidth;
        j = WinHeight;
        fwrite(&i, sizeof(i), 1, pWritingFile);
        fwrite(&j, sizeof(j), 1, pWritingFile);
        fseek(pWritingFile, 0, SEEK_END);
        fwrite(pPixelData, PixelDataLength, 1, pWritingFile);
        fclose(pDummyFile); fclose(pWritingFile); free(pPixelData);
    }

    //判断一个数是否是 2 的整数次方
    int power_of_two(int n)
    {
        if( n <= 0 )
            return 0;
         return (n & (n-1)) == 0;
    }
    /*****************************载入一副位图作为纹理，返回的是纹理编
号*********************************************/
    GLuint load_texture(const char* file_name)
    {
          GLint width, height, total_bytes;
        GLubyte* pixels = 0;
        GLint last_texture_ID=0;
        GLuint texture_ID = 0;
```

```
    // 打开文件，如果失败，返回
        FILE* pFile = fopen(file_name, "rb");
        if( pFile == 0 )
        return 0;
    // 读取文件中图象的宽度和高度
        fseek(pFile, 0x0012, SEEK_SET);
        fread(&width, 4, 1, pFile);
        fread(&height, 4, 1, pFile);
        fseek(pFile, BMP_Header_Length, SEEK_SET);
    // 计算每行像素所占字节数，并根据此数据计算总像素字节数
    {
        GLint line_bytes = width * 3;
        while( line_bytes % 4 != 0 )
        ++line_bytes;
        total_bytes = line_bytes * height;
    }
    // 根据总像素字节数分配内存
        pixels = (GLubyte*)malloc(total_bytes);
        if( pixels == 0 )
        {
            fclose(pFile);
            return 0;
        }
    // 读取像素数据
        if( fread(pixels, total_bytes, 1, pFile) <= 0 )
        {
            free(pixels);
            fclose(pFile);
            return 0;
        }
    // 在旧版本的 OpenGL 中
    // 如果图象的宽度和高度不是的整数次方，则需要进行缩放
    // 这里并没有检查 OpenGL 版本，出于对版本兼容性的考虑，按旧版本处
理
    // 另外，无论是旧版本还是新版本，
    // 当图象的宽度和高度超过当前 OpenGL 实现所支持的最大值时，也要进
行缩放
    {
        GLint max;
        glGetIntegerv(GL_MAX_TEXTURE_SIZE, &max);
        if( !power_of_two(width)|| !power_of_two(height)|| width > max||
height > max )
        {
            const GLint new_width = 256;
```

```
            const GLint new_height = 256; // 规定缩放后新的大小为边长的正方形
              GLint new_line_bytes, new_total_bytes;
            GLubyte* new_pixels = 0;
       // 计算每行需要的字节数和总字节数
            new_line_bytes = new_width * 3;
            while( new_line_bytes % 4 != 0 )
            ++new_line_bytes;
            new_total_bytes = new_line_bytes * new_height;
      // 分配内存
            new_pixels = (GLubyte*)malloc(new_total_bytes);
            if( new_pixels == 0 )
            {
                free(pixels);
                fclose(pFile);
                return 0;
            }
      // 进行像素缩放
            gluScaleImage(GL_RGB,width, height, GL_UNSIGNED_BYTE, pixels,new_width, new_height, GL_UNSIGNED_BYTE, new_pixels);
      // 释放原来的像素数据，把 pixels 指向新的像素数据，并重新设置 width 和 height
            free(pixels);
            pixels = new_pixels;
            width = new_width;
            height = new_height;
          }
      }
      // 分配一个新的纹理编号
            glGenTextures(1, &texture_ID);
            if( texture_ID == 0 )
            {
                free(pixels);
                fclose(pFile);
                return 0;
            }
      // 绑定新的纹理，载入纹理并设置纹理参数
      // 在绑定前，先获得原来绑定的纹理编号，以便在最后进行恢复
            glGetIntegerv(GL_TEXTURE_BINDING_2D, &last_texture_ID);
            glBindTexture(GL_TEXTURE_2D, texture_ID);
            glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
            glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```

```
            glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
            glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
            glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE);
            glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0,
GL_BGR_EXT, GL_UNSIGNED_BYTE, pixels);
            glBindTexture(GL_TEXTURE_2D, last_texture_ID);
    // 之前为 pixels 分配的内存可在使用 glTexImage2D 以后释放
    // 因为此时像素数据已经被 OpenGL 另行保存了一份（可能被保存到专门
的图形硬件中）
            free(pixels);
            return texture_ID;
    }
    /********************************** 定 义 各 个 纹 理 对 象 的 名 称
**********************************/
        GLuint texblackboard,texwindow,texdesk,texsound;
        GLuint texceiling,texdoor,texfloor,texbackwall,texpole;
