南 开 大 学计算机图形学大作业论文

中文题目：OpenGL 模拟 3D 别墅

学 号： 2013852

姓 名： 郁万祥

年 级： 2020 级

专 业： 软件工程

学 院： 软件学院

指导教师： 王靖

完成日期： 2022 年 12 月 23 日

摘 要

视觉是人类获取信息最直接、快速、多样化的途径，也因此图形是计算机向人展示信息的重要手段。如何利用计算机实现图形的表示、生成、处理、显示，将信息转换成直观或逼真的图像，是计算机图形学研究的重点。在计算机辅助设计制造，计算机动画，科学计算可视化等方面，都有重要的应用。

当下研究计算机图形学的工具比较主流的就是 opengl，考虑到大一对 C++语言进行了比较深入的学习，并且本学期上机作业主要是通过 C++借助 opengl 完成，因此本次期末课程作业将通过 C++语言，借助 opengl 工具，实现 3D 别墅庭院的模拟展示。

关键字：opengl、C++、计算机图形学、3D、纹理

Abstract

Vision is the most direct, fast and diversified way for humans to obtain information, so graphics is an important means for computers to display information to people. How to use computers to represent, generate, process, and display graphics, and convert information into intuitive or realistic images, is the focus of computer graphics research. It has important applications in computer-aided design and manufacturing, computer animation, scientific computing visualization, etc.

At present, the more mainstream tool for studying computer graphics is opengl, considering that the freshman has studied the C++ language in depth, and this semester's homework is mainly completed through C++ with the help of opengl, so this final coursework will be through the C++ language, with the help of opengl tools, to achieve a simulation display of 3D villa courtyard.

Keywords: opengl, C++, computer graphics, 3D, textures

目录

第一章 背景知识

1.1 开发环境

项目名称：OpenGL 实现 3D 别墅模拟

软件环境：Windows 11, version 22H2

硬件环境：AMD Ryzen 7 4800U with Radeon Graphics 1.80 GHz

开发工具：visual studio 2022

1.2 OpenGL

OpenGL(英语：Open Graphics Library,译名：开放图形库或者“开放式图形库")是用于渲染 2D、3D 矢量图形的跨语言、跨平台的应用程序编程接口(API)。这个接口由近 350 个不同的函数调用组成,用来从简单的图形比特绘制复杂的三维景象。而另一种程序接口系统是仅用于 Microsoft Windows_上的 Direct3D。OpenGL 常用于 CAD、虚拟实境、科学可视化程序和电子游戏开发。

OpenGL 的高效实现（利用了图形加速硬件）存在于 Windows,部分 UNIX 平台和 Mac OS。这些实现一般由显示设备厂商提供，而且非常依赖于该厂商提供的硬件。开放源代码库 Mesa 是一个纯基于软件的图形 API,它的代码兼容于 OpenGL。但是，由于许可证的原因，它只声称是一个“非常相似”的 API。

OpenGL 规范由 1992 年成立的 OpenGL 架构评审委员会(ARB)维护。ARB 由一些对创建一个统一的、普遍可用的 API 特别感兴趣的公司组成。

常用函数：

颜色编辑	glShadeModel,glColor,glColorPointer,gllndex, gllndexPointer,glColorTableEXT,glColorSubTableEXT
绘制几何图形	glVertex,glVertexPointer,glArrayElement,glBegin glEnd, glEdgeFlag,glEdgeFlagv,gIPointSize,glLineWidth,glLineStipple, glPolygonMode,gIFrontFace,glPolygonStipple,glDrawElements,gIRect

