《计算机图形学实验》综合实验报告

题目 基于 Opengl 实现对三维茶壶的渲染

学 号 20201060386

姓 名 李运

指导教师 钱文华

日 期 2022 年 6 月 18 日

摘要

三维渲染是将三维场景中的模型，按照设定好的环境、灯光、材质及渲染参数。渲染是三维计算机图形学中的最重要的研究课题之一，并且在实践领域它与其它技术密切相关。在图形流水线中，渲染是最后一项重要步骤，通过它得到模型与动画最终显示效果。自从二十世纪七十年代以来，随着计算机图形的不断复杂化，渲染也越来越成为一项重要的技术。

本实验通过利用 opengl 实现对三维茶壶的渲染，在渲染过程加入了纹理、色彩、光照、阴影等效果。使用 glutSolidTeapot（）函数画出茶壶，后面一步一步地添加光照、阴影、色彩和纹理给茶壶润色。在添加纹理的过程中，采用了自动生成纹理坐标的方法给茶壶添加上了纹理；光照阴影部分则采用了漫反射光和镜面反射光给茶壶添加光照阴影的效果，定义光源即光源的位置、光源的属性、颜色等，定义完光源后开启光源则可以对茶壶有光照和阴影的渲染效果。

关键词：三维图形渲染；Opengl；纹理；光照；阴影

目录

一、 实验背景 4
二、 实验内容 5
三、 程序设计 5
四、 实验步骤 6
五、 结果分析 8
六、 总结体会 9
参考文献 10
附录 10

一、 实验背景

随着计算机软、硬件突飞猛进的发展，计算机图形学在各个行业的应用也得到迅速普及和深入。随着几十年图形学的发展，计算机已进入三维时代，三维图形在人们周围无所不在。科学计算可视化、计算机动画和虚拟现实已经成为近年来计算机图形学的三大热门话题，计算机图形学领域的研究层次逐步深入，并以越来越快的速度向前发展。

在计算机中，由于用图形表达各种信息，其容量大、直观方便，更符合人们观察了解事物运动规律的习惯，所以三维图形技术在建筑虚拟、城市规划、场景漫游、效果场景制作、城市规划、房地产开发、虚拟教育、展馆展示、古迹复原、交通线路设计、3D 游戏等各方面都有广泛的实际应用。

特别是近几年，中国的网络游戏产业目前正处于一个迅猛发展的时期，网络游戏主流也朝 3D 游戏发展。因此研究三维图形引擎中的关键技术是一次有意义的尝试。目前该领域主要研究方向是使用数学算法将二维或三维图形数据转化为计算机显示器的栅格形式，包括图形硬件、图形标准、图形交互技术、光栅图形生成算法、曲线曲面造型、实体造型、真实感图形计算。场景渲染是三维引擎中最重要的子模块之一，负责实现基本图元的绘制，光线处理和纹理处理等，实际上是三维真实感图形的再现过*程*[1]。

二、 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。

三、 程序设计

1、 开发工具

基于 codeblocks 平台，利用 Opengl 实现对三维茶壶的渲染。

2、 程序设计

首先使用 glutSolidTeapot（）函数画出茶壶，后面一步一步地添加光照、阴影、色彩和纹理给茶壶润色。在添加纹理的过程中，采用了自动生成纹理坐标的方法给茶壶添加上了纹理；光照阴影部分则采用了漫反射光和镜面反射光给茶壶添加光照阴影的效果，定义光源即光源的位置、光源的属性、颜色等，定义完光源后开启光源则可以对茶壶有光照和阴影的渲染效果。

3、 实现目的

根据计算机图形学所学知识利用 Opengl 画一个茶壶，并给茶壶

添加一些光照、色彩、纹理等。最后撰写实验报告，完成实验所给要求。

4、 模块介绍

（1） void makeStripeImage()函数：贴图纹理函数的初始化。设置纹理的颜色以及宽度等参数。

（2） void init()函数：初始化函数。里面了定义了光源的实现和自动生成纹理坐标、纹理和物体表面颜色处理方式（和物体表面颜色做与运算）、设置自动生成纹理坐标的计算方式以及激活纹理坐标生成等。

