**《计算机图形学实验》综合实验报告**

**题目：基于Opengl实现三维茶壶的渲染**

**学 号：20201120486**

**姓 名：李奥**

**指导教师：钱文华**

**日 期:2022年6月**

目录

**[关键字：OpenGL，纹理，三维图形，渲染。 3](#_Toc32571)**

**[一、 实验背景及内容 3](#_Toc16840)**

**[二、 开发工具 4](#_Toc2038)**

[2.1开发程序截图： 4](#_Toc857)

[2.2开发工程文件截图： 4](#_Toc288)

**[三、 程序介绍 4](#_Toc27370)**

[3.1 程序核心算法： 4](#_Toc11100)

[3.2程序设计流程： 5](#_Toc15111)

**[四、 运行结果 6](#_Toc31155)**

**[控制键盘，将带有纹理的茶壶向x,y,z三个方向旋转： 7](#_Toc4756)**

**[y方向： 7](#_Toc12148)**

**[z方向： 8](#_Toc12214)**

**[1.2运行结果分析： 8](#_Toc16890)**

**[此案例能够基本实现预期任务，但是： 8](#_Toc19202)**

**[1. 茶壶在键盘控制旋转时，转动的速率较快，难以观察转动轨迹。 8](#_Toc17824)**

**[2. 茶壶的纹理不是特别精致，需要在后期的学习中进行加工。 8](#_Toc21782)**

**[3. 光照处理中，暗处理部分相对较好，但是光照程度不够，可以观察的视野过于狭小。 8](#_Toc10099)**

**[4. 纹理坐标自动生成，是本次实验的一个优点，但在精度的处理上仍需要更好的完善。 8](#_Toc29195)**

**[五、 实验总结： 8](#_Toc16742)**

**[六、 参考文献 8](#_Toc10591)**

**[七、 附录（代码材料） 9](#_Toc1286)**

**[实验代码展示： 9](#_Toc13897)**

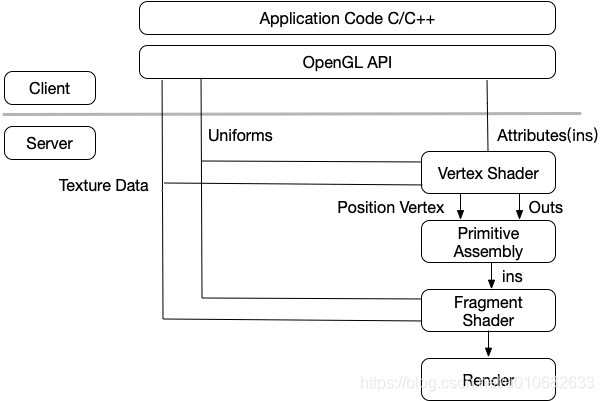
摘 要：根据所学知识可知，OpenGL规范描述了绘制2D和3D图形的抽象API。OpenGL不仅语言无关，而且平台无关。规范只字未提获得和管理OpenGL上下文相关的内容，而是将这些作为细节交给底层的**[窗口系统](https://baike.so.com/doc/28314975-29734619.html" \t "https://baike.so.com/doc/_blank)**。出于同样的原因，OpenGL纯粹专注于渲染，而不提供输入、音频以及窗口相关的API。本次实验基于OpenGL，实现了三维图象-茶壶的纹理贴图，使用了纹理贴图和消隐相关的算法。

关键字：OpenGL，纹理，三维图形，渲染。

1. **实验背景及内容**

本次实验的渲染具有一定的流程：

[OpenGL](https://so.csdn.net/so/search?q=OpenGL&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/u010682633/article/details/_blank) ，被定义为“图形硬件的一种软件接口”。从本质上说，它是一个3D图形和模型库，具有高度的可移植性，具有非常快的速度。可视化技术、三维地形等领域方面在全球取得了巨大的进步,都是基于计算机图形技术的迅速发展的基础上实现的。也正是有了计算机相关的硬件以及相关技术有了迅速发展,使得三维图形建模、可视化技术相关的研究越来越广泛,越来越多的大、中、小型企业和高等院校也加入其行列,并且取得的成果也是丰富的。OpenGL可以进行图像的渲染，渲染使用客户端<——>服务端的形式实现，客户端是我们编写的代码，服务端是计算机图形硬件厂商所提供的OpenGL实现。

