《计算机图形学实验》综合实验报告

题目　基于 OpenGL 的三维图形的实现

学　号　20191060003

姓　名　刘文长

指导教师　钱文华

日　期　2022.6.8

摘要

使用 OpenGL 实现三维图形渲染，设计一个茶壶的三维图形，渲染过程加入纹理、光照等效果，采用了光照明模型、纹理贴图等算法。运用计算机图形学理论、算法、技术等要求设计并实现程序。

关键词：OpenGL；三维图形；Utah 茶壶

目录

1. 实验背景

三维图形渲染是指在电子计算机上将三维模型转换为二维计算机图形的三维计算机图形制图过程，也就是从准备的场景创建实际的二维景象或动画的最后阶段。这可以和现实世界中在布景完成后的照相或摄制场景的过程相比。

用于诸如游戏或模拟程序这样的交互式媒体的渲染需要实时计算和显示，速度约为 20 到 120 帧每秒。非交互式媒体（譬如录象或电影），渲染的慢得多。非实时渲染使得有限的计算能力得以放大以获得高质量的画面。复杂场景的单帧的渲染速度可能从几秒到一个小时或者更多。渲染完成的帧存贮在硬盘，然后可能转录到其它媒介，例如电影胶卷或者光盘。然后这些帧以高帧率播放，通常为 24，25，或 30 帧每秒，以达成运动的假象。

2. 实验内容

实现三维图形渲染，自定义三维图形，三维图形不能仅仅是简单的茶壶、球体、圆柱体、圆锥体等图形，渲染过程须加入纹理、色彩、光照、阴影、透明等效果，可采用光线跟踪、光照明模型、纹理贴图、纹理映射等算法。评分标准包括实验设计、实验完成情况、报告撰写情况等。学生独立完成，严禁抄袭，发现抄袭不记成绩。

3. 开发工具及实现目的

开发工具：Visual C++, OpenGL, Java 等

目的:设计一个茶壶。运用计算机图形学理论、算法、技术等按照要求设计并实现程序，撰写实验报告。

4. 关键代码及算法理论

glutInit()是用 glut 来初始化 OpenGL 的，我将把所有的事情都留给这个函数，这基本上是无关紧要的，尽管它有参数，这基本上是无用的。

glutInitDisplayMode(MODE)这告诉系统我们如何需要一个显示模式。至于其参数 GLUT_RGBA 就是使用(red, green, blue)的颜色系统。

glutInitWindowSize(400, 400) 这个函数很容易理解，可以设置显示窗口的大小。际上还有个 glutInitWindowPosition()也很常用，用来设置窗口出现的位置。

glutCreateWindow(“Teapot”)，一旦被调用，就会出现一个窗口，参数就是窗口的标题。

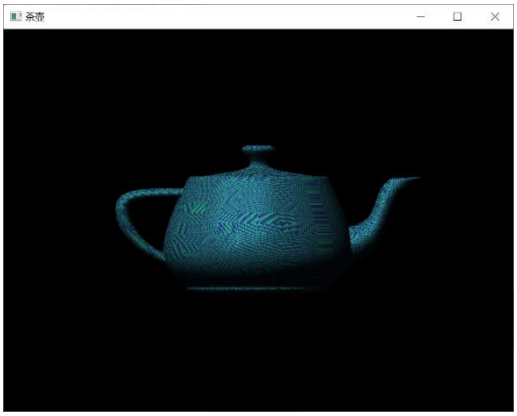
glutDisplayFunc(drawFunc) Glut 是一个很好的特性，它注册一个函数来绘制 OpenGL 窗口，那个函数有很多 OpenGL 的绘图操作等等，那是我们要学的主要东西。glutMainLoop()，主循环，一旦它被调用，我们的 OpenGL 就会继续运行。与许多程序一样，主循环不断地运行，绘制即时图像，处理输入，等等。

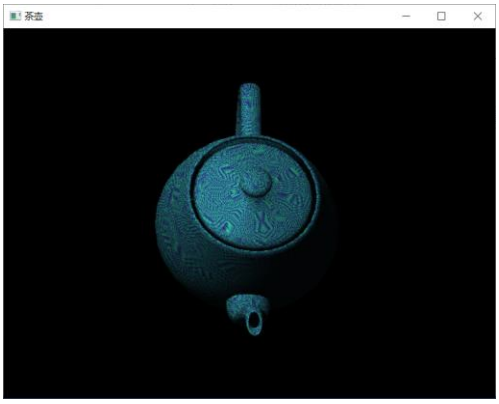
glClear(GL_COLOR_BUFFER_BIT) 擦去之前的图片。这基本上是一条定律。在重画之前先擦除原图，否则当它被叠加时你什么也看不见。

