**《计算机图形学实验》综合实验报告**

**题目： 三维图形渲染实验**

**学号： 20201050470**

**姓名： 王印**

**指导教师： 钱文华**

**日期： 2022.6.21**

**中文摘要**

利用Visual C++, OpenGL, Java等工具，实现三维图形渲染，绘制茶壶三维图形，渲染过程中加入纹理、色彩、光照、阴影、透明等效果，光线跟踪、光照明模型、纹理映射等算法。

**关键词:**三维图形渲染，OpenGL,光照模型，纹理，色彩，阴影

目录

[1、实验背景 2](#_Toc6353)

[2、实验内容 2](#_Toc6015)

[3、程序设计 3](#_Toc22600)

[4、程序实现步骤 3](#_Toc29536)

[5、运行结果 3](#_Toc3568)

[6、实验小结 6](#_Toc11821)

[7、参考文献 6](#_Toc9051)

[8、附录（实验代码） 6](#_Toc4366)

# 1、实验背景

图像渲染是将三维的光能传递处理转换为一个二维图像的过程。场景和实体用三维形式表示，更接近于现实世界，便于操纵和变换，而图形的显示设备大多是二维的光栅化显示器和点阵化打印机。从三维实体场景的表示N--维光栅和点阵化的表示就是图像渲染——即光栅化。光栅显示器可以看作是一个象素矩阵，在光栅显示器上显示的任何一个图形，实际上都是一些具有一种或多种颜色和灰度象素的集合。

在图像渲染前，需要准备好三维几何模型信息，三维动画定义信息和材质信息。其中三维几何模型通过三维扫描，三维交互几何建模和三维模型库中获取；三维动画定义通过运动设计，运动捕捉，运动计算和动态变形中获取：材质从扫描的照片，计算机计算出的图像和人画出的图画中获取。图像渲染中要完成的工作是：通过几何变换，投影变换，透视变换和窗口剪裁，再通过获取的材质与光影信息，生成图像。图像渲染结束后，会把图像信息输出到图像文件或视频文件，或者是显示设备的帧缓存器中完成图形生成。

# 2、实验内容

利用Visual C++, OpenGL, Java等工具，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。