坐标转换	glTranslate,glRotate,glScale,glViewPoint,glFrustum,glOrtho,glClipPlane
堆栈操作	glLoadMatrix,gIMultMatrix,gIMatrixMode,glPushMatrix,glPopMatrix, glPushAttrib,glPopAttrib,glPushName,glPopName,gllnitName glLoadName
显示列表	gINewList,glEndList,glCallList,glCallLists,glGenLists,glDeleteLists, gllsList,
使用光照和材质	gINormal,gINormalPointer,glLight,glLightModel,gIMaterial, glColorMaterial,
像素操作	glRasterPos,glBitmap,glReadPixels,glDrawPixels,glCopyPixels, glCopyTexImage1D,glCopyTexImage2D,glCopyTexSublmage1D, glCopyTexSublmage2D,glPixelZoom,glPixelStore,glPixelTransfer,glPixelMap
特效操作	glBlendFunc,glHint,glFog
帧缓存操作	glClear,glClearAccum,glClearColor,glClearDepth,glClearlndex, gIClearStencil,glDrawBuffer,gllndexMask,glColorMask,glDepthMask, glStencilMask,glAlphaFunc,glStencilFunc,glStencilOp, glDepthFunc,glDepthRange,glLogicOp,glAccum,
曲线或曲面绘制	glEvalCoord,glMap1,glMap2,glMapGrid,glEvalMesh,glEvalPoint,
查询函数	glGet,glGetClipPlane,glGetColorTableEXT,glGetColorTableParameterfvEXT, glGetColorTableParameterivEXT,glGetError,glGetLight,glGetMap, glGetMaterial,glGetPixelMap,glGetPointerv,glGetPolygonStipple, glGetString,glGetTexEnv,glGetTexlmage,glGetTexLevelParameter, glGetTexParameter,

第二章 实现细节

2.1 建模过程

对三维立体场景以及场景中的各对象进行模型建立，包括光照和面绘制。设置了环境光和平行光两种光照类型，面绘制则采用了两种绘制方式，单层绘制负责圆形的绘制，两层层级绘制则用于其他图形的绘制。

1）首先，简单地调用 Open GL 的初始化函数，实现应用程序的初始化工作。之后，建立窗口的位置，定义窗口的宽度和高度，指定窗口的显示模式。在 iit 函数中对光照和视角进行了初始化，注册了绘图函数 display 负责三维场景中所有对象的绘制。

```
int main(int argc, char** argv)    //  main函数 增加键盘事件
{
    glutInit(&argc, argv);
    glutInitDisplayMode(displayMode:GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(width:w, height:h);
    glutInitWindowPosition(x:100, y:100);
    glutCreateWindow("The House");
    init();
    glBlendFunc(sfactor:GL_SRC_ALPHA, dfactor:GL_ONE_MINUS_SRC_ALPHA);
    glEnable(cap:GL_DEPTH_TEST);

    glEnable(cap:GL_TEXTURE_2D);    // 启用纹理

    texGround = load_texture(file_name:"grass4.bmp");  //加载纹理
    texPoor = load_texture(file_name:"swimmingpoor3.bmp");
    texFloor = load_texture(file_name:"floor.bmp");

    texRoof = load_texture(file_name:"roof.bmp");
    texwall = load_texture(file_name:"wall.bmp");
    glutDisplayFunc(callback:display);

    glutSpecialFunc(callback:specialkeys);  //调用键盘控制函数
    glutTimerFunc(time:1000, callback:timerFunction, value:1);    //注册定时器
    glutMainLoop();
    system(_Command:"pause");
    return 0;
}
```

2）面绘制中两层层级绘制方式，上层设置了 constract 函数和 cons 函数分别根据输入的坐标参数对长方体和梯形的顶点数组进行赋值，再由下层 build2 函数和 build 函数分别根据顶点坐标进行面绘制。

```
void constract(double x, double y, double z, double x1, double y1, double z1) { //长方体
    fang[0][0] = x;
    fang[0][1] = y;
    fang[0][2] = z;         //  第0个点

    fang[1][0] = x;
    fang[1][1] = y;
    fang[1][2] = z + z1;      //  第一个点

    fang[2][0] = x + x1;
    fang[2][1] = y;
    fang[2][2] = z + z1;      //  第二个点

    fang[3][0] = x + x1;
    fang[3][1] = y;
    fang[3][2] = z;         //  第三个点
    for (int i = 0; i < 4; i++) {  // for()循环来完成其余的四个点
        for (int j = 0; j < 3; j++) {
            if (j == 1)
                fang[i + 4][j] = fang[i][j] + y1;
            else
                fang[i + 4][j] = fang[i][j];
            }
    }
}
```

```
void build2() {
    glBegin(mode:GL_POLYGON);
    //glColor3f(red);
    glNormal3f(nx:0.0, ny:-1.0, nz:0.0);
    glVertex3f(x:san[0][0], y:san[0][1], z:san[0][2]);
    glVertex3f(x:san[1][0], y:san[1][1], z:san[1][2]);
    glVertex3f(x:san[2][0], y:san[2][1], z:san[2][2]);
    glVertex3f(x:san[3][0], y:san[3][1], z:san[3][2]);
    glEnd();   //      下底

    glBegin(mode:GL_POLYGON);
    //glColor3f(green);
    glNormal3f(nx:-1.0, ny:0.0, nz:0.0);
    glVertex3f(x:san[1][0], y:san[1][1], z:san[1][2]);
    glVertex3f(x:san[0][0], y:san[0][1], z:san[0][2]);
    glVertex3f(x:san[4][0], y:san[4][1], z:san[4][2]);
    glVertex3f(x:san[5][0], y:san[5][1], z:san[5][2]);
    glEnd();   //      左面

    glBegin(mode:GL_POLYGON);
    //glColor3f(green);
    glNormal3f(nx:1.0, ny:0.0, nz:0.0);
    glVertex3f(x:san[7][0], y:san[7][1], z:san[7][2]);
    glVertex3f(x:san[6][0], y:san[6][1], z:san[6][2]);
    glVertex3f(x:san[3][0], y:san[3][1], z:san[3][2]);
    glVertex3f(x:san[2][0], y:san[2][1], z:san[2][2]);
    glEnd();   //      右面
```