glFlush()不用说，画了这么多画之后，自然要刷新一下陈列。然而，这里的刷新不仅仅是屏幕上的更新;事实上，它是一个处理 OpenGL 的渲染管道，允许队列中的所有命令被执行

glRotatef(1, 0, 1, 0) 第一个是角度，最后三个是矢量，也就是绕这个矢量旋转，在这种情况下绕 y 轴旋转 1 度。这些角度的总和最终导致茶壶绕 y 轴旋转。我们还可以从这里看到，在指定了旋转角度之后，重绘不会重置，而是在前一次旋转的结果上继续旋转。这是一个非常重要的概念，OpenGL 是一个状态机，一旦你指定了一个状态，直到你指定了一个位置，它会保持在那个状态。不仅是旋转，还有光照贴图等等，都遵循这个模式。

运行结果：





茶壶.mp4

5. 实验心得及小结

使用 OpenGL 实现三维图形渲染，设计一个茶壶的三维图形，在渲染过程加入了纹理、色彩、光照、阴影、透明等效果。采用了光照模型、纹理映射等算法。运用计算机图形学理论、算法、技术。选了钱文华老师作为导师，对计算机图形学有着浓厚的兴趣，虽然目前由于技术受限，茶壶的纹理虽然有艺术感，做不出非常有高级感的材质，以后会继续深入学习。

参考文献：

肖泽群. 基于 OpenGL 的实时三维人体动画展示方法[J]. 科学与信息化,2021(20):25-26.

白庆月,岳俊瑞,郭肖. 基于 OpenGL 的雨伞建模与仿真技术研究[J]. 机电信息,2020(33):78-79. DOI:10.3969/j.issn.1671-0797.2020.33.043.

csdn 果冻柠檬 (2020). 【Python】使用 OpenGL 绘制茶壶 北京创新乐知网络技术有限公司 (on-line). Retrieved 15 December 2020 from https://blog.csdn.net/lemon4869/article/details/106648332

页 as (2019). OpenGL 入门学习二——绘制旋转的茶壶 北京创新乐知网络技术有限公司 (on-line). Retrieved 15 December 2020 from https://blog.csdn.net/qq_32159463/article/details/99682262、