# 3、程序设计

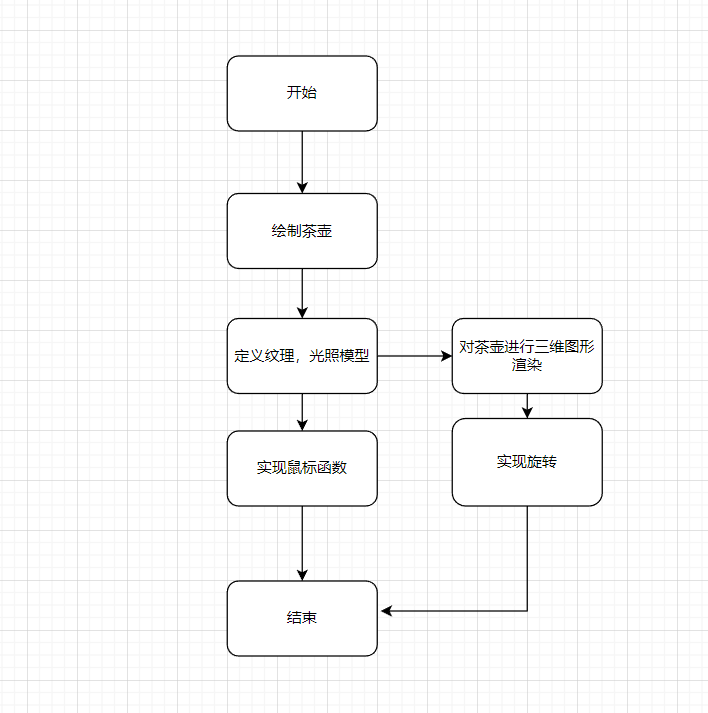
开发工具：Visual Studio 2019、OpenGL

程序设计：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，期末前提交电子版和纸质版。

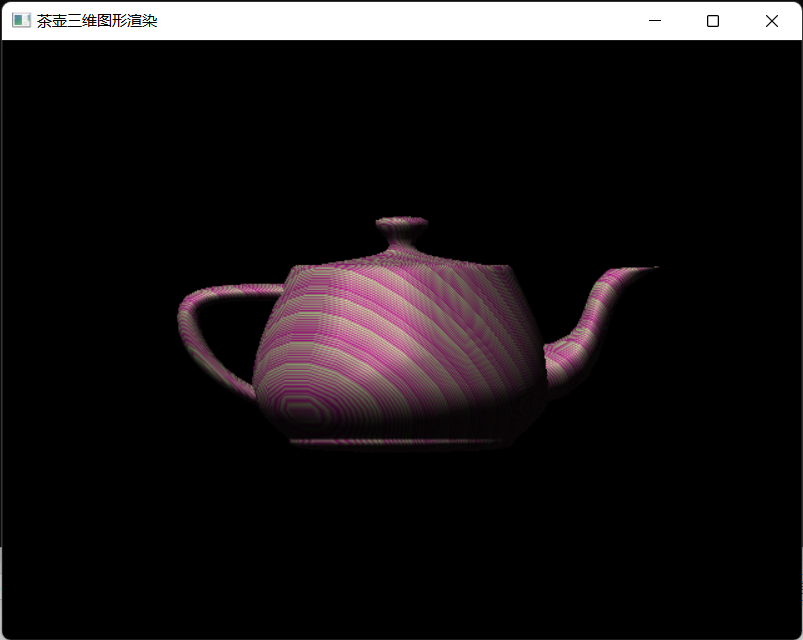
实现目的：将设计的茶壶实现三维图形渲染

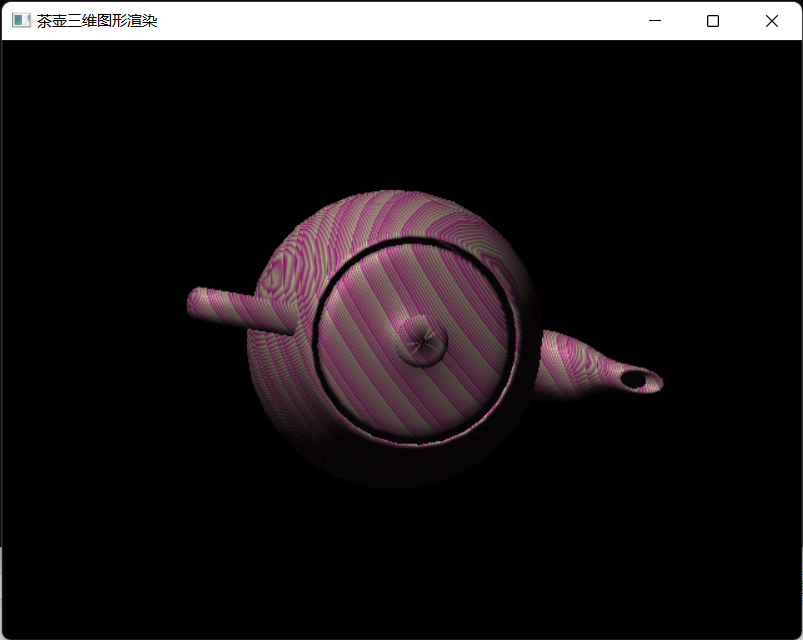
|  |  |
| --- | --- |
| TextureImage() | 定义纹理图像 |
| Draw\_Triangle() | 绘制茶壶 |
| reshape() | 光照纹理函数 |
| ChangeSize() | 图形大小 |
| mouse() | 鼠标函数 |
| motion() | 鼠标旋转函数 |