（3） void display()函数：绘制图形函数。设置茶壶的方向、大小、位置以及纹理生成等。

（4） void reshape()函数：修正窗内的图形显示函数。处理茶壶移动等引起的变化。

四、 实验步骤

1、 算法理论

光照：

镜面反射：当物体的表面很光滑的时候，在一定的角度范围内观察这个平滑的表面表面时，能够看到一个高光的效果。如在用光源照射金属时，在一定的角度内可以看到一个有一定大小的光点，但偏离

一定的角度之后，就不能再看到这个高光效果。则具体方法可以是冯氏光照模型：可以用反射光的方向向量和观察方向的方向向量做点乘，得到其夹角的余弦值，点乘的结果越大，则反射光和观察方向就越近。

漫反射：在建立表面的漫反射模型时假设入射光在各个方面以相同强度发散而与观察位置无关，这样的表面称为理想漫反射体。可以用朗伯余弦定律来表示，该定律表明：在与对象表面法向量夹角为ϕ_N的方向上，每个面积为dA的平面单位所散发的光线与$cos\phi_N$成正比。

纹理：

纹理既包括通常意义上物体表面的纹理即使物体表面呈现凹凸不平的沟纹，同时也包括在物体的光滑表面上的彩色图案，称之为花纹。在顶点中显式指定的纹理坐标一样，可以使用自动生成的纹理坐标作为纹理坐标转换的输入。在许多情况下，系统生成的纹理坐标可与转换一起使用，以产生特殊效果[2]。

2、 关键代码：

（1） void makeStripeImage(void)：定义贴图纹理，定义纹理的形状以及颜色等，为后续自动生成纹理坐标做准备。

（2） void init(void)：初始化。其中 GLfloat light_position[] = { -200.0,250.0,400.0,0.0 }; //定义（点）光源的位置；GLfloat light_ambient[] = { 0.1f,0.1f,0.1f,1.0f }; // 定义环境光；glEnable(GL_LIGHTING); // 开启光源；

glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);//和物体表面颜色做与运算（设置纹理和物体表面颜色处理方式）；