附录：

```
#include <windows.h>
#include<GL/glut.h>
#define imageWidth 50
GLfloat roate = 0.0;// 设置旋转速率
GLfloat rote = 0.0;//旋转角度
GLfloat anglex = 0.0;//X 轴旋转
GLfloat angley = 0.0;//Y 轴旋转
GLfloat anglez = 0.0;//Z 轴旋转
GLint WinW = 400;
GLint WinH = 400;
GLfloat oldx;//当左键按下时记录鼠标坐标
GLfloat oldy;
GLfloat x1 = 0.0f;
GLfloat y1 = 0.0f;
GLfloat rsize = 25;

GLfloat xstep = 1.0f;
GLfloat ystep = 1.0f;

GLfloat windowWidth;
GLfloat windowHeight;
float xrot;
GLubyte stripeImage[3 * imageWidth];
//定义纹理图像
void makeStripeImage(void)
{
        int j;
        for(j = 0; j < imageWidth; j++)
        {
                stripeImage[3 * j] = 300;
                stripeImage[3 * j + 1] = 165 / 3 * j;
                stripeImage[3 * j + 2] = 140;
        }
}
/* 参数设置 */
GLfloat sgenparams[] = {1.0, 1.0, 1.0, 0.0};

void  Scene(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glEnable(GL_DEPTH_TEST);//开启深度测试
```

```
42.         glDepthFunc(GL_LESS);//深度测试函数
43.         glColor3f(1.0, 0.0, 0.0);
44.         glLoadIdentity();//加载矩阵
45.         glPushMatrix();//矩阵入栈
46.         glRotatef(rote, 0.0f, 1.0f, 0.0f);
47.         glRotatef(anglex,1.0,0.0,0.0);
48.         glRotatef(angley,0.0,1.0,0.0);
49.         glRotatef(anglez,0.0,0.0,1.0);
50.         rote += roate;
51.         glutSolidTeapot(50);//绘制茶壶
52.         glPopMatrix();//矩阵出栈
53.         glutPostRedisplay();
54.         glutSwapBuffers();
55. }
56.
57. void RC(void)
58. {
59.         GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };//材质的镜面反射系数
60.         GLfloat mat_shininess[] = { 50.0 };//材质的镜面光指数
61.         // 光源 0
62.         GLfloat light_position[] = { -50.0, 100.0, 100.0, 0.0};//光源位置
63.         GLfloat light_ambient[] = {0.0, 0.0, 0.0, 1.0};//环境光
64.         GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };//漫反射
65.         GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };//镜面光
66.
67.         glClearColor(0.0, 0.0, 0.0, 0.0);
68.         glShadeModel(GL_SMOOTH);//光暗处理
69.         makeStripeImage();//绘制纹理
70.         //函数设定从内存中读取纹理图并放到屏幕上的方式
71.         //指定内存中每个象素行起始的排列要求为字节排列（1）
72.         glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
73.         //定义纹理环境参数：调整当前亮度和颜色信息，使之适应纹理图像
74.         glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
75.         //纹理绕转使用重复方式
76.         glTexParameterf(GL_TEXTURE_1D, GL_TEXTURE_WRAP_S, GL_REPEAT);
77.         //定义纹理放大和缩小函数均为 GL_LINEAR
78.         glTexParameterf(GL_TEXTURE_1D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
79.         glTexParameterf(GL_TEXTURE_1D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
80.         //定义纹理
81.         glTexImage1D(GL_TEXTURE_1D, 0, 3, imageWidth, 0, GL_RGB,
    GL_UNSIGNED_BYTE, stripeImage);
82.         //控制纹理坐标的生成
83.         //指定单值纹理生成参数
84.         glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);
```

```
85.         //指定纹理坐标生成函数,系数由 sgenparams 指定
86.         glTexGenfv(GL_S, GL_OBJECT_PLANE, sgenparams);
87.
88.         glEnable(GL_TEXTURE_GEN_S);//开启纹理坐标映射
89.         glEnable(GL_TEXTURE_1D);//开启纹理
90.
91.         glEnable(GL_LIGHT0);//开启 0 光源
92.         //设置材质
93.         glMaterialf(GL_FRONT, GL_SHININESS, 64.0);
94.         glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
95.         glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
96.
97.         glEnable(GL_LIGHTING);//开启光照效果
98.         //设置光照材质与位置
99.         glLightfv(GL_LIGHT0, GL_POSITION, light_position);
100.         glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
101.         glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
102.         glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
103.
104. }
105.
106. void  ChangeSize(GLsizei w, GLsizei h)
107. {
108.         glViewport(0, 0, (GLsizei) w, (GLsizei) h);
109.         glMatrixMode(GL_PROJECTION);
110.         glLoadIdentity();
111.         if(w <= h)
112.                 glOrtho(-100.0, 100, -100.0 * (GLfloat)h / (GLfloat)w, 100.0
    * (GLfloat)h / (GLfloat)w, -1000.0, 1000.0);
113.         else
114.                 glOrtho(-100.0 * (GLfloat)w / (GLfloat)h, 100.0 * (GLfloat)w
    / (GLfloat)h, -100.0, 100.0, -1000.0, 1000.0);
115.         glMatrixMode(GL_MODELVIEW);
116.         glLoadIdentity();
117. }
118. void mouse(int button, int state, int x, int y) // 鼠标函数
119.     {
120.         if (button == GLUT_LEFT_BUTTON)
121.         {
122.             if (state == GLUT_DOWN)
123.             {
124.                 roate = 0;
125.                 rote = 0;
126.                 oldx = x;//当左键按下时记录鼠标坐标
```

```
127.                oldy = y;
128.            }
129.        }
130.        if (button == GLUT_RIGHT_BUTTON)
131.        {
132.            if (state == GLUT_DOWN)
133.            {
134.                roate += 1.0f;
135.            }
136.        }
137.    }
138.
139.  void motion(int x, int y)
140.    {
141.        GLint deltax = oldx - x;
142.        GLint deltay = oldy - y;
143.        anglex  += 360 * (GLfloat)deltax / (GLfloat)WinW;//根据屏幕上鼠标滑动
    的距离来设置旋转的角度
144.        angley += 360 * (GLfloat)deltay / (GLfloat)WinH;
145.        anglez += 360 * (GLfloat)deltay / (GLfloat)WinH;
146.        oldx = x;//记录此时的鼠标坐标，更新鼠标坐标
147.        oldy = y;//若是没有这两句语句，滑动是旋转会变得不可控
148.        glutPostRedisplay();
149.    }
150.
151. int main(int argc,char** argv)
152. {
153.        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
154.        glutInitWindowSize(640, 480);
155.        glutInitWindowPosition(100, 150);
156.        glutInit(&argc,argv);
157.        glutCreateWindow("茶壶");
158.        glutDisplayFunc(Scene);
159.        glutReshapeFunc(ChangeSize);
160.        glutMouseFunc(mouse);
161.        glutMotionFunc(motion);
162.        RC();
163.        glutMainLoop();
164.        return 0;
165. }
```