3）场景中各对象的实现只需调用上层 constract 函数和 cons 函数或 glutSolidSphere 函数即可。

```
//*****************************增加车库墙（颜色为 白色）********
glColor3f(red: green: blue:white);
constract(x:320, y:10, z:250, x1:10, y1:100, z1:220);
build();
constract(x:460, y:10, z:250, x1:10, y1:100, z1:220);
build();
//*****************************增加车库顶棚（颜色为 深灰色）********
glColor3f(red: green: blue:hgray);
constract(x:320, y:110, z:250, x1:150, y1:10, z1:220);
build();
//*****************************增加卧室和客厅的墙壁（颜色为 白色）********

glColor3f(red: green: blue:white);
constract(x:438, y:10, z:50, x1:12, y1:150, z1:197);
build();     //右墙壁
glColor3f(red: green: blue:white);
constract(x:62, y:10, z:50, x1:376, y1:150, z1:12);
build();     //上墙壁

glColor3f(red: green: blue:white);
constract(x:62, y:10, z:235, x1:376, y1:35, z1:12);
build();     //下墙壁1
glColor3f(red: green: blue:white);
constract(x:62, y:45, z:235, x1:50, y1:65, z1:12);
build();     //下墙壁2
glColor3f(blue: green: red:white);
```

2.2 纹理绘制

实现效果：实现场景中的草地，泳池以及房价内地板的纹理的映射。

实现过程：首先对纹理对象进行了定义，在纹理绑定函数中读取纹理绘制所需的 bmp 图片。之后进行启动纹理、绑定纹理、关闭纹理三步骤。

```
glEnable(cap:GL_TEXTURE_2D);    //泳池纹理
glBindTexture(target:GL_TEXTURE_2D, texture:texPoor);
glBegin(mode:GL_QUADS);
glTexCoord2f(s:0.0f, t:0.0f); glVertex3f(x:490, y:16, z:75);
glTexCoord2f(s:0.0f, t:1.0f); glVertex3f(x:490, y:16, z:425);
glTexCoord2f(s:1.0f, t:1.0f); glVertex3f(x:640, y:16, z:425);
glTexCoord2f(s:1.0f, t:0.0f); glVertex3f(x:640, y:16, z:75);
glEnd();
glDisable(cap:GL_TEXTURE_2D);
```

2.3 交互控制

实现效果：实现键盘对于相机角度的操控，可以对 3D 场景进行 360 度的查看。

实现过程：键盘的交互控制通过 specialkeysl 函数实现，函数参数为键盘按下的键 key。方向键上下左右控制三维场景的观测视角，该旋转通过改变场景轴旋转变量 rotate_x、rotate_y 实现。

```
void specialkeys(int key, int x, int y) {   //键盘相应函数


    //旋转
    if (key == GLUT_KEY_RIGHT)
        rotate_y -= 1;
    else if (key == GLUT_KEY_LEFT)
        rotate_y += 1;
    else if (key == GLUT_KEY_DOWN)
        rotate_x -= 1;
    else if (key == GLUT_KEY_UP)
        rotate_x += 1;

    glutPostRedisplay();
}
```

2.4 摄像机漫游

1）、函数原型

```
void gluLookAt(GLdouble eyex,GLdouble eyey,GLdouble eyez,GLdouble center
x,GLdouble centery,GLdouble centerz,GLdouble upx,GLdouble upy,GLdouble u
pz);
```