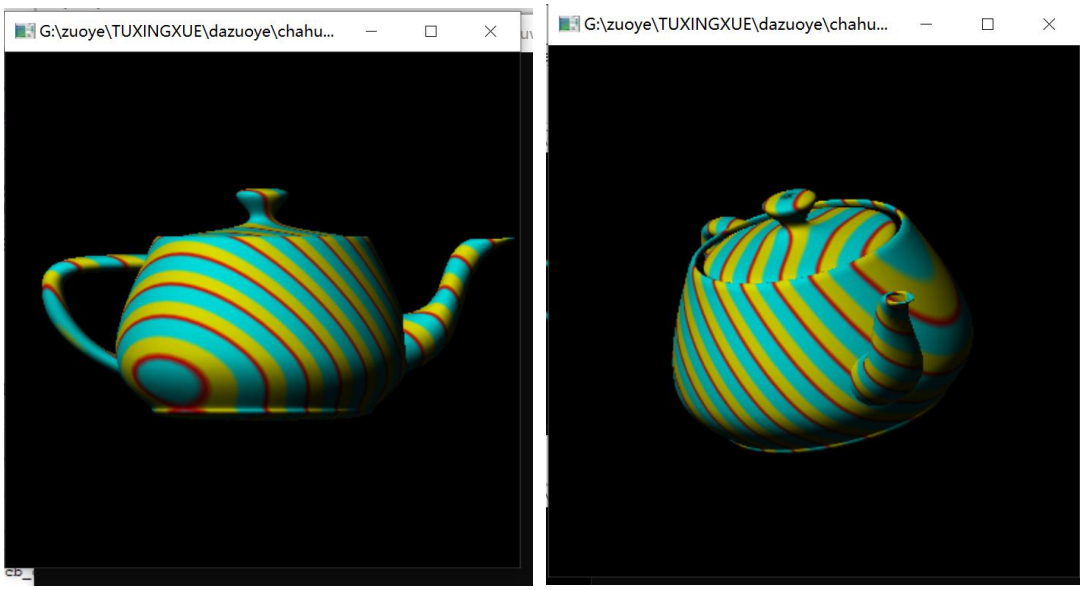
glEnable(GL_TEXTURE_GEN_S);//激活纹理坐标生成。

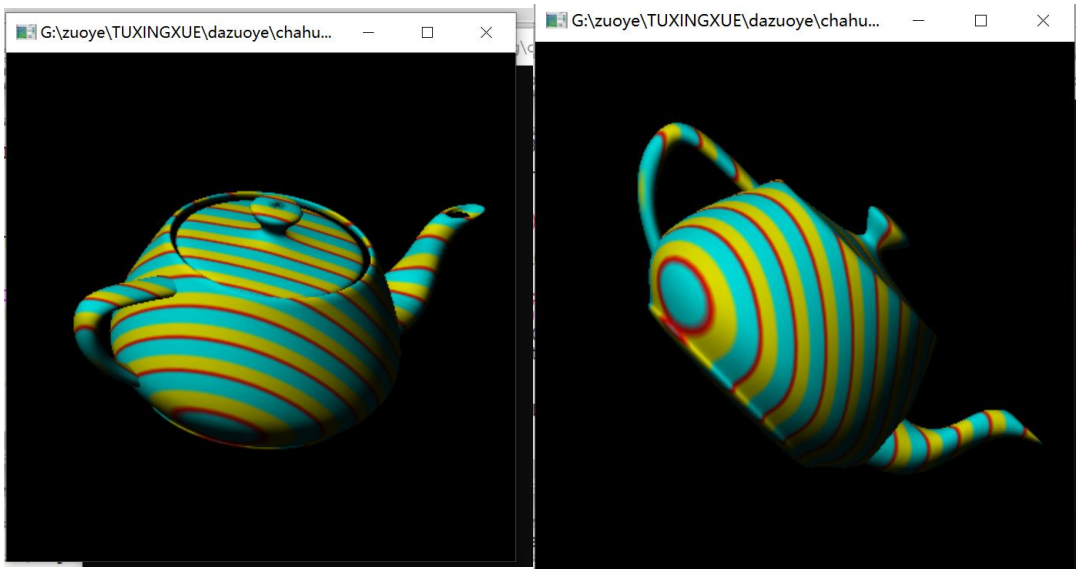
（3） glRotatef(-70.0, 0.0, 0.0, 1.0);//设置茶壶的角度

（4） glutSolidTeapot(2.0); //绘制茶壶

五、 结果分析

1、 运行结果





2、 结果分析

从运行结果可知，我们可以看到给茶壶添加上了光照和纹理效果，同时可以看到茶壶各个不同的面的的效果图。并且可以通过改变参数获得不同形状纹理色彩以及不同光照角度的茶壶。

六、 总结体会

通过本次实验，我学习了如何给三维图形添加色彩、光照、阴影、纹理等效果，并且通过本次学习，对我以后深入学习计算机图形学有很大的帮助。在实验过程中，我刚开始对如何渲染三维图形还不是很了解，可是通过上网查询相关知识之后有很大的感触，并且通过网上以及课本的上的学习，一步一步地给茶壶添加上了光照纹理等效果，最终顺利地完成了实验。

参考文献

[1]. 百 度 文 库 ：https://wenku.baidu.com//view/774abced8aeb172ded630b1c59eef8c75fbf95da?fr=wklm_a8790fe5da38376baf1fae46_200040009. 浏览时间 2022-6-18.

[2].张志华,基于 OpenGl 的三维模型渲染算法研究[J],2011,2:114-117.