        GLuint texairfro,texairback,texgaodi,texsdesk,texclock;
    /****************************** 绘 制 相 关 函 数
***********************************************/

    //绘制教室这个大场景
    void drawscence()
    {
        //设置材质相关参数
        glMaterialfv(GL_FRONT, GL_AMBIENT, no_mat);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse1);
        glMaterialfv(GL_FRONT, GL_SPECULAR, no_mat);
        glMaterialfv(GL_FRONT, GL_SHININESS, no_shininess);
        glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);

        //首先绘制天花板
        glEnable(GL_TEXTURE_2D);
        glBindTexture(GL_TEXTURE_2D, texceiling);
        glColor3f(0.3,0.3,0.3);
          glBegin(GL_QUADS);
          glNormal3f(0.0f, -1.0f, 0.0f);                    //用于定义法线向量
          glTexCoord2f(0.0f, 0.0f);
        glVertex3f(-40.0f,30.0f, 30.0f);
        glTexCoord2f(0.0f, 3.0f);
        glVertex3f(-40.0f, 30.0f, -30.0f);
        glTexCoord2f(6.0f, 3.0f);
        glVertex3f(40.0f, 30.0f, -30.0f);
        glTexCoord2f(6.0f, 0.0f);
        glVertex3f(40.0f, 30.0f, 30.0f);
```

```
glEnd();
glDisable(GL_TEXTURE_2D);
//绘制地板
 glColor3f(0.8f, 1.0f, 0.8f);
 glBegin(GL_QUADS);
 glNormal3f(0.0f, 1.0f, 0.0f);                    //用于定义法线向量
glVertex3f(-40.0f,0.0f, 30.0f);
glVertex3f(-40.0f, 0.0f, -30.0f);
glVertex3f(40.0f, 0.0f, -30.0f);
glVertex3f(40.0f, 0.0f, 30.0f);
glEnd();
//绘制左边墙加左边窗户
glColor3f(0.8f,0.8f, 0.8f);
glBegin(GL_QUADS);
 glNormal3f(1.0f, 0.0f, 0.0f);                    //用于定义法线向量
 glVertex3f(-40.0f,0.0f, 30.0f);
 glVertex3f(-40.0f, 30.0f, 30.0f);
 glVertex3f(-40.0f, 30.0f, -30.0f);
 glVertex3f(-40.0f, 0.0f, -30.0f);
 glEnd();
 glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, texwindow);
for(int n=0;n<=1;n++)
{
    glBegin(GL_QUADS);
      glNormal3f(1.0, 0.0f, 0.0f);                    //用于定义法线向
量
      glTexCoord2f(1.0f, 0.0f);glVertex3f(-39.9, 10, -8+n*18);
    glTexCoord2f(1.0f, 1.0f);glVertex3f(-39.9, 20, -8+n*18);
    glTexCoord2f(0.0f, 1.0f);glVertex3f(-39.9, 20, -18+n*18);
    glTexCoord2f(0.0f, 0.0f);glVertex3f(-39.9, 10, -18+n*18);
    glEnd();
    }
    glDisable(GL_TEXTURE_2D);
    //绘制右边墙加右边窗户
    glColor3f(0.8f,0.8f, 0.8f);
      glBegin(GL_QUADS);
      glNormal3f(-1.0f, 0.0f, 0.0f); //用于定义法线向量
    glVertex3f(40.0f,0.0f, 30.0f);
    glVertex3f(40.0f, 30.0f, 30.0f);
    glVertex3f(40.0f, 30.0f, -30.0f);
    glVertex3f(40.0f, 0.0f, -30.0f);
    glEnd();
    glEnable(GL_TEXTURE_2D);
```

```
glBindTexture(GL_TEXTURE_2D, texwindow);
  glBegin(GL_QUADS);
  glNormal3f(-1.0, 0.0f, 0.0f);                    //用于定义法线向
量
  glTexCoord2f(1.0f, 0.0f);glVertex3f(39.5, 10, 10);
glTexCoord2f(1.0f, 1.0f);glVertex3f(39.5, 20, 10);
glTexCoord2f(0.0f, 1.0f);glVertex3f(39.5, 20, 0);
glTexCoord2f(0.0f, 0.0f);glVertex3f(39.5, 10, 0);
glEnd();
glDisable(GL_TEXTURE_2D);
  //绘制后边墙
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, texbackwall);
  glColor3f(0.8f,0.8f, 0.8f);
  glBegin(GL_QUADS);
  glNormal3f(0.0f, 0.0f, 1.0f); //用于定义法线向量
  glTexCoord2f(0.0f, 0.0f);
glVertex3f(-40.0f,0.0f, 30.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(-40.0f, 30.0f, 30.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(40.0f, 30.0f, 30.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(40.0f, 0.0f, 30.0f);
glEnd();
 //绘制前边墙
  glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, texbackwall);
  glColor3f(0.8f,0.8f, 0.8f);
  glBegin(GL_QUADS);
  glNormal3f(0.0f, 0.0f, 1.0f); //用于定义法线向量
  glTexCoord2f(0.0f, 0.0f);
glVertex3f(-40.0f,0.0f, -30.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(-40.0f, 30.0f, -30.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(40.0f, 30.0f, -30.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(40.0f, 0.0f, -30.0f);
glEnd();
  //画钟
glBindTexture(GL_TEXTURE_2D, texclock);
glBegin(GL_QUADS);
  glNormal3f(0.0f, 0.0f, 1.0f); //用于定义法线向量
```

```
  glTexCoord2f(0.0f, 0.0f);
glVertex3f(23.0f,18.0f, -29.8f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(23.0f, 20.0f, -29.8f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(25.0f, 20.0f, -29.8f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(25.0f, 18.0f, -29.8f);
glEnd();
glDisable(GL_TEXTURE_2D);
 //绘制空调给空调贴纹理
  glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, texairfro);
  glBegin(GL_QUADS);
  glNormal3f(0.0f, 0.0f, 1.0f);
glTexCoord2f(0.0f, 0.0f);glVertex3f(33.0f,0.0f, -26.0f);
glTexCoord2f(0.0f, 1.0f);glVertex3f(33.0f, 15.0f, -26.0f);
glTexCoord2f(1.0f, 1.0f);glVertex3f(37.0f, 15.0f, -26.0f);
glTexCoord2f(1.0f, 0.0f);glVertex3f(37.0f, 0.0f, -26.0f);
glEnd();
glBindTexture(GL_TEXTURE_2D, texairback);
glBegin(GL_QUADS);
  glNormal3f(-1.0f, 0.0f, 0.0f);
glTexCoord2f(0.0f, 0.0f);glVertex3f(33.0f,0.0f, -26.0f);
glTexCoord2f(0.0f, 1.0f);glVertex3f(33.0f, 15.0f, -26.0f);
glTexCoord2f(1.0f, 1.0f);glVertex3f(33.0f, 15.0f, -29.0f);
glTexCoord2f(1.0f, 0.0f);glVertex3f(33.0f, 0.0f, -29.0f);
glEnd();
glDisable(GL_TEXTURE_2D);
//绘制教室两边石柱前边两根
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, texpole);
for(int i=0;i<=1;i++)
{
    glColor3f(1.0f,1.0f,1.0f);
    //石柱上表面
    glBegin(GL_QUADS);
    glNormal3f(0.0f, -1.0f, 0.0f); //用于定义法线向量
    glTexCoord2f(0.0f, 0.0f);glVertex3f(-40.0+i*78,30.0f, -4.0f);
    glTexCoord2f(1.0f, 0.0f);glVertex3f(-40.0+i*78, 30.0f, -6.0f);
    glTexCoord2f(1.0f, 1.0f);glVertex3f(-38.0+i*78, 30.0f, -6.0f);
    glTexCoord2f(0.0f, 1.0f);glVertex3f(-38.0f+i*78, 30.0f, -4.0f);
    glEnd();
    //石柱前表面
```

```
        glBegin(GL_QUADS);
        glNormal3f(0.0f, 0.0f, 1.0f); //用于定义法线向量
        glTexCoord2f(0.0f, 0.0f);glVertex3f(-40.0+i*78,0.0f, -4.0f);
        glTexCoord2f(0.0f, 1.0f);glVertex3f(-40.0+i*78, 30.0f, -4.0f);
        glTexCoord2f(1.0f, 1.0f);glVertex3f(-38.0+i*78, 30.0f, -4.0f);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(-38.0+i*78, 0.0f, -4.0f);
        glEnd();
        //石柱后表面
        glBegin(GL_QUADS);
        glNormal3f(0.0f, 0.0f, -1.0f); //用于定义法线向量
        glTexCoord2f(0.0f, 0.0f);glVertex3f(-40.0+i*78,0.0f, -6.0f);
        glTexCoord2f(0.0f, 1.0f);glVertex3f(-40.0+i*78, 30.0f, -6.0f);
        glTexCoord2f(1.0f, 1.0f);glVertex3f(-38.0+i*78, 30.0f, -6.0f);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(-38.0+i*78, 0.0f, -6.0f);
        glEnd();
        //石柱右表面
        glBegin(GL_QUADS);
        glNormal3f(0.0f, 0.0f, -1.0f); //用于定义法线向量
        glTexCoord2f(0.0f, 0.0f);glVertex3f(-38.0+i*76, 0.0f, -4.0f);
        glTexCoord2f(0.0f, 1.0f);glVertex3f(-38.0+i*76, 30.0f, -4.0f);
        glTexCoord2f(1.0f, 1.0f);glVertex3f(-38.0+i*76, 30.0f, -6.0f);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(-38.0+i*76, 0.0f, -6.0f);
        glEnd();
        }
        //绘制教室两边石柱，后边两根
    for(int j=0;j<=1;j++)
    {
        glColor3f(1.0f,1.0f,1.0f);
        //石柱上表面
        glBegin(GL_QUADS);
        glNormal3f(0.0f, -1.0f, 0.0f); //用于定义法线向量
        glTexCoord2f(0.0f, 0.0f);glVertex3f(-40.0+j*78,30.0f, 14.0f);
        glTexCoord2f(0.0f, 1.0f);glVertex3f(-40.0+j*78, 30.0f, 12.0f);
        glTexCoord2f(1.0f, 1.0f);glVertex3f(-38.0+j*78, 30.0f, 12.0f);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(-38.0f+j*78, 30.0f, 14.0f);
        glEnd();
        //石柱前表面
        glBegin(GL_QUADS);
        glNormal3f(0.0f, 0.0f, 1.0f); //用于定义法线向量
        glTexCoord2f(0.0f, 0.0f);glVertex3f(-40.0+j*78,0.0f, 14.0f);
        glTexCoord2f(0.0f, 1.0f);glVertex3f(-40.0+j*78, 30.0f, 14.0f);
        glTexCoord2f(1.0f, 1.0f);glVertex3f(-38.0+j*78, 30.0f, 14.0f);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(-38.0+j*78, 0.0f, 14.0f);
        glEnd();
```

```
                    //石柱后表面
                    glBegin(GL_QUADS);
                    glNormal3f(0.0f, 0.0f, -1.0f); //用于定义法线向量
                    glTexCoord2f(0.0f, 0.0f);glVertex3f(-40.0+j*78,0.0f, 12.0f);
                    glTexCoord2f(0.0f, 1.0f);glVertex3f(-40.0+j*78, 30.0f, 12.0f);
                    glTexCoord2f(1.0f, 1.0f);glVertex3f(-38.0+j*78, 30.0f, 12.0f);
                    glTexCoord2f(1.0f, 0.0f);glVertex3f(-38.0+j*78, 0.0f, 12.0f);
                    glEnd();
                    //右表面
                    glBegin(GL_QUADS);
                    glNormal3f(0.0f, 0.0f, -1.0f); //用于定义法线向量
                    glTexCoord2f(0.0f, 0.0f);glVertex3f(-38.0+j*76, 0.0f, 14.0f);
                    glTexCoord2f(0.0f, 1.0f);glVertex3f(-38.0+j*76, 30.0f, 14.0f);
                    glTexCoord2f(1.0f, 1.0f);glVertex3f(-38.0+j*76, 30.0f, 12.0f);