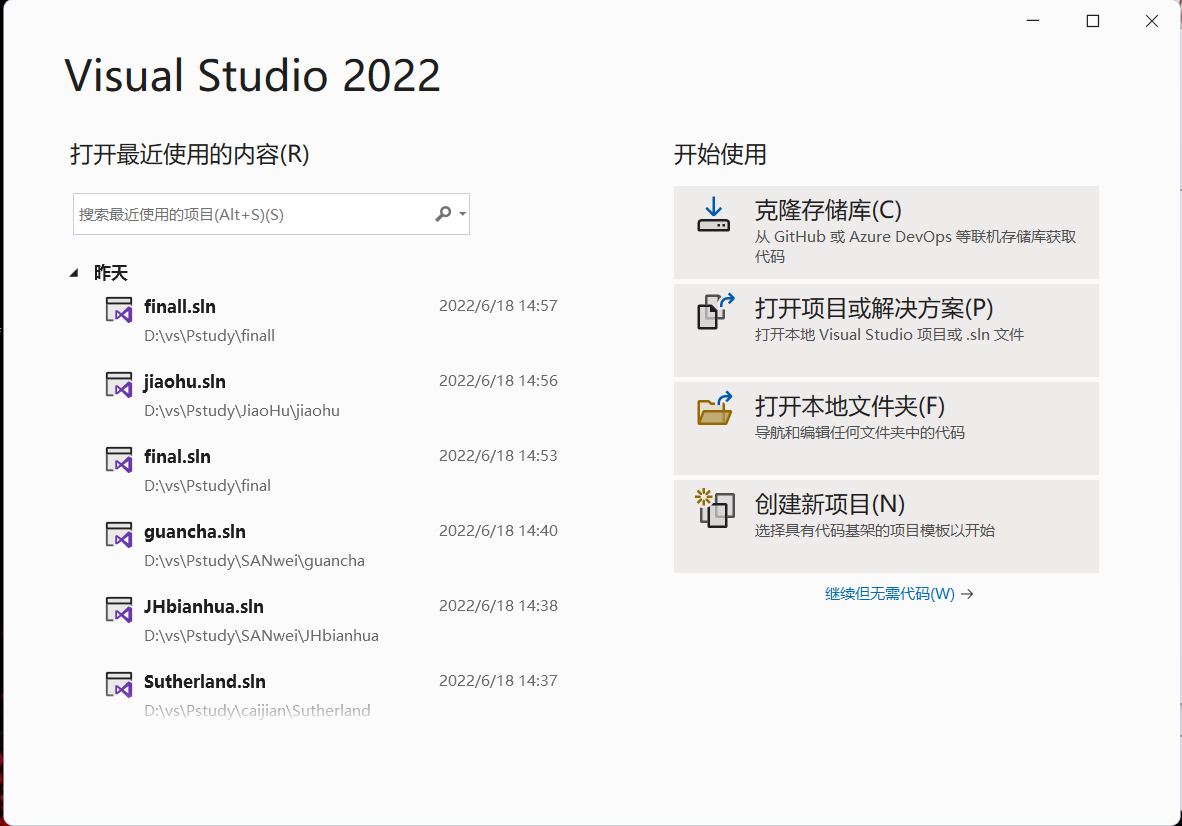


本次实验，是在之前实现交互式茶壶的基础上，对三维立体的茶壶进行光照，加入纹理，进行消隐等相关操作，目的是为了对本学期的计算机图形学实验课有一个更好的总结。

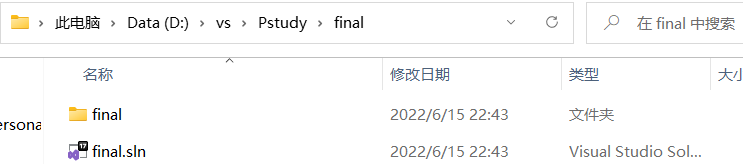
1. **开发工具**

本次实验中使用visual studio 2022版本，安装nupengl程序包，可以进行C++程序的编写，从而实现对OpenGL的应用。

2.1开发程序截图：



2.2开发工程文件截图：



1. **程序介绍**

3.1 程序核心算法：

void makeTexture(void)：这个算法，可以实现纹理的添加，包含紫色纹理，绿色纹理，紫绿色镶嵌纹理

void Light(void)：这个算法可以实现对光照的处理，包含各种光种，以及光暗，材质的处理。

void keyboard(unsigned char key, int x, int y)：这个算法实现了键盘的交互，可以通过键盘输入x，y，z的值将图像经行响应方向的旋转（右手螺旋定则方向）

