# 《计算机图形学实验》综合实验报告

题目 OpenGl 绘制体现纹理和光照的茶壶

学 号 20201060294

姓 名 赵 晶

指导教师 钱 文 华

日 期 2022 年 6 月 20 日

# OpenGl 绘制体现纹理和光照的茶壶

**摘要：**随着科学技术的不断发展，计算机图形学也得到了极大地发展。真实的三维游戏场景、3D 电影、虚拟现实等无不体现了计算机图形学的贡献。计算机图形学(Computer Graphics，简称 CG)是一种使用数学算法将二维或三维图形转化为计算机显示器的栅格形式的科学。作为三维场景中必不可少的部分，纹理可以使物体表面呈现凹凸不平的沟纹、也可使物体的光滑表面显示彩色图案，而光照则可以使绘制的图象更加真实。茶壶作为生活中常见的物体，在 OpenGl 中也可直接调用相关函数显示一个逼真的茶壶，本文正是通过 OpenGl 中典型的三维图形——茶壶来展示光照和纹理的实现。

关键词：茶壶 纹理 光照 基本变换

# OpenGl draws the teapot with texture and lighting

Abstract: With the continuous development of science and technology, computer graphics has also been greatly developed. Real 3D game scenes, 3D movies, virtual reality and so on all embody the contribution of computer graphics. Computer Graphics (CG) is the science of using mathematical algorithms to convert two-dimensional or three-dimensional Graphics into the grid form of a Computer display. As an essential part of a THREE-DIMENSIONAL scene, texture can make the surface of an object show uneven grooves, or make the smooth surface of the object show colorful patterns, and lighting can make the drawn image more realistic. Teapot as a common object in life, in OpenGl can also directly call the relevant functions to display a lifelike teapot, this paper is through OpenGl's typical THREE-DIMENSIONAL graphics - teapot to show the realization of lighting and texture.

Key words: teapot texture light basic transformation

# 目录

# 一. 实验简介

## 1.1 实验背景

计算机图形学的主要研究内容是研究如何在计算机中表示图形、以及利用计算机进行图形的计算、处理和显示的相关原理与算法。自从 2010 年后，计算机图形学的工作主要集中在集成更复杂的多阶段的图像生成。纹理映射也已经发展为一个复杂的多阶段过程，使用着色器（shader）将纹理渲染、反射技术等多种算法集成到一个渲染引擎中的操作并不少见[1]。游戏中的建模就是计算机图形学的一大应用，对真实场景的建模自然离不开对图像的基本操作即基本变换、纹理、光照渲染等。本文就是对部分基本操作进行实现以加深对计算机图形学的理解。

## 1.2 实验内容

利用 OpenGl 工具，实现三维图形的渲染，渲染过程包括纹理、色彩、光照等。本文主要通过调用 OpenGl 中自带的 glutSolidTeapot（）函数直接绘制茶壶，然后运用计算机图形学知识对其实现平移、旋转、实心图转化为线框图、光照渲染和纹理转换，以加深对计算机图形学知识的理解。

# 二. 程序简介

2.1 **开发工具**:codeblocks，OpenGl

## 2.2 程序设计

**2.2.1 分析问题**:本文的任务主要是对 OpenGl 可自行显示的茶壶进行一系列基本操作，共实现 8 个模块：①对茶壶进行上下左右的平移变换；②实现或停止场景的自转；③将实心的茶壶变成线框图；④环境光的移动以及颜色变换；⑤聚

光灯的移动及角度变换；⑥纹理转换；⑦视线转变（展示 3 个茶壶嵌套，转换视线方便看到内层茶壶）；⑧退出。

**2.2.2 设计算法**:在分析问题的过程中发现需要实现的功能较多，因此决定采用键盘交互的形式实现相应的功能，即通过用户按不同的键来实现不同的功能，使得运行结果更加清晰条理。

**2.2.3 编写程序**:本文使用 codeblocks 来编写程序，实现相应 8 个模块的功能. 运用键盘交互，用户按下相应的键，程序实现对应的功能并展示在窗口。详细实现步骤见 3.2。

**2.2.4 运行程序**:根据程序的运行结果找到存在的问题并寻找解决方法解决问题，对程序不断进行优化与改进。

**2.2.5 编写程序报告**:通过实验报告来详细介绍程序的使用方法和实现的功能。并介绍在实现的过程中遇到的问题及最终的解决方法和待改进的部分。

**2.3 实现目的**:利用 opengl 的函数功能，使用 opengl 库函数对 SD 图像的坐标位置进行定位，使用相关函数进行旋转，平移、光照和纹理渲染。设置输入输出语句，配合键盘人机交互，由用户对将要进行的操作进行指定。加深对计算机图形学基本知识尤其是三维渲染的理解与应用。