                    glTexCoord2f(1.0f, 0.0f);glVertex3f(-38.0+j*76, 0.0f, 12.0f);
                    glEnd();
                }
                    glDisable(GL_TEXTURE_2D);
                      //绘制黑板
                    glEnable(GL_TEXTURE_2D);
                    glBindTexture(GL_TEXTURE_2D, texblackboard);
                    glBegin(GL_QUADS);
                    glNormal3f(0.0f, 0.0f, 1.0f); //用于定义法线向量
                    glTexCoord2f(0.0f, 0.0f);glVertex3f(-20.0,8.0f, -29.9f);
                    glTexCoord2f(0.0f, 1.0f);glVertex3f(-20.0, 18.0f, -29.9f);
                    glTexCoord2f(1.0f, 1.0f);glVertex3f(20.0, 18.0f, -29.9f);
                    glTexCoord2f(1.0f, 0.0f);glVertex3f(20.0, 8.0f, -29.9f);
                    glEnd();
                    glDisable(GL_TEXTURE_2D);
                    //画黑板上方的灯
                       GLfloat blacklight[]={0.9,0.9,0.9,1};
                    glColor3f(1.0f,1.0f,1.0f);
                    glPushMatrix();
                    glTranslatef(-15,20.4,-29.5);
                     glScalef(8.0f,0.8,1.0f);

glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,blacklight);
                     glutSolidCube(1.0f);
                    glPopMatrix();
                    glPushMatrix();
                    glTranslatef(12,20.4,-29.5);
                     glScalef(8.0f,0.8,1.0f);

glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,blacklight);
```

```
 glutSolidCube(1.0f);
glPopMatrix();
//绘制教室前边一块高地并贴纹理
   glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, texgaodi);
//贴上面
glBegin(GL_QUADS);
glNormal3f(0.0f, 1.0f, 0.0f); //用于定义法线向量
glTexCoord2f(0.0f, 0.0f);glVertex3f(-30.0f, 1.5f, -22.0f);
glTexCoord2f(0.0f, 1.0f);glVertex3f(-30.0f, 1.5f, -30.0f);
glTexCoord2f(1.0f, 1.0f);glVertex3f(30.0f, 1.5f, -30.0f);
glTexCoord2f(1.0f, 0.0f);glVertex3f(30.0f,    1.5f, -22.0f);
glEnd();
//贴左边
glBegin(GL_QUADS);
glNormal3f(0.0f, 0.0f, 1.0f); //用于定义法线向量
glTexCoord2f(0.0f, 0.0f);glVertex3f(-30.0f, 0, -22.0f);
glTexCoord2f(0.0f, 1.0f);glVertex3f(-30.0f, 1.5f, -22.0f);
glTexCoord2f(1.0f, 1.0f);glVertex3f(-30.0f,    1.5f, -30.0f);
glTexCoord2f(1.0f, 0.0f);glVertex3f(-30.0f,    0, -30.0f);
glEnd();
//贴前边
glBegin(GL_QUADS);
glNormal3f(0.0f, 1.0f, 0.0f); //用于定义法线向量
glTexCoord2f(0.0f, 0.0f);glVertex3f(-30.0f, 0, -22.0f);
glTexCoord2f(0.0f, 1.0f);glVertex3f(-30.0f, 1.5f, -22.0f);
glTexCoord2f(1.0f, 1.0f);glVertex3f(30.0f,    1.5f, -22.0f);
glTexCoord2f(1.0f, 0.0f);glVertex3f(30.0f,    0, -22.0f);
glEnd();
//贴右边
glBegin(GL_QUADS);
glNormal3f(0.0f, 1.0f, 0.0f); //用于定义法线向量
glTexCoord2f(0.0f, 0.0f);glVertex3f(30.0f, 0, -22.0f);
glTexCoord2f(0.0f, 1.0f);glVertex3f(30.0f, 1.5f, -22.0f);
glTexCoord2f(1.0f, 1.0f);glVertex3f(30.0f,    1.5f, -30.0f);
glTexCoord2f(1.0f, 0.0f);glVertex3f(30.0f,    0, -30.0f);
glEnd();
glDisable(GL_TEXTURE_2D);
//绘制讲台
   //贴讲台纹理
glBindTexture(GL_TEXTURE_2D, texsdesk);
glEnable(GL_TEXTURE_2D);
glBegin(GL_QUADS);
glNormal3f(0.0f, 1.0f, 0.0f); //用于定义法线向量
```

```
glTexCoord2f(0.0f, 0.0f);glVertex3f(-7.5f, 1.5f, -24.0f);
glTexCoord2f(0.0f, 1.0f);glVertex3f(-7.5f, 9.5f, -24.0f);
glTexCoord2f(1.0f, 1.0f);glVertex3f(7.5f,  9.5f, -24.0f);
glTexCoord2f(1.0f, 0.0f);glVertex3f(7.5f,  1.5f, -24.0f);
glEnd();
glBegin(GL_QUADS);
glNormal3f(0.0f, 1.0f, 0.0f); //用于定义法线向量
glTexCoord2f(0.0f, 0.0f);glVertex3f(7.5f, 1.5f, -24.0f);
glTexCoord2f(0.0f, 1.0f);glVertex3f(7.5f, 9.5f, -24.0f);
glTexCoord2f(1.0f, 1.0f);glVertex3f(7.5f,  9.5f, -28.0f);
glTexCoord2f(1.0f, 0.0f);glVertex3f(7.5f,  1.5f, -28.0f);
glEnd();
glBegin(GL_QUADS);
glNormal3f(0.0f, 1.0f, 0.0f); //用于定义法线向量
glTexCoord2f(0.0f, 0.0f);glVertex3f(-7.5f, 1.5f, -24.0f);
glTexCoord2f(0.0f, 1.0f);glVertex3f(-7.5f, 9.5f, -24.0f);
glTexCoord2f(1.0f, 1.0f);glVertex3f(-7.5f,  9.5f, -28.0f);
glTexCoord2f(1.0f, 0.0f);glVertex3f(-7.5f,  1.5f, -28.0f);
glEnd();
glBegin(GL_QUADS);
glNormal3f(0.0f, 1.0f, 0.0f); //用于定义法线向量
glTexCoord2f(0.0f, 0.0f);glVertex3f(-7.5f, 9.5f, -24.0f);
glTexCoord2f(0.0f, 1.0f);glVertex3f(-7.5f, 9.5f, -26.0f);
glTexCoord2f(1.0f, 1.0f);glVertex3f(7.5f,  9.5f, -26.0f);
glTexCoord2f(1.0f, 0.0f);glVertex3f(7.5f,  9.5f, -24.0f);
glEnd();
//画门
   glColor3f(0.521f,0.121f,0.0547f);
   glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, texdoor);
glBegin(GL_QUADS);
glNormal3f(-1.0f, 0.0f, 0.0f); //用于定义法线向量
glTexCoord2f(0.0f, 0.0f);glVertex3f(39.9f, 0.0f, -25.0f);
glTexCoord2f(0.0f, 1.0f);glVertex3f(39.9f, 14.0f, -25.0f);
glTexCoord2f(1.0f, 1.0f);glVertex3f(39.9f, 14.0f, -19.0f);
glTexCoord2f(1.0f, 0.0f);glVertex3f(39.9f,  0.0f, -19.0f);
glEnd();
glDisable(GL_TEXTURE_2D);
//绘制音响
glColor3f(0.0f,0.0f,0.0f);
glPushMatrix();
glTranslatef(-37.5,26.25f,-5.5f);
 glScalef(1.0f,1.5f,1.0f);
```

```
glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,sound);
                    glutSolidCube(1.0f);
                   glPopMatrix();
                   glPushMatrix();
                   glTranslatef(37.5,26.25f,-5.5f);
                    glScalef(1.0f,1.5f,1.0f);

glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,sound);
                    glutSolidCube(1.0f);
                   glPopMatrix();



    }
    /************************************  绘  制  投  影  仪
*********************************************/
    void drawprojector()
    {
        glColor3f(1.0f,1.0f,1.0f);
        glBegin(GL_QUADS);
       glNormal3f(1.0f, 0.0f, 1.0f); //用于定义法线向量
       glVertex3f(-40.0f,30.0f,-30.0+10*sqrt(2));
       glVertex3f(-40.0+10*sqrt(2), 30.0f, -30.0f);
       glVertex3f(-40.0+10*sqrt(2), pro_up_down, -30.0f);
       glVertex3f(-40.0f, pro_up_down, -30.0+10*sqrt(2));
       glEnd();
       glColor3f(0.0f,0.0f,0.0f);
       glLineWidth(4.0f);
       glBegin(GL_LINES);
       glVertex3f(-25.0f,30.0f,-15.0f);
       glVertex3f(-25.0f,25.0f,-15.0f);
       glEnd();
        glColor3f(0.5f,0.5f,0.5f);
       glPushMatrix();
       glTranslatef(-25.0f,24.0f,-15.0f);
       glScalef(4.0f,2.0f,2.0f);
       glutSolidCube(1.0f);
       glPopMatrix();
       glColor3f(0.0,0.0,0.0);
       glBegin(GL_LINE_LOOP);
       glVertex3f(-40.0f,30.0f,-30.0+10*sqrt(2));
       glVertex3f(-40.0+10*sqrt(2), 30.0f, -30.0f);
       glVertex3f(-40.0+10*sqrt(2), pro_up_down, -30.0f);
       glVertex3f(-40.0f, pro_up_down, -30.0+10*sqrt(2));
```

```
    glEnd();

}
void drawdesks()
{
    //画桌子
    GLfloat desk[]={1,0.9647,0.56078};
     for(int y=0;y<=4;y++)
    {
        for(int x=0;x<=1;x++)
        {
            //桌子上边
            glColor4f(1,0.9647,0.56078,1);
            glPushMatrix();
            glTranslatef(-20.0+x*40,8.1,-17.5+y*8);
            glScalef(10,0.2,3);

glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,desk);
            glutSolidCube(1.0f);
            glPopMatrix();
            //桌子下边
            glColor4f(1,0.9647,0.56078,1);
            glPushMatrix();
            glTranslatef(-20.0+x*40,6.1,-17.5+y*8);
            glScalef(9,0.2,3);
            glutSolidCube(1.0f);
            glPopMatrix();
            //桌子前边
            glColor4f(1,0.9647,0.56078,1);
            glPushMatrix();
            glTranslatef(-20.0+x*40,7,-18.9+y*8);
            glScalef(10,2,0.2);

glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,desk);
            glutSolidCube(1.0f);
            glPopMatrix();
              //桌腿
            glColor3f(0.0,0.0,0.0);
            glBegin(GL_LINES);
            glLineWidth(3.0f);
            glVertex3f(-25.0+x*40,6.0f, -19+y*8);
            glVertex3f(-25.0+x*40,0.0f, -19+y*8);
            glEnd();
            glBegin(GL_LINES);
```

```
                glLineWidth(3.0f);
                glVertex3f(-15.0+x*40,6.0f, -19+y*8);
                glVertex3f(-15.0+x*40,0.0f, -19+y*8);
                glEnd();
                glBegin(GL_LINES);
                glLineWidth(3.0f);
                glVertex3f(-25.0+x*40,0.0f, -18+y*8);
                glVertex3f(-25.0+x*40,0.0f, -20+y*8);
                glEnd();
                glBegin(GL_LINES);
                glLineWidth(3.0f);
                glVertex3f(-15.0+x*40,0.0f, -18+y*8);
                glVertex3f(-15.0+x*40,0.0f, -20+y*8);
                glEnd();
            }
            //画中间一排桌子
            //桌子上边
                glColor3f(1,0.9647,0.56078);
                glPushMatrix();
                glTranslatef(0,8.1,-17.5+y*8);
                glScalef(20,0.2,3);

glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,desk);
                glutSolidCube(1.0f);