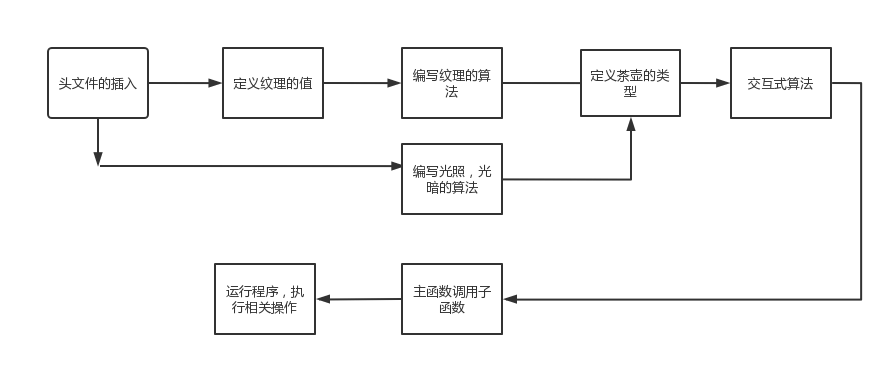
void idle()：该算法控制旋转的弧度大小

void myinit()该算法实现对纹理的显示，主要包含：

|  |  |
| --- | --- |
| 创建纹理： | makeTexture();  glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1); |
| 控制纹理: | glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);  glTexParameterf(GL\_TEXTURE\_1D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT);  glTexParameterf(GL\_TEXTURE\_1D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);  glTexParameterf(GL\_TEXTURE\_1D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);  glTexImage1D(GL\_TEXTURE\_1D, 0, 3, TEXTUREWIDTH, 0,  GL\_RGB, GL\_UNSIGNED\_BYTE, Texture); |
| 纹理的方向S: | glTexGeni(GL\_S, GL\_TEXTURE\_GEN\_MODE, GL\_OBJECT\_LINEAR);  glTexGenfv(GL\_S, GL\_OBJECT\_PLANE, sgenparams); |
| 启用纹理: | glEnable(GL\_TEXTURE\_1D);  glEnable(GL\_TEXTURE\_GEN\_S); |
| 启用消隐: | glEnable(GL\_DEPTH\_TEST);  glDepthFunc(GL\_LESS);  glDepthFunc(GL\_LESS); |
| 一些绘图控制: | glEnable(GL\_CULL\_FACE);  glEnable(GL\_LIGHTING);  glEnable(GL\_LIGHT0);  glEnable(GL\_AUTO\_NORMAL);  glEnable(GL\_NORMALIZE);  glFrontFace(GL\_CW);  glCullFace(GL\_BACK);  glMaterialf(GL\_FRONT, GL\_SHININESS, 64.0); |