附录

```
#include <windows.h>
#include <GL/glut.h>
#include <stdio.h>
#include <stdlib.h>
#define stripeImageWidth 10//设置贴图的宽度为 10

//定义全局变量
GLubyte stripeImage[4*stripeImageWidth];
static GLuint texName;
static GLfloat xequalzero[] = {1.5, 1.5, 1.5, 0.0};
static GLfloat *currentCoeff;
```

```
static GLenum currentPlane;
static GLint currentGenMode;
//贴图纹理
void makeStripeImage(void)
{
    int j;
    for (j = 0; j < stripeImageWidth; j++) {
        stripeImage[4*j] = (GLubyte) ((j<=4) ? 255 : 0);
        stripeImage[4*j+2] = (GLubyte) ((j>4) ? 255 : 0);
        stripeImage[4*j+4] = (GLubyte) 0;
        stripeImage[4*j+5] = (GLubyte) 255;
    }
}

//初始化函数
void init(void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glEnable(GL_DEPTH_TEST);
    glShadeModel(GL_SMOOTH);

    // 定义光源
```

```
    GLfloat  light_position[] = { -200.0,250.0,400.0,0.0 };
//  （点）光源的位置
    GLfloat light_ambient[] = { 0.1f,0.1f,0.1f,1.0f };  // 环
境光
    GLfloat light_diffuse[] = { 1.0f, 1.0f, 1.0f, 1.0f };  //
漫反射光，白色
    GLfloat  light_specular[] = { 0.9f,0.4f,0.6f,1.0f };  //
镜面反射光，稍浅的棕色
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);//指定
光源的位置
    glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);//光源中
的环境光强度
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);//光源中
的散射光强度
    glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);//材质
属性中的镜面反射光


    // 开启光源
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);
```

```
    makeStripeImage();//纹理映射函数
    glPixelStorei(GL_UNPACK_ALIGNMENT, 1);

    glGenTextures(1, &texName);
    glBindTexture(GL_TEXTURE_1D, texName);

    glTexParameteri(GL_TEXTURE_1D,          GL_TEXTURE_WRAP_S,
GL_REPEAT);
    glTexParameteri(GL_TEXTURE_1D,     GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
    glTexParameteri(GL_TEXTURE_1D,     GL_TEXTURE_MIN_FILTER,
GL_LINEAR);

    glTexImage1D(GL_TEXTURE_1D, 0, GL_RGBA, stripeImageWidth,
0,
                 GL_RGBA, GL_UNSIGNED_BYTE, stripeImage);

    /*设置纹理和物体表面颜色处理方式*/
    glTexEnvf(GL_TEXTURE_ENV,              GL_TEXTURE_ENV_MODE,
GL_MODULATE);//和物体表面颜色做与运算

    /*设置自动生成纹理坐标的计算方式*/
```

```
    currentCoeff = xequalzero;
    currentGenMode = GL_OBJECT_LINEAR;
    currentPlane = GL_OBJECT_PLANE;

    //一维纹理，只有 S 坐标
    glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, currentGenMode);//整形形式
    glTexGenfv(GL_S, currentPlane, currentCoeff);//向量形式
    glEnable(GL_TEXTURE_GEN_S);//激活 s 坐标的纹理坐标生成
    glEnable(GL_TEXTURE_1D);
    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);
}

//绘制图形函数
void display(void)
```

```
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPushMatrix ();
    glRotatef(-70.0, 0.0, 0.0, 1.0);//设置茶壶的角度
    glBindTexture(GL_TEXTURE_1D, texName);
    glutSolidTeapot(2.0);//绘制茶壶
    glPopMatrix ();

    glBlendFunc( GL_SRC_ALPHA, GL_ONE );

    glFlush();
}
//修正窗内的图形显示函数
void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h)
        glOrtho (-3.5, 3.5, -3.5*(GLfloat)h/(GLfloat)w,
                 3.5*(GLfloat)h/(GLfloat)w, -3.5, 3.5);
    else
```

```
        glOrtho (-3.5*(GLfloat)w/(GLfloat)h,
                 3.5*(GLfloat)w/(GLfloat)h, -3.5, 3.5, -3.5,
3.5);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
//主函数，实现功能函数的调用与图形的显示
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode    (GLUT_SINGLE    |    GLUT_RGB    |
GLUT_DEPTH);
    glutInitWindowSize(400, 400);//设置窗口大小为 400x400
    glutInitWindowPosition(100, 100);//设置窗口的位置
    glutCreateWindow (argv[0]);//创建窗口
    init ();//初始化操作
    glutDisplayFunc(display);//绘制图形
    glutReshapeFunc(reshape);//修正窗内的图形显示
    glutMainLoop();//事件处理循环
    return 0;
}
```