                glPopMatrix();
                //桌子下边
                glColor3f(1,0.9647,0.56078);
                glPushMatrix();
                glTranslatef(0,6.1,-17.5+y*8);
                glScalef(19,0.2,3);

glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,desk);
                glutSolidCube(1.0f);
                glPopMatrix();
                //桌子前边
                glColor3f(1,0.9647,0.56078);
                glPushMatrix();
                glTranslatef(0,7,-18.9+y*8);
                glScalef(20,2,0.2);

glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,desk);
                glutSolidCube(1.0f);
                glPopMatrix();
                  //桌腿
```

```
                glColor3f(0.0,0.0,0.0);
                glBegin(GL_LINES);
                glLineWidth(3.0f);
                glVertex3f(-10.0,6.0f, -19+y*8);
                glVertex3f(-10.0,0.0f, -19+y*8);
                glEnd();
                glBegin(GL_LINES);
                glLineWidth(3.0f);
                glVertex3f(10.0,6.0f, -19+y*8);
                glVertex3f(10.0,0.0f, -19+y*8);
                glEnd();
                glBegin(GL_LINES);
                glLineWidth(3.0f);
                glVertex3f(-10.0,0.0f, -18+y*8);
                glVertex3f(-10,0.0f, -20+y*8);
                glEnd();
                glBegin(GL_LINES);
                glLineWidth(3.0f);
                glVertex3f(10,0.0f, -18+y*8);
                glVertex3f(10,0.0f, -20+y*8);
                glEnd();
    }
}
//绘制椅子
void drawchairs()
{
      GLfloat chair[]={0.1,0.67,0.62};
      for(int j=0;j<=4;j++)
    {
         for(int i=0;i<=1;i++)
        {
            //画椅子底部
            glColor3f(0.1,0.67,0.62);
            glPushMatrix();
            glTranslatef(-20+i*40,3.1,-14.5+j*8);
            glScalef(10,0.2,3);

glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,chair);
            glutSolidCube(1.0f);
            glPopMatrix();
            //画椅子靠背
            glColor3f(0.1,0.67,0.62);
            glPushMatrix();
            glTranslatef(-20+i*40,5,-13+j*8);
```

```
                glScalef(10,4,0.2);

glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,chair);
                glutSolidCube(1.0f);
                glPopMatrix();
                //画椅子腿
                  glColor3f(0.0,0.0,0.0);
                glBegin(GL_LINES);
                glLineWidth(3.0f);
                glVertex3f(-25+i*40,3.0f, -13+j*8);
                glVertex3f(-25+i*40,0.0f, -13+j*8);
                glEnd();
                glColor3f(0.0,0.0,0.0);
                glBegin(GL_LINES);
                glLineWidth(3.0f);
                glVertex3f(-15.0+i*40,3.0f, -13+j*8);
                glVertex3f(-15.0+i*40,0.0f, -13+j*8);
                glEnd();
                glColor3f(0.0,0.0,0.0);
                glBegin(GL_LINES);
                glLineWidth(3.0f);
                glVertex3f(-25.0+i*40,0.0f, -12.5+j*8);
                glVertex3f(-25+i*40,0.0f, -13.5+j*8);
                glEnd();
                glColor3f(0.0,0.0,0.0);
                glBegin(GL_LINES);
                glLineWidth(3.0f);
                glVertex3f(-15+i*40,0.0f, -12.5+j*8);
                glVertex3f(-15+i*40,0.0f, -13.5+j*8);
                glEnd();

            }
                //画椅子底部
                glColor3f(0.1,0.67,0.62);
                glPushMatrix();
                glTranslatef(0,3.1,-14.5+j*8);
                glScalef(20,0.2,3);

glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,chair);
                glutSolidCube(1.0f);
                glPopMatrix();
                //画椅子靠背
                glColor3f(0.1,0.67,0.62);
                glPushMatrix();
```

```
                glTranslatef(0,5,-13+j*8);
                glScalef(20,4,0.2);

    glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,chair);
                glutSolidCube(1.0f);
                glPopMatrix();
                //画椅子腿
                glColor3f(0.0,0.0,0.0);
                glBegin(GL_LINES);
                glLineWidth(3.0f);
                glVertex3f(-10,3.0f, -13+j*8);
                glVertex3f(-10,0.0f, -13+j*8);
                glEnd();