接下来就在主函数中调用相应的算法即可得到程序需要的显示图像。

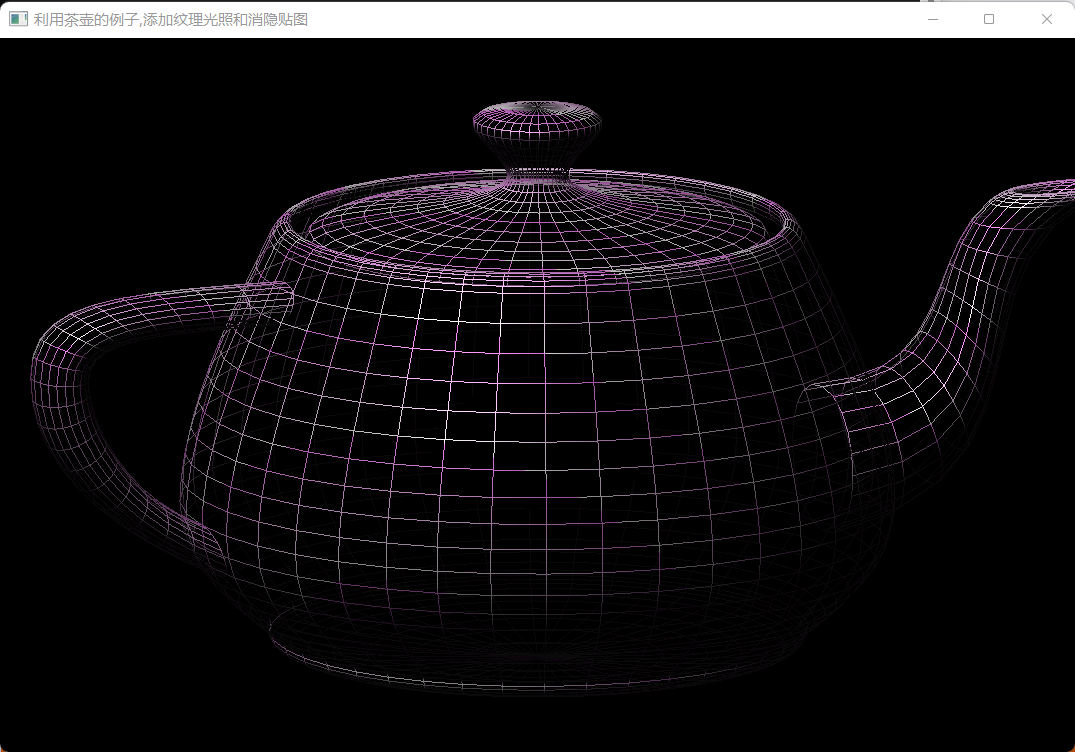
3.2程序设计流程：



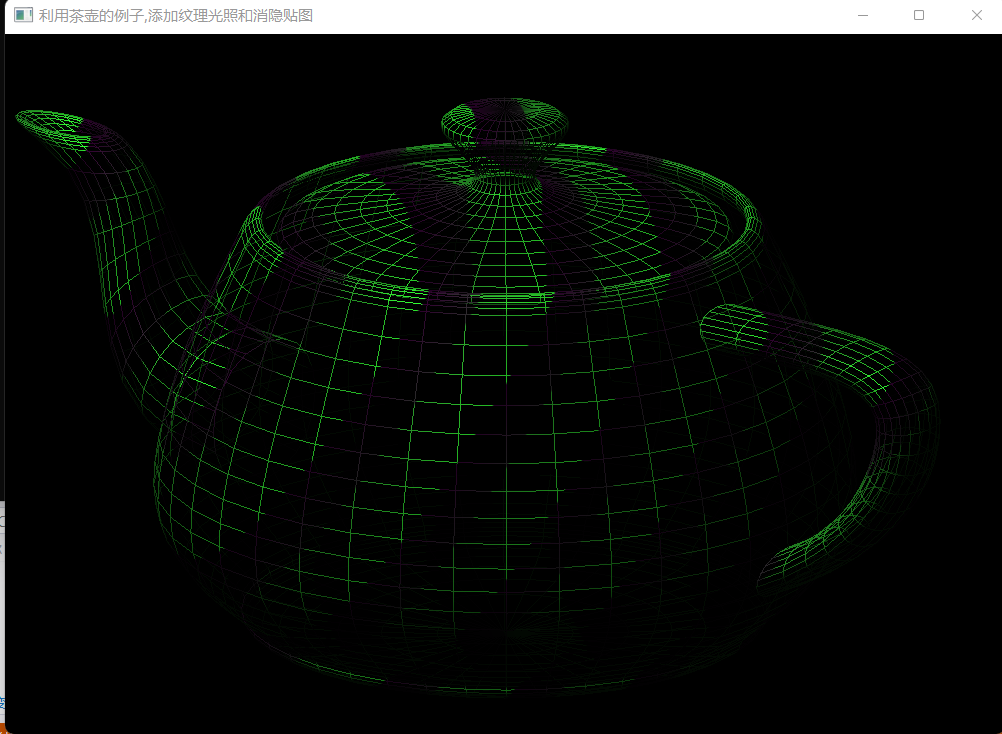
1. **运行结果**

1.1运行结果展示

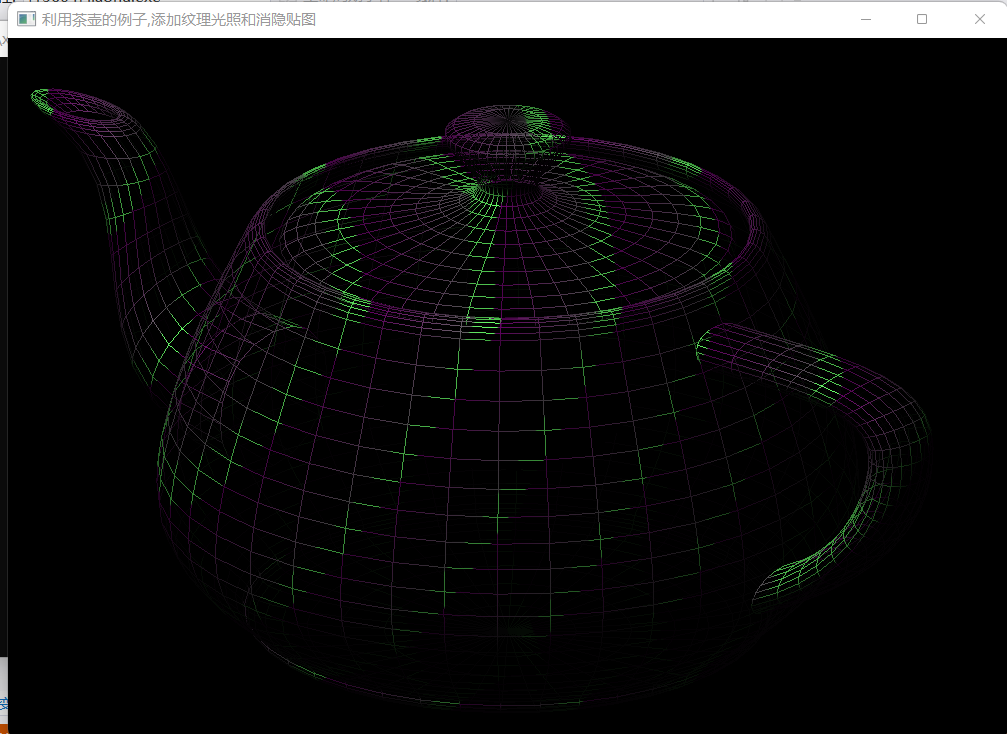
1. 茶壶紫色纹理：



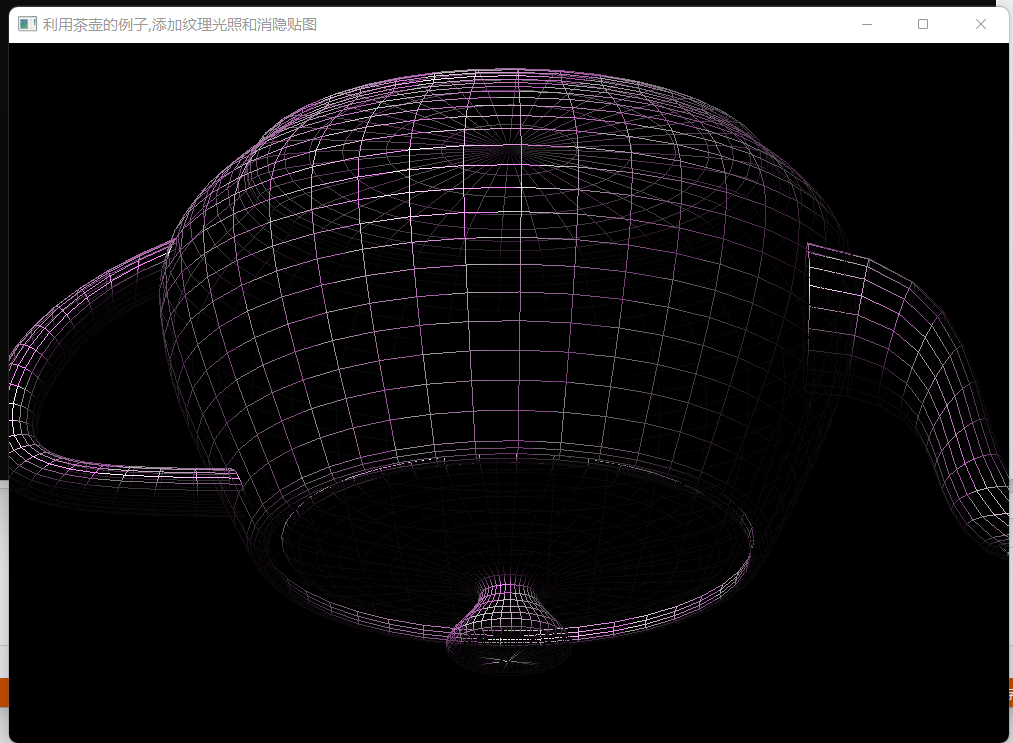
1. 茶壶绿色纹理：



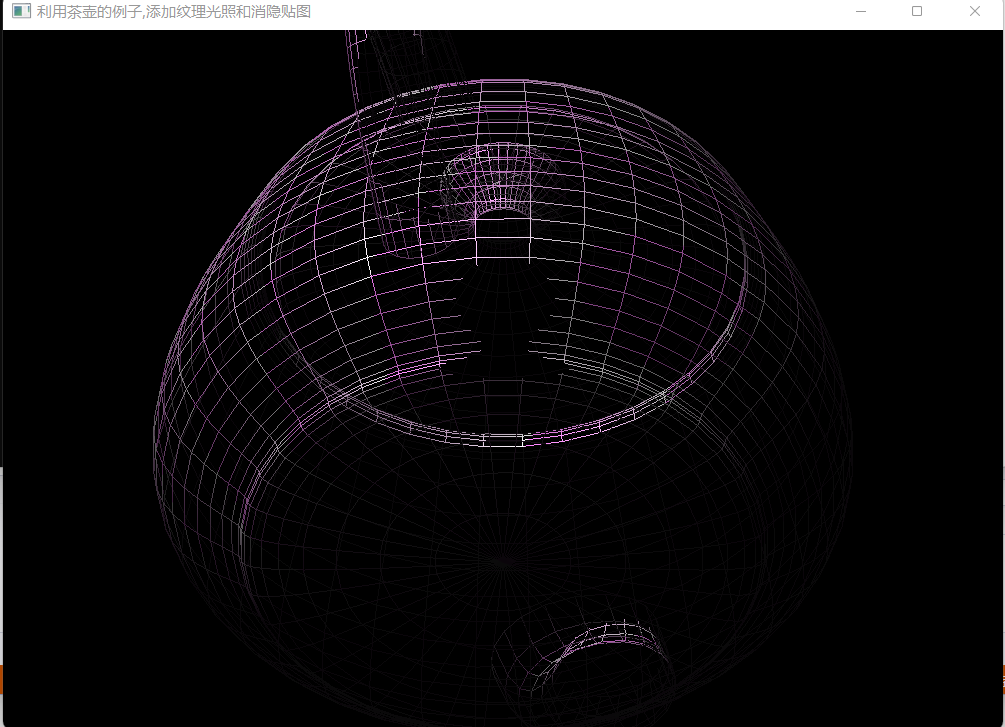
1. 茶壶紫色和绿色交叉纹理：



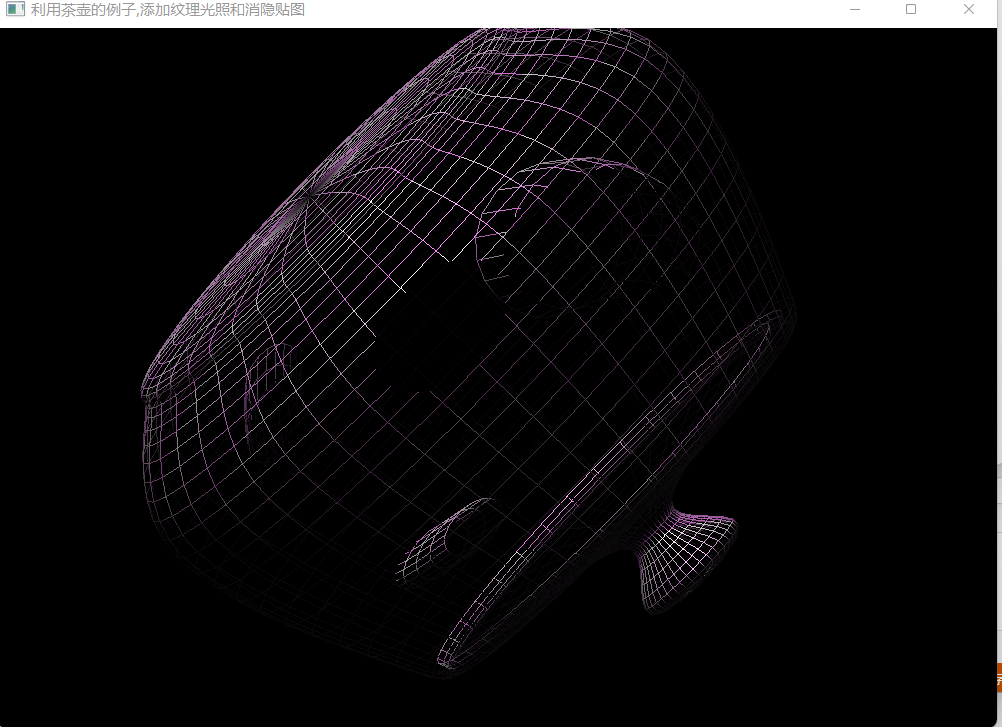
控制键盘，将带有纹理的茶壶向x,y,z三个方向旋转：

x方向：  


y方向：



z方向：



1.2运行结果分析：

此案例能够基本实现预期任务，但是：

1. 茶壶在键盘控制旋转时，转动的速率较快，难以观察转动轨迹。
2. 茶壶的纹理不是特别精致，需要在后期的学习中进行加工。
3. 光照处理中，暗处理部分相对较好，但是光照程度不够，可以观察的视野过于狭小。