**2.4 基本模块介绍**:①按键回调函数 key：用户通过敲击键盘实现相应功能，本程序从键盘上的“q“开始从左到右从上到下依次实现程序设计中分析问题所决定的待实现模块；②空闲回调函数 idle：程序没有事件要执行的时候便会执行该函数，该函数主要用于重新绘制当前窗口以使显示的图像稳定；③显示图像的函数 display：该函数主要用于绘制想要显示的图像，此处是 3 个相互嵌套的茶壶。分别定义 3 个茶壶，改变调用的茶壶函数的参数（改变各个茶壶的大小以实现相互嵌套），采用堆栈来实现他们的显示；④设置茶壶纹理颜色的函数 makeStripeImage：通过设置数组 stripeImage 的数值来改变纹理眼样式和颜色；⑤绘制回调函数 redraw：调用 glEnable 函数开启深度测试和光照模式、定义环境光和聚光灯的位置、设置光照的各个参数（光成分、光源方向、光照颜色）。

# 三．理论及实现

## 3.1 关键算法的理论介绍

### 3.1.1 光照

（1） OpenGL可以设置至少8种光源，在这里我们使用了两种光源，一种是环境光，另一种是聚光灯。在设置光照时，我们需要考虑这样三种光：环境反射光、镜面反射光、漫反射光。在Phong光照模型中，就是通过这三种分量的取值来模拟真实光照的。其中，环境反射光是光源多次反射后的光，可以理解为背景光，镜面反射和漫反射反映了物体表面的粗糙/光滑程度，两者的和是一定的。

（2） 在OpenGL中，如果想要使用光源，需要输入glEnable( GL_LIGHTING );来开启光照模式。对于一个光源，我们首先需要确定它的位置，这个光源可以是点光源，也可以是平行光源，本文使用点光源。

（3） 可以使用函数： *glLightfv*(光源编号,光源特性, 参数数据);同时设置漫反射、镜面反射、环境反射的颜色参数：

gl*Lightfv*(*GL _ LIGHT*0,*GL _ SPECULAR*,*color*);/ / 设置镜面反射参数

gl*Lightfv*(*GL _ LIGHT*0,*GL _ DIFFUSE*,*color*);/ / 设置漫射光成分

gl*Lightfv*(*GL _ LIGHT*0,*GL _ AMBIENT*,*color*);/ / 设置第0号光源多次反射后的光照颜色（环境光颜色）

在本次实验中，我们把环境光设为了白色和红色，而镜面反射光和漫反射光都设置为白色[2]。

### 3.1.2 纹理

（1） 纹理坐标自动生成：有两种形式，一种整形形式(glTexGeni)，一种向量形式(glTexGenfv)，使用它比手动设置纹理坐标方便(glTexCoord)[1]。

（2） 用于指定一维、二维和三维纹理的函数分别为：

$$Void\ glTexImage1D\begin{pmatrix} GLenum\ target,\ Glint\ level,\ Glint\ components,\ GLsizei\ width, \\ Glint\ border,\ GLenum\ format,\ GLenum\ type,\ const\ GLvoid\ *texels \end{pmatrix};$$

$$Void\ glTexImage2D\begin{pmatrix} GLenum\ target,\ Glint\ level,\ Glint\ components,\ GLsizei\ width, \\ GLsizei\ height,\ GLint\ border,\ GLenum\ format,\ GLenum\ type, \\ const\ GLvoid\ *texels \end{pmatrix};$$

$$Void\ glTexImage3D\begin{pmatrix} GLenum\ target,\ Glint\ level,\ Glint\ components,\ GLsizei\ width, \\ GLsizei\ height,\ GLsizei\ depth,\ Glint\ border,\ GLenum\ format, \\ GLenum\ type,\ const\ GLvoid\ *texels \end{pmatrix};$$

其中，参数target取值一般为GL_TEXTURE_1D, GL_TEXTURE_2D和GL_TEXTURE_3D，分别与一维、二维和三维的纹理相对应。参数Level表示纹理多分辨率层数，通常取值为0，表示只有一种分辨率。参数components的可能取值为1～4的整数以及多种符号常量(如GL_RGBA)，表示纹理元素中存储的哪些分量（RGBA颜色、深度等）在纹理映射中被使用，1表示使用R颜色分量，2表示使用R和A颜色分量，3表示使用RGB颜色分量，4表示使用RGBA颜色分量。参数width，height，depth分别指定纹理的宽度、高度、深度。参数format和type表示给出的图像数据的数据格式和数据类型，这两个参数的取值都是符号常量（比如format指定为GL_RGBA,type指定为GL_UNSIGNED_BYTE，参数texels指向内存中指定的纹理图像数据[2]。

（3） 在定义了纹理之后，需要启用纹理的函数：

$$glEnable(GL\_TEXTURE\_1D);$$
$$glEnable(GL\_TEXTURE\_2D);$$
$$glEnable(GL\_TEXTURE\_3D);$$

（4） 在启用纹理之后，需要建立物体表面上点与纹理空间的对应关系，即在绘制基本图元时，在glVertex函数调用之前调用glTexCoord函数，明确指定当前顶点所对应的纹理坐标，例如：

```
glBegin（GL_TRIANGLES）;
glTexCoord2f (0.0,0.0); glVertex2f (0.0,0.0);
glTexCoord2f (1.0,1.0); glVertex2f (15.0,15.0);
glTexCoord2f (1.0,0.0); glVertex2f (30.0,0.0);
glEnd();
```