                glColor3f(0.0,0.0,0.0);
                glBegin(GL_LINES);
                glLineWidth(3.0f);
                glVertex3f(10,3.0f, -13+j*8);
                glVertex3f(10,0.0f, -13+j*8);
                glEnd();
                glColor3f(0.0,0.0,0.0);
                glBegin(GL_LINES);
                glLineWidth(3.0f);
                glVertex3f(-10,0.0f, -12.5+j*8);
                glVertex3f(-10,0.0f, -13.5+j*8);
                glEnd();
                glColor3f(0.0,0.0,0.0);
                glBegin(GL_LINES);
                glLineWidth(3.0f);
                glVertex3f(10,0.0f, -12.5+j*8);
                glVertex3f(10,0.0f, -13.5+j*8);
                glEnd();

        }

    }
    /********************************************** 窗 口 刷 新 函 数
**********************************************/
    void reshape(int we,int he)
    {

        WinWidth=we;
        WinHeight=he;
        glViewport(0,0,(GLsizei) we, (GLsizei) he);
         glMatrixMode(GL_PROJECTION);
```

```
        glLoadIdentity();
        gluPerspective(90.0f, (GLfloat)we/(GLfloat)he, 0.01f,100.0f);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        gluLookAt(myEye.x, myEye.y, myEye.z, vPoint.x+30*sin(vAngle),
vPoint.y,-30*cos(vAngle), 0.0f, 1.0f, 0.0f);
    }
    /************************* 制 造 投 影 仪 升 起 放 下 效 果
*****************************************************/
    void projectup()
    {
        pro_up_down=pro_up_down+1.0f;
        if(pro_up_down>=28.0f)
            pro_up_down=28.0f;
          glutPostRedisplay();

    }
    void projectdown()
    {
        pro_up_down=pro_up_down-1.0f;
        if(pro_up_down<=10.0f)
            pro_up_down=10.0f;
          glutPostRedisplay();

    }
    /*********************************************** 显 示 函 数
*****************************************************/
    void myDisplay()
    {
        // 清除屏幕
            glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        //调用绘制函数
              drawscence();
            drawprojector();
            drawdesks();
            drawchairs();
            glFlush();
    }
    /*****************************************响应普通键盘操作，w，s，
a，d 以及退出 esc 键*******************************/
    GLvoid OnKeyboard(unsigned char key, int x, int y)
    {
        switch(key)
        {
```

```
        case 97:
            myEye.x-=0.5;
            vPoint.x-=0.5;
            if(myEye.x<=-40)
                myEye.x=-40;
            break;
        case 100:
            myEye.x+=0.5;
            vPoint.x+=0.5;
              if(myEye.x>=40)
                myEye.x=40;
            break;
        case 119:
             myEye.z-=0.5;
            if(myEye.z<=-30)
                 myEye.z=-30;
            break;
        case 115:
              myEye.z+=0.5;
            if(myEye.z>=30)
                 myEye.z=30;
            break;
        case 27:
            exit(0);

        }
        reshape(WinWidth,WinHeight);
        glutPostRedisplay();

    }
    /************************************* 响 应 特 殊 键 盘 操 作
*************************************************/
    GLvoid OnSpecial(int key, int x, int y)
    {
        switch(key)
        {
        case GLUT_KEY_LEFT:
            vAngle-=0.05;
            break;
        case GLUT_KEY_RIGHT:
            vAngle+=0.05;
            break;
        case GLUT_KEY_UP:
            myEye.y+=0.05;
```

```
            if(myEye.y>=30)
                myEye.y=30;
            break;
        case GLUT_KEY_DOWN:
            myEye.y-=0.5;
            if(myEye.y<=0)
                myEye.y=30;
            break;
        case GLUT_KEY_PAGE_DOWN:
            myEye.z+=0.5;
            if(myEye.z>=30)
                myEye.z=30;
            break;
        case GLUT_KEY_PAGE_UP:
            myEye.z-=0.5;
            if(myEye.z<=-30)
                myEye.z=-30;
            break;
        case GLUT_KEY_F1:
            projectup();
              break;
        case GLUT_KEY_F2:
            projectdown();
            break;
        case GLUT_KEY_F3:
            glEnable(GL_LIGHT1);
            break;
        case GLUT_KEY_F4:
            glDisable(GL_LIGHT1);
    }
        reshape(WinWidth,WinHeight);
        glutPostRedisplay();
    }