4. 纹理坐标自动生成，是本次实验的一个优点，但在精度的处理上仍需要更好的完善。
5. **实验总结：**

本次实验学习到了交互式控制茶壶，然后在茶壶中增添纹理等一系列的操作，收益良多，也激发了我对真实感图形的更深入的了解，第一节课上老师就提醒我们，“我在重复这句话的时候可能已经是17周了”，真的感觉时间飞逝，我学到的东西也甚是浅薄，但是一个学期已经来到了尾声。通过这学期的学习，认识到了OpenGL的巨大魅力所在，我们做了许许多多的实验，例如裁剪，反走样，甚至最基础的划线，DDA，bresenham算法等等，这些基础都会向我们指明一个方向，想着想学习的方向继续前进。

本次完成的实验难度较低，仅只是对实验完成了纹理和光照添加，消隐等简单的处理，但这也会激起我对真实感图形研究的热情，本次实验采用c++编程，但很多语法和c语言类似，所以容易上手。感谢图形学老师和助教老师一个学期的耐心指导，愿在今后的学习中，取得更好的进步。

1. **参考文献**
2. OpenGL ES 3.0编程指南 Dan [Ginsburg](http://search.dangdang.com/?key2=Ginsburg&medium=01&category_path=01.00.00.00.00.00" \t "http://product.dangdang.com/_blank), [Budirijanto](http://search.dangdang.com/?key2=Budirijanto&medium=01&category_path=01.00.00.00.00.00" \t "http://product.dangdang.com/_blank) [Purnomo](http://search.dangdang.com/?key2=Purnomo&medium=01&category_path=01.00.00.00.00.00" \t "http://product.dangdang.com/_blank) 出版社:[机械工业出版社](http://search.dangdang.com/?key3=%BB%FA%D0%B5%B9%A4%D2%B5%B3%F6%B0%E6%C9%E7&medium=01&category_path=01.00.00.00.00.00" \t "http://product.dangdang.com/_blank)
3. CHENG Peng-gen, GONG Jian-ya, SHTW en-zhong, et al. Geological object modeling basedon quasitriprism volume and its application.Geometrics and Information Science ofWuhan University[J].2004, 8(3): 330-350.
4. Dave Shreiner,Mason Woo, Jackie Neider, et al. OpenGL Programming Guide, FifthEditionLM].机械工业出版社,2006.
5. **附录（代码材料）**

实验代码展示：

#include<stdlib.h>

#include<GL/glut.h>

#include<GL/GL.h>

#include<math.h>

float theta[] = { 0, 0, 0, 0 };

int axis = 3;

float step = 2.0;

int win\_w, win\_h, mx, my;

#define TEXTUREWIDTH 64

GLubyte Texture[3 \* TEXTUREWIDTH];