2）、参数说明

第一组 eyex,eyey,eyez 相机在世界坐标的位置

第二组 centerx,centery,centerz 相机镜头对准的物体在世界坐标的位置

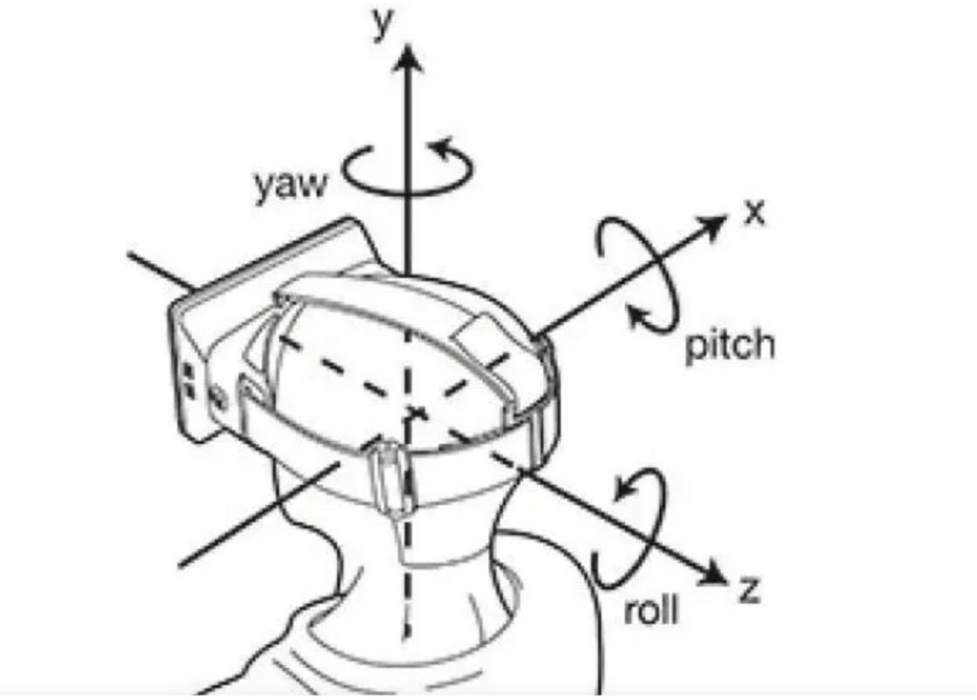
第三组 upx,upy,upz 相机向上的方向在世界坐标中的方向

3）、摄像机移动

摄像机移动时，在自身坐标系中的朝向并不改变，因此只需要改变摄像机在世界坐标系中的位置坐标即可。

4）、摄像机旋转

关于 OpenGL 坐标系，简单地说就是窗口中心是原点，水平向右是 x 轴正方向，垂直向上是 y 轴正方向，垂直屏幕向外是 Z 轴正方向。摄像机方向与 xOy 平面，xOz 平面，yOz 平面都会呈一定的角度。在摄像机旋转时，摄像机位置不改变，仅改变朝向。