    GLvoid OnIdle()
    {
        glutPostRedisplay();
    }
    /*************************************初始化函数，对各项参数进行初始化*****************************************/
    void initial()
    {
         glClearColor(0,0,0,0);
        glEnable(GL_TEXTURE_2D);
        glTexEnvf(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE,GL_REPLACE);
```

```
        /*****************************************对灯光进行初始化
*************************************************/
          glLightModelfv(GL_LIGHT_MODEL_AMBIENT,model_ambient);
        glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER,GL_TRUE);
        glMaterialfv(GL_FRONT,GL_SPECULAR,mat_specular);
        glMaterialfv(GL_FRONT,GL_SHININESS,mat_shininess);

        glLightfv(GL_LIGHT0,GL_POSITION,light_position0);
        glLightfv(GL_LIGHT0,GL_AMBIENT,mat_ambient);
        glLightfv(GL_LIGHT0,GL_DIFFUSE,light);
        glLightfv(GL_LIGHT0,GL_SPECULAR,light);


        glLightfv(GL_LIGHT1,GL_POSITION,light_position1);
        glLightfv(GL_LIGHT1,GL_AMBIENT,mat_ambient);
        glLightfv(GL_LIGHT1,GL_DIFFUSE,white_light);
        glLightfv(GL_LIGHT1,GL_SPECULAR,white_light);


        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
        glEnable(GL_COLOR_MATERIAL);


/*****************************************************************
*************************/
        glShadeModel(GL_SMOOTH);
        glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);   //指定材料着
色的面
        glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);  //指定材料对镜面
光的反射
        glEnable(GL_DEPTH_TEST);//
    }
    void print()
    {
        printf("*****************************************************\n");
        printf(" \n");
        printf("操作按键信息提示：  \n");
        printf("上键和下键分别控制视角向下和向上 \n");
        printf("左键和右键分别控制向左环视和向右环视 \n");
        printf("w,s,a,d 键分别表示向前，后，左，右，进行平移（注意键盘大小
写）\n");
        printf("pgup 和 pgdn 分别控制向前漫游和向后漫游 \n");
        printf("F3、F4 键分别控制开灯、关灯 \n");
```

```
        printf("F1、F2 键分别控制投影仪放下和收起 \n");
        printf("ESC 键退出窗口 \n");
        printf(" \n");
        printf("********************************************************** \n");
        printf("***************最后，编者有话要说**********************
\n");
        printf("*********首先感谢网上面各种 openGL 的学习资料以及例子程序
********** \n");
        printf("********教室整体风格还是不够显真实,但编者是尽量模仿东大
教室风格...***** \n");
        printf("*******前面墙和后面墙以及天花板是偷懒贴的纹理,导致并没有
反应出灯光的效果*** \n");
        printf("****还有很多可改进的地方我知道，但还是花了很多很多时
间...***\n");
        printf("************************ 敬 请 老 师 批 评 指 正 !
************************ \n");
         printf("********************************************************** ");

    }
    int main(int argc, char* argv[])
    {
        myEye.x=0;
        myEye.y=15;
        myEye.z=25;
         vPoint.x=0;
        vPoint.y=15;
        vPoint.z=-30;
         vAngle=0;
         glEnable(GL_DEPTH_TEST);
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_RGBA | GLUT_SINGLE);
        glutInitWindowPosition(400, 0);
        glutInitWindowSize(800, 600);
        glutCreateWindow("classroom");
        initial();
        glutDisplayFunc(&myDisplay);
        glutReshapeFunc(reshape);
         glutKeyboardFunc(OnKeyboard);
        glutSpecialFunc(OnSpecial);
         glutIdleFunc(OnIdle);
        /************************************    设    置    纹    理
*****************************************/
         texblackboard=load_texture("blackboard.bmp");
        texwindow=load_texture("window.bmp");
```

```
        texsound=load_texture("sound.bmp");
        texceiling=load_texture("ceiling.bmp");
        texdoor=load_texture("door.bmp");
        texfloor=load_texture("floor.bmp");
        texbackwall=load_texture("backwall.bmp");
        texpole=load_texture("pole.bmp");
        texairfro=load_texture("airconditionfront.bmp");
        texairback=load_texture("airconditionback.bmp");
        texgaodi=load_texture("gaodi.bmp");
        texsdesk=load_texture("sdesk.bmp");
        texclock=load_texture("clock.bmp");
        /*****************************************************************
*********************************/
        print();
        //开始显示
        glutMainLoop();
         return 0;
 }
```