//定义紫色，绿色，紫绿色的渐变纹理（紫色：255，绿色：50，紫绿色：150）（texture[]值）

void makeTexture(void)

{

int i;

for (i = 0; i < TEXTUREWIDTH; i++)

{

Texture[3 \* i] = 255;

Texture[3 \* i + 1] = 255 - 2 \* i;

Texture[3 \* i + 2] = 255;

}

}

GLfloat sgenparams[] = { 1.0,1.0,1.0,0.0 };

void Light(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

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);

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT| GL\_DEPTH\_BUFFER\_BIT);

glColor3f(1.0, 0.0, 0.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glRotatef(theta[0], 1.0, 0, 0);

glRotatef(theta[1], 0, 1.0, 0);

glRotatef(theta[2], 0, 0, 1.0);

glutWireTeapot(2);

glutSwapBuffers();

}

void keyboard(unsigned char key, int x, int y)

{

//该步骤是利用键盘输入x，y，z

switch (key)

{

case 'x':

axis = 0;

step = 2.0;

break;

case 'X':

axis = 0;

step = -2.0;

break;

case 'y':

axis = 1;

step = 2.0;

break;

case 'Y':

axis = 1;

step = -2.0;

break;

case 'z':

axis = 2;

step = 2.0;

break;

case 'Z':

axis = 2;

step = -2.0;

break;

case 27:

exit(0);

}

}

void idle()

{

theta[axis] += step;

if (theta[axis] >= 360)

theta[axis] -= 360;

if (theta[axis] < 0)

theta[axis] += 360;

glutPostRedisplay();

}

void move(int x, int y)

{

if (x < mx)

theta[1] += 2.0;

else if (x > mx)

theta[1] -= 2.0;

if (y < my)

theta[0] -= 2.0;

else if (y > my)

theta[0] += 2.0;

mx = x, my = y;

glutPostRedisplay();

}

void reshape(int w, int h)

{

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if (w <= h)

glOrtho(-2.0, 2.0, -2.0 \* h / w, 2.0 \* h / w, -2.0, 2.0);

else

glOrtho(-2.0 \* w / h, 2.0 \* w / h, -2.0, 2.0, -2.0, 2.0);

glViewport(0, 0, w, h);

win\_w = w, win\_h = h;

glMatrixMode(GL\_MODELVIEW);

}

void myinit()

{

glEnable(GL\_DEPTH\_TEST);

glEnableClientState(GL\_COLOR\_ARRAY);

glEnableClientState(GL\_VERTEX\_ARRAY);

my = win\_h / 2;

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGBA);

glutInitWindowPosition(500, 500);

glClearColor(0.0, 0.0, 0.0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

// 创建纹理

makeTexture();

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

// 控制纹理

glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);

glTexParameterf(GL\_TEXTURE\_1D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT);

glTexParameterf(GL\_TEXTURE\_1D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_1D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexImage1D(GL\_TEXTURE\_1D, 0, 3, TEXTUREWIDTH, 0,

GL\_RGB, GL\_UNSIGNED\_BYTE, Texture);

// 唯一与前面例子不同的地方：启用纹理坐标自动产生，生成环境纹理

// 纹理的方向S

glTexGeni(GL\_S, GL\_TEXTURE\_GEN\_MODE, GL\_OBJECT\_LINEAR);

glTexGenfv(GL\_S, GL\_OBJECT\_PLANE, sgenparams);

// 启用纹理

glEnable(GL\_TEXTURE\_1D);

glEnable(GL\_TEXTURE\_GEN\_S);

// 启用消隐

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LESS);

glDepthFunc(GL\_LESS);

// 一些绘图控制

glEnable(GL\_CULL\_FACE);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_AUTO\_NORMAL);

glEnable(GL\_NORMALIZE);

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glMaterialf(GL\_FRONT, GL\_SHININESS, 64.0);

// glShadeModel(GL\_FLAT);

}

int main(int argc, char\* argv[])

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowPosition(500, 500);

glutInitWindowSize(1000, 700);

glutCreateWindow("利用茶壶的例子,添加纹理光照和消隐贴图");

myinit();

glutDisplayFunc(display);

Light();

glutReshapeFunc(reshape);

glutIdleFunc(idle);

glutKeyboardFunc(keyboard);

glutMotionFunc(move);

glutMainLoop();

return 0;

}