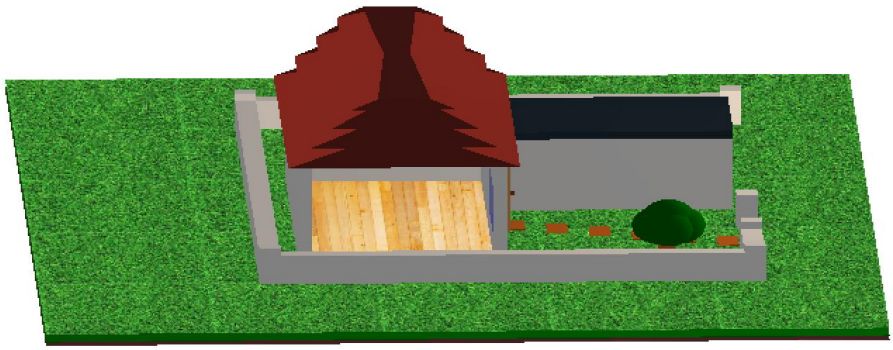
pitch 是围绕 X 轴旋转，也叫做俯仰角

yaw 是围绕 Y 轴旋转，也叫偏航角

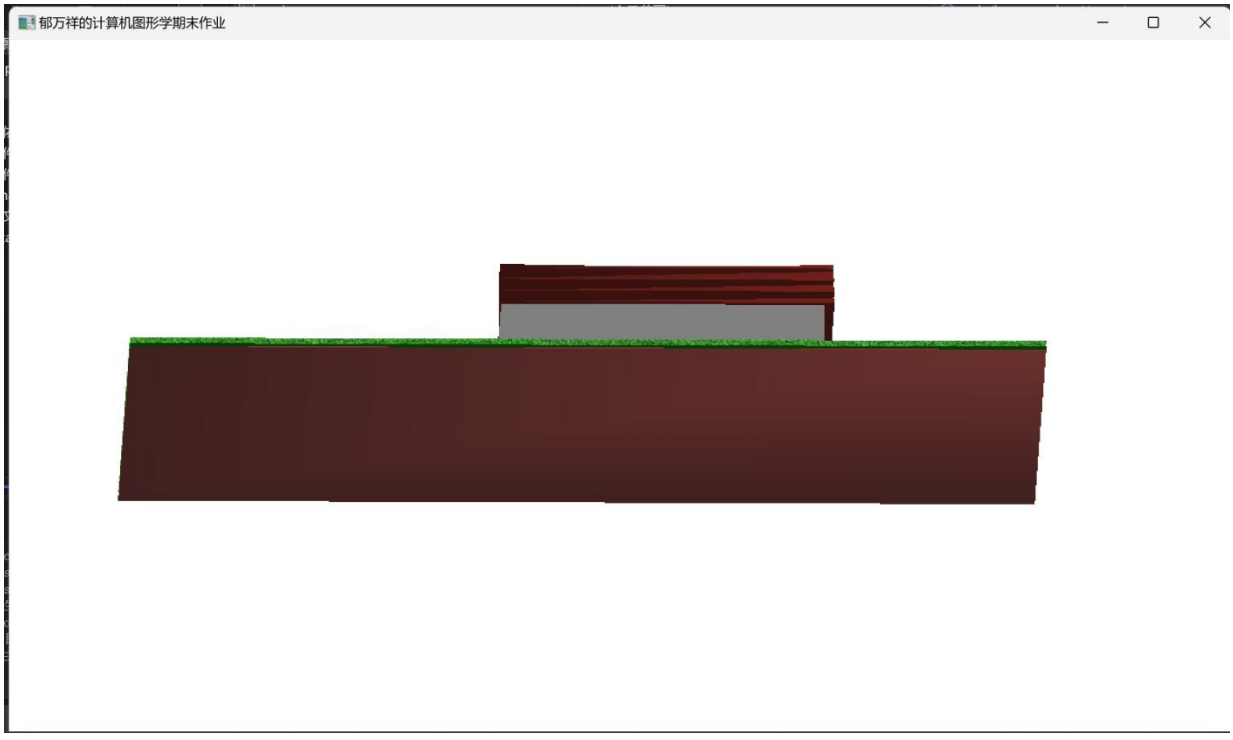
roll 是围绕 Z 轴旋转，也叫翻滚角

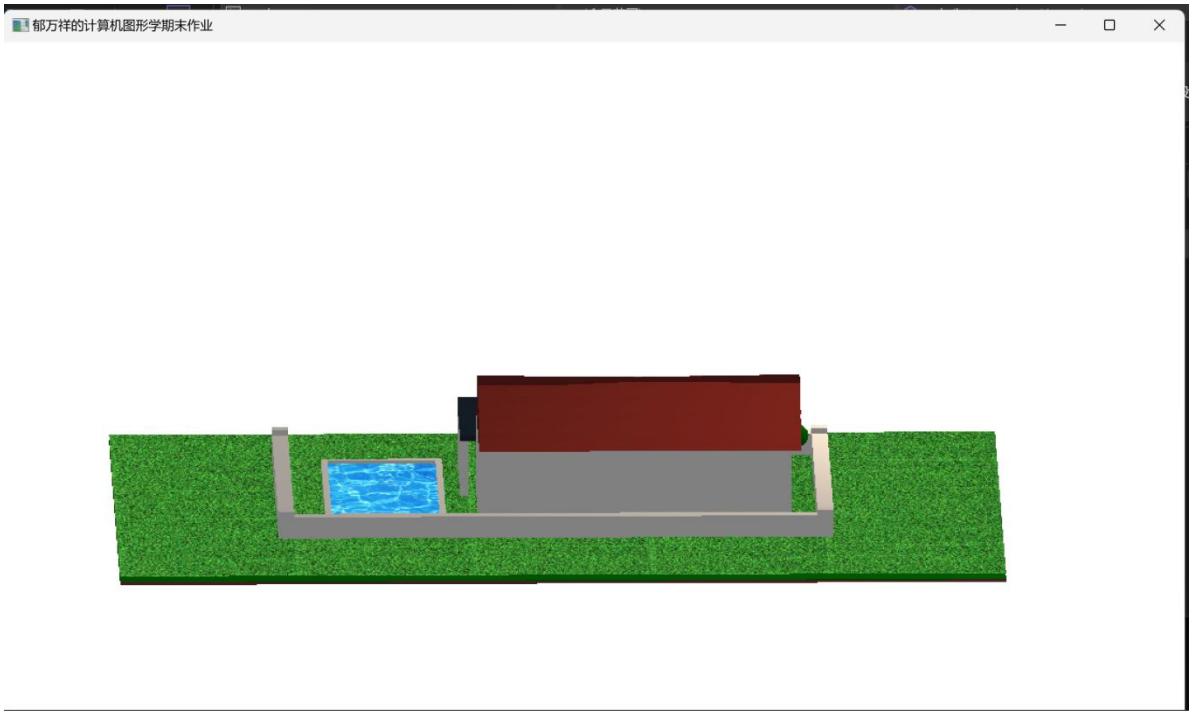
如图，摄像机的旋转有俯仰(pitch),偏航(yaw)和翻滚(roll)三种方式。

