《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

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**。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。

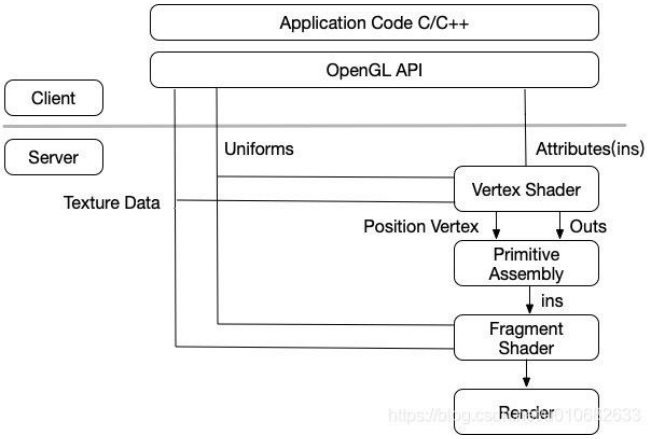
关键字：OpenGL，纹理，三维图形，渲染。

目录

一、实验背景及内容

本次实验的渲染具有一定的流程：

OpenGL，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。

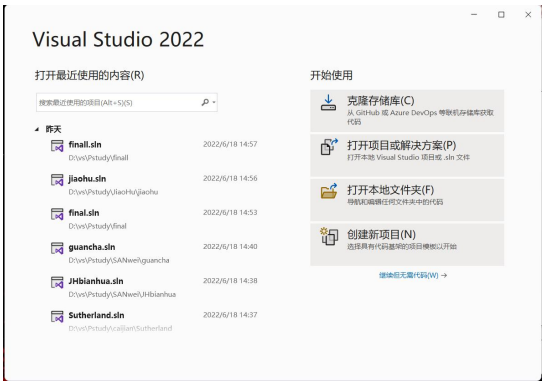


本次实验,是在之前实现交互式茶壶的基础上,对三维立体的茶壶进行光照,加入纹理,进行消隐等相关操作，目的是为了对本学期的计算机图形学实验课有一个更好的总结。

一、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：

二、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,

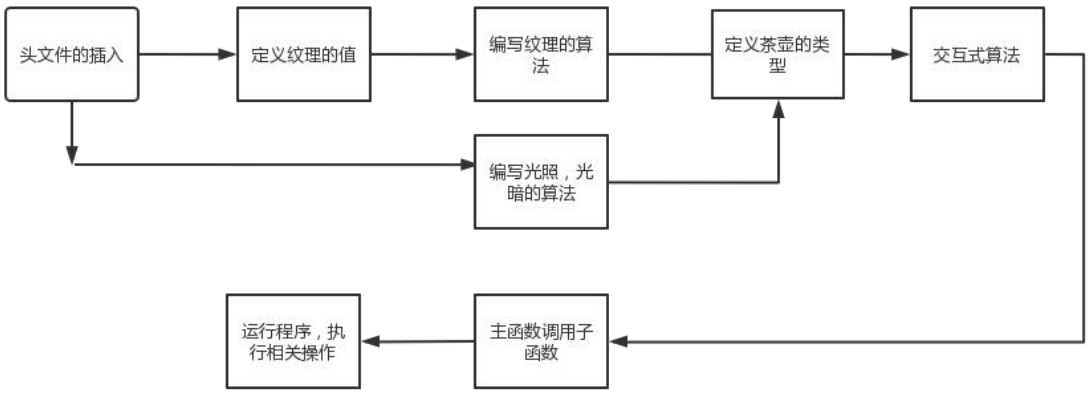
	Texture);
纹理的方向 S:	glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR); glTexGenfv(GL_S, GL_OBJECT_PLANE, sgenparams);
启用纹理:	glEnable(GL_TEXTURE_1D); glEnable(GL_TEXTURE_GEN_S);