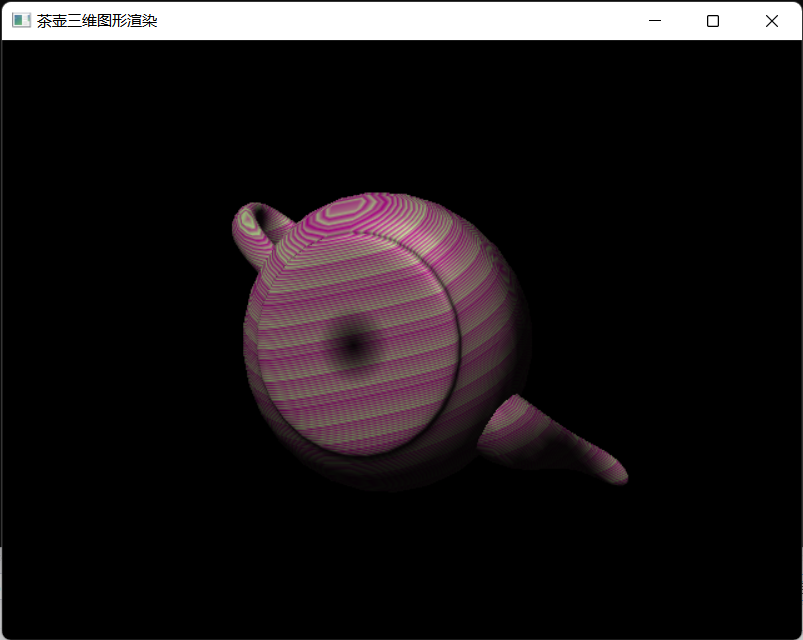
# 4、程序实现步骤

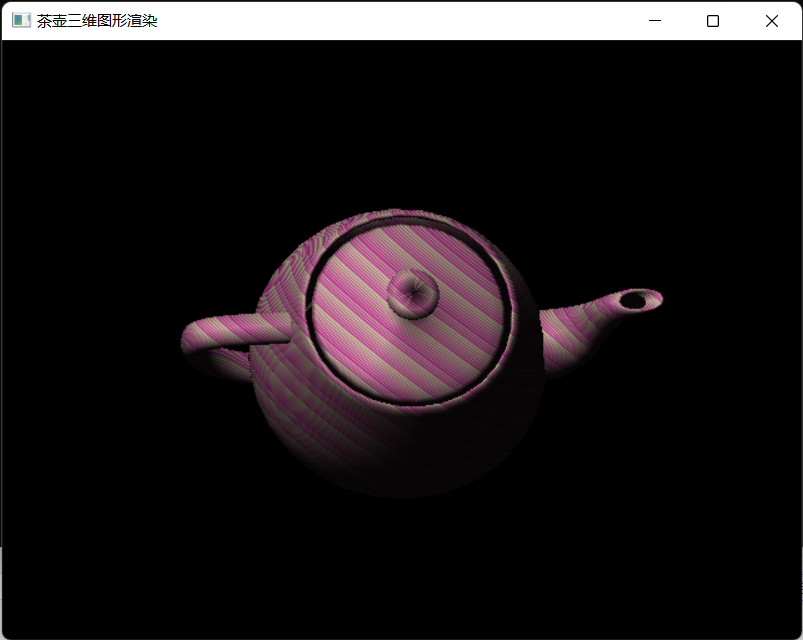
****

# 5、运行结果

****

****

****

****

实验中没有实现图形的纹理贴图，实现了光照，阴影，纹理等效果。

# 6、实验小结

通过本次综合实验，我对图形的三维渲染进行了学习，渲染过程中加入纹理、色彩、光照、阴影、透明等效果使图形更加的真实，采用光线跟踪、光照明模型、纹理贴图、纹理映射等算法使图形的渲染变得更容易。对真实感图形渲染有了一定的学习和了解。

真实感图形学是计算机图形学中的一个重要组成部分，它的基本要求就是在计算机中生成三维场景的真实感图形图像。计算机真实感图像是一种光栅图像，由象素构成。生成一幅真实感图像时，我们必须逐个象素地计算画面上相应内容表面区域的颜色。显然在计算可见景物表面区域的颜色时，不但要考虑光源对该区域入射光及光亮度和光谱组成，而且还要考虑该表面区域对光源的朝向，表面的材料和反射性质等。这种计算必须基于一定的光学物理模型——光照明模型。而光照模型恰好运用于了本次综合实验，使我们渲染的茶壶更加的真实。