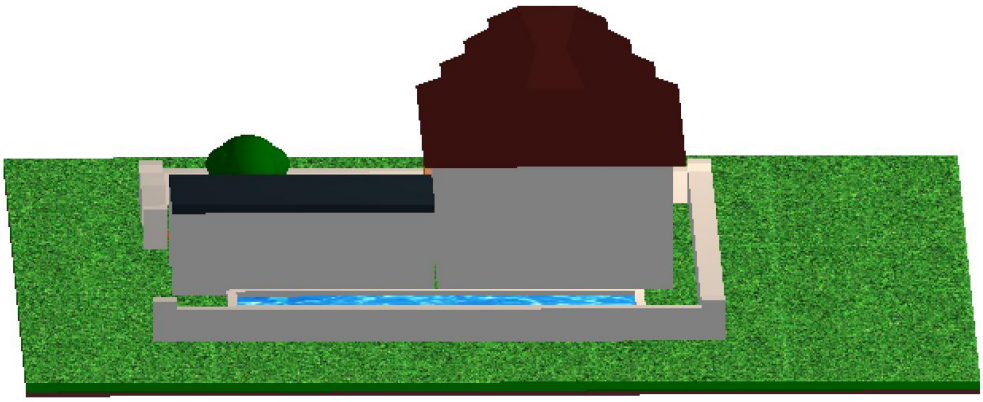
第三章 效果展示




郁万祥的计算机图形学期末作业


郁万祥的计算机图形学期末作业


郁万祥的计算机图形学期末作业



第四章 总结与思考

至此，本学期的计算机图形学课程完美收官。这次计算机图形学的大作业，是借助 OpenGL 这款 API，使用 C++语言，进行了 3D 别墅的建模，同时添加了纹理效果使得模型更加真实，同时通过添加键盘交互和摄像机的旋转漫游，实现了用户可以通过操作键盘的上下左右键，进行对建模多角度的查看。

对于我而言，相对于理论课程上对于图形学的理论知识，这门课程更加直观让我学习到的就是使用 openGL 这款工具，使用编程语言在计算机上进行图形化的编辑操作。不仅仅是在计算机上画出图形甚至 3D 模型，它还提供了许多我们经常使用到的工具，比如对于摄像机的角度的转换，还有摄像机的移动，这可以提供我们类似于第一人称或者第三人称视角的观感。同时，还提供给了我们许多真实效果，类似于光照的编辑、纹理的编辑以及雾化效果，这使得我们对于图形化的操作更加贴近真实。

这门课更多提供的是一种启蒙作用，带我进入到了计算机图形学的大门，然而，计算机图形学的功能不仅仅只是在计算机进行图形化的编辑操作，通过查阅资料，计算机图形学在现代很多的领域都发挥了很大的作用，包括 CAD/CAM，计算机辅助教学，计算机动画，计算机游戏，虚拟现实等等。这些都是现在非常热门的技术领域，因此，这让我不仅仅满足于编辑 3D 图形.....