（5） 在OpenGL中，纹理坐标的范围被指定在[0,1]之间，而在使用映射函数进行纹理坐标计算时，有可能得到不在[0,1]之间的坐标。此时OpenGL有两种处理方式，一种是截断，另一种是重复，它们被称为环绕模式。在截断模式（GL_CLAMP）中，将大于1.0的纹理坐标设置为1.0，将小于0.0的纹理坐标设置为0.0。在重复模式（GL_REPEAT）中，如果纹理坐标不在[0,1]之间，则将纹理坐标值的整数部分舍弃，只使用小数部分，这样使纹理图像在物体表面重复出现。例如，使用下面的函数：

$$glTexParameterf\ (GL\_TEXTURE\_2D,\ GL\_TEXTURE\_WRAP\_S,\ GL\_CLAMP);$$
$$glTexParameterf\ (GL\_TEXTURE\_2D,\ GL\_TEXTURE\_WRAP\_S,\ GL\_REPEAT);$$

分别指定二维纹理中 s 坐标采用截断或重复处理方式[3]。

### 3.1.3 投影

OpenGl 中有两种投影：正射投影和透视投影。透视投影通过指定一个平截头体来定义视见体的范围，平截头体如图 3.1.1 所示：

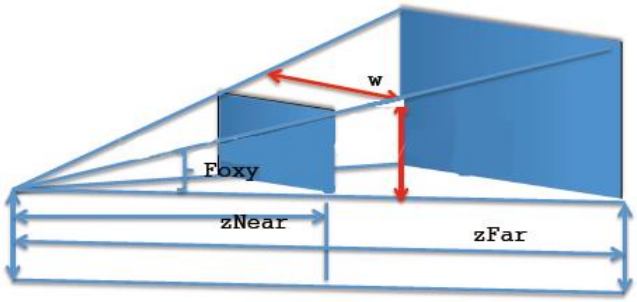


**图 3.1.1 平截头体**

实现代码如下：

```
void gluPerspective (GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar)
```

参数：fovy：眼睛上下睁开的幅度，角度值，值越小，视野范围越狭小（眯眼），值越大，视野范围越宽阔；zNear：近裁剪面到眼睛的距离；zFar：远裁剪面到眼睛的距离，注意 zNear 和 zFar 不能设置设置为负值（看不到眼睛后面的东西）；aspect：裁剪面的宽 w 高 h 比，这个影响到视野的截面有多大[4]。

## 3.2 程序实现步骤

主函数的实现流程图见图 3.2.1，显示茶壶的函数 display 的流程图见图 3.2.1。

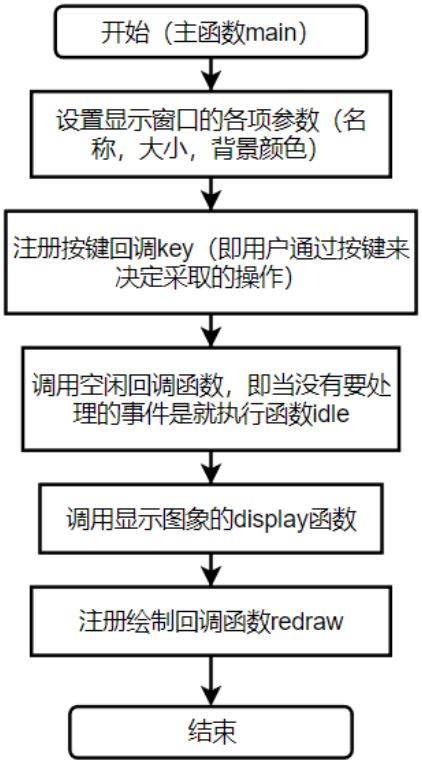


图 3.2.1 主函数流程图

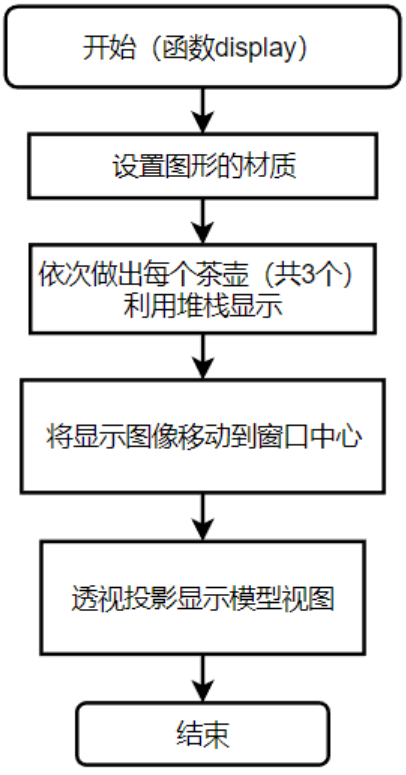


图 3.2.2 显示函数的流程图

# 四．结果展示和分析

## 4.1 结果图展示

点击运行，显示窗口显示图 4