# 7、参考文献

（1）OpenGL中GLSL渲染茶壶光照完整程序

**<https://blog.csdn.net/dcrmg/article/details/54024989>**

（2）[OpenGL] 茶壶与光照

<https://blog.csdn.net/zju_fish1996/article/details/51365518>

（3）OpenGL光照渲染技术

<https://www.jianshu.com/p/ad3088e158dc>

# 8、附录（实验代码）

|  |
| --- |
| //茶壶的三维图形渲染  #include <windows.h>  #include<GL/glut.h>  #include<stdio.h>  #include <stdlib.h>  #define imageWidth 60  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 TextureImage(void)  {  int j;  for (j = 0; j < imageWidth; j++)  {  stripeImage[3 \* j] = 192;  stripeImage[3 \* j + 1] = 267 / 3\* j;  stripeImage[3 \* j + 2] = 151;  }  }  /\* 参数设置 \*/  GLfloat sgenparams[] = { 1.0, 1.0, 1.0, 0.0 };  //绘制茶壶  void Draw\_Triangle(void)  {  glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);  glEnable(GL\_DEPTH\_TEST);//开启深度测试  glDepthFunc(GL\_LESS);//深度测试函数  glColor3f(1.0, 0.0, 0.0);  glLoadIdentity();//加载矩阵  glPushMatrix();//矩阵入栈  glRotatef(rote, 0.0f, 1.0f, 0.0f);  glRotatef(anglex, 1.0, 0.0, 0.0);  glRotatef(angley, 0.0, 1.0, 0.0);  glRotatef(anglez, 0.0, 0.0, 1.0);  rote += roate;  glutSolidTeapot(50);//绘制茶壶  glPopMatrix();//矩阵出栈  glutPostRedisplay();  glutSwapBuffers();  }  //光照  void reshape(void)  {  GLfloat mat\_specular[] = { 1.0, 1.0, 1.0, 1.0 };//材质的镜面反射系数  GLfloat mat\_shininess[] = { 50.0 };//材质的镜面光指数  // 光源 0  GLfloat light\_position[] = { -50.0, 100.0, 100.0, 0.0 };//光源位置  GLfloat light\_ambient[] = { 0.0, 0.0, 0.0, 1.0 };//环境光  GLfloat light\_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };//漫反射  GLfloat light\_specular[] = { 1.0, 1.0, 1.0, 1.0 };//镜面光  glClearColor(0.0, 0.0, 0.0, 0.0);  glShadeModel(GL\_SMOOTH);//光暗处理  TextureImage();//绘制纹理  //函数设定从内存中读取纹理图并放到屏幕上的方式  //指定内存中每个象素行起始的排列要求为字节排列（1）  glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);  //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  glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);  //纹理绕转使用重复方式  glTexParameterf(GL\_TEXTURE\_1D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT);  //定义纹理放大和缩小函数均为GL\_LINEAR  glTexParameterf(GL\_TEXTURE\_1D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);  glTexParameterf(GL\_TEXTURE\_1D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);  //定义纹理  glTexImage1D(GL\_TEXTURE\_1D, 0, 3, imageWidth, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, stripeImage);  //控制纹理坐标的生成  //指定单值纹理生成参数  glTexGeni(GL\_S, GL\_TEXTURE\_GEN\_MODE, GL\_OBJECT\_LINEAR);  //指定纹理坐标生成函数,系数由sgenparams指定  glTexGenfv(GL\_S, GL\_OBJECT\_PLANE, sgenparams);  glEnable(GL\_TEXTURE\_GEN\_S);//开启纹理坐标映射  glEnable(GL\_TEXTURE\_1D);//开启纹理  glEnable(GL\_LIGHT0);//开启0光源  //设置材质  glMaterialf(GL\_FRONT, GL\_SHININESS, 64.0);  glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);  glMaterialfv(GL\_FRONT, GL\_SHININESS, mat\_shininess);  glEnable(GL\_LIGHTING);//开启光照效果  //设置光照材质与位置  glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);  glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient);  glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);  glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular);  }  //图形大小  void ChangeSize(GLsizei w, GLsizei h)  {  glViewport(0, 0, (GLsizei)w, (GLsizei)h);  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  if (w <= h)  glOrtho(-100.0, 100, -100.0 \* (GLfloat)h / (GLfloat)w, 100.0 \* (GLfloat)h / (GLfloat)w, -1000.0, 1000.0);  else  glOrtho(-100.0 \* (GLfloat)w / (GLfloat)h, 100.0 \* (GLfloat)w / (GLfloat)h, -100.0, 100.0, -1000.0, 1000.0);  glMatrixMode(GL\_MODELVIEW);  glLoadIdentity();  }  //鼠标函数  void mouse(int button, int state, int x, int y) // 鼠标函数  {  if (button == GLUT\_LEFT\_BUTTON)  {  if (state == GLUT\_DOWN)  {  roate = 0;  rote = 0;  oldx = x;//当左键按下时记录鼠标坐标  oldy = y;  }  }  if (button == GLUT\_RIGHT\_BUTTON)  {  if (state == GLUT\_DOWN)  {  roate += 1.0f;  }  }  }  //鼠标旋转函数  void motion(int x, int y)  {  GLint deltax = oldx - x;  GLint deltay = oldy - y;  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  angley += 360 \* (GLfloat)deltay / (GLfloat)WinH;  anglez += 360 \* (GLfloat)deltay / (GLfloat)WinH;  oldx = x;//记录此时的鼠标坐标，更新鼠标坐标  oldy = y;//若是没有这两句语句，滑动是旋转会变得不可控  glutPostRedisplay();  }  int main(int argc, char\*\* argv)  {  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);  glutInitWindowSize(640, 480);  glutInitWindowPosition(100, 150);  glutInit(&argc, argv);  glutCreateWindow("茶壶三维图形渲染");  glutDisplayFunc(Draw\_Triangle);  glutReshapeFunc(ChangeSize);  glutMouseFunc(mouse);  glutMotionFunc(motion);  reshape();  glutMainLoop();  return 0;  } |