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。

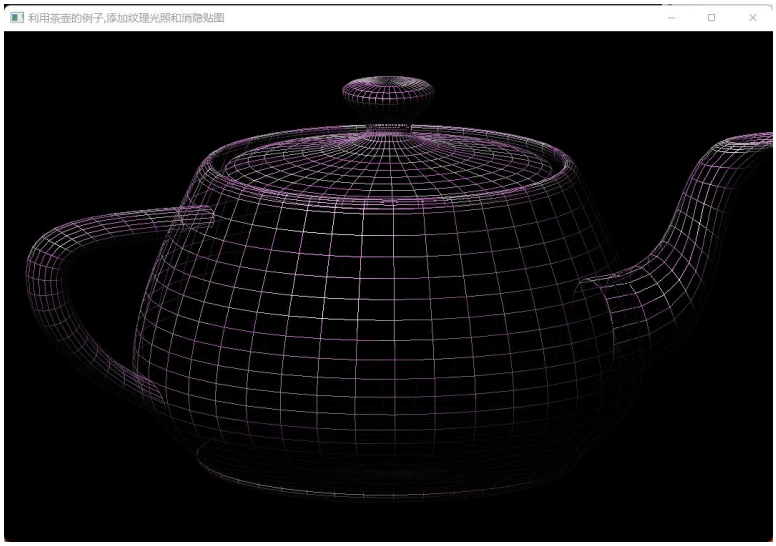
3.2 程序设计流程：



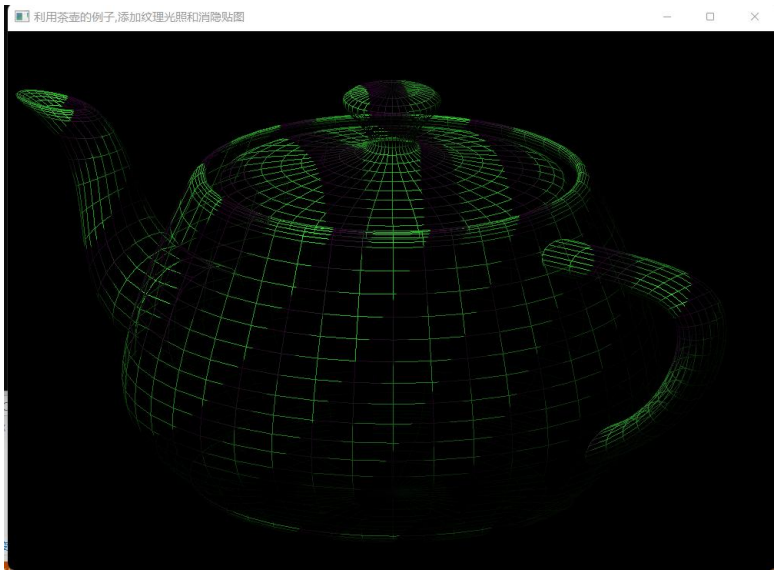
三、运行结果

1.1 运行结果展示

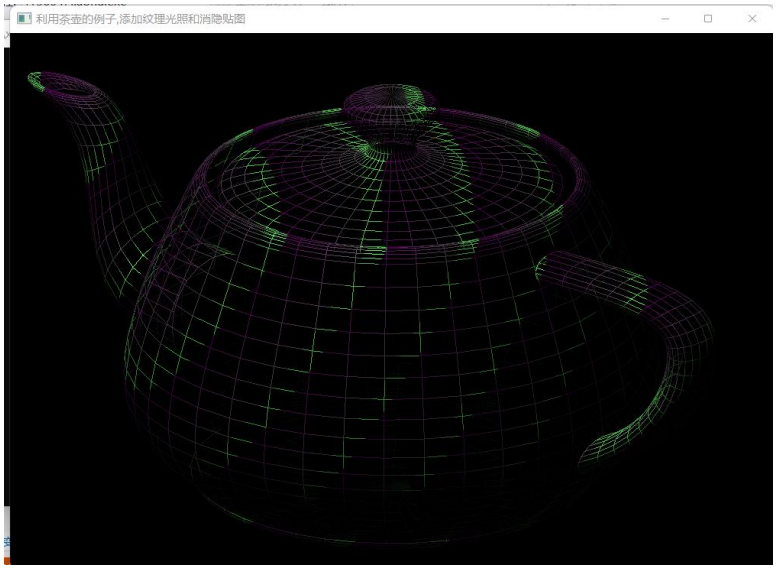
1. 茶壶紫色纹理：



2. 茶壶绿色纹理：

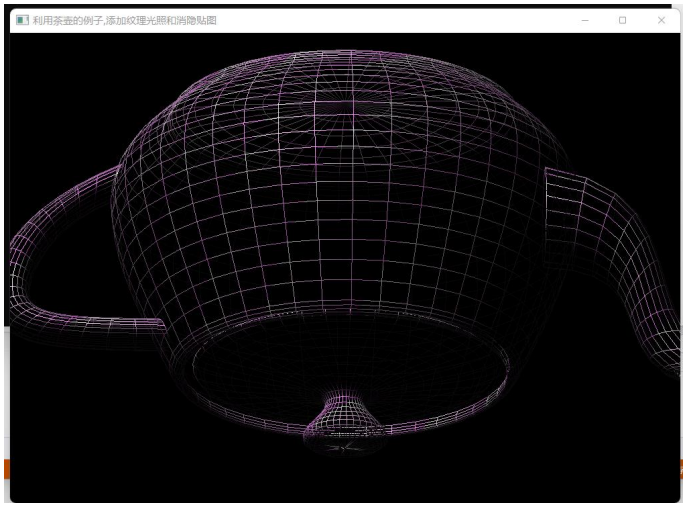


3. 茶壶紫色和绿色交叉纹理：

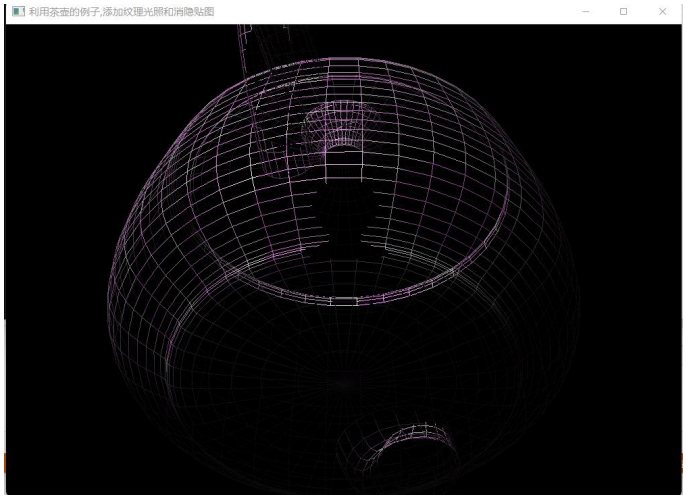


控制键盘，将带有纹理的茶壶向 x, y, z 三个方向旋转：

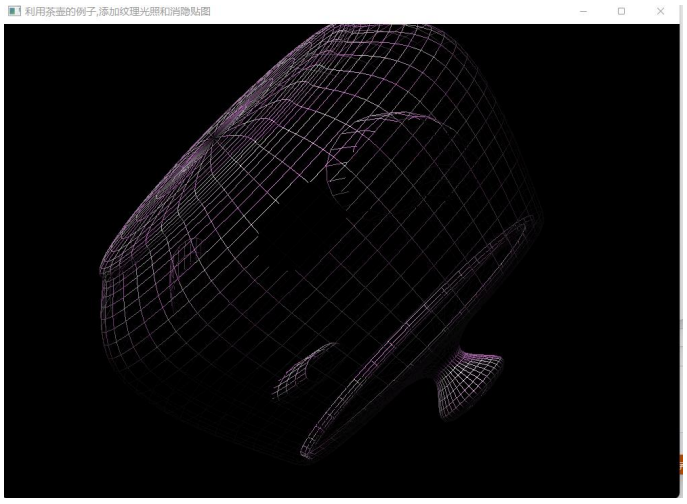
x 方向：



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。

四、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。

五、参考文献

[1] OpenGL ES 3.0 编程指南 Dan Ginsburg, Budirijanto Purnomo 出版社:机械工业出版社

[2] 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.

[3] Dave Shreiner, Mason Woo, Jackie Neider, et al. OpenGL Programming Guide, FifthEditionLM]. 机械工业出版社, 2006.

六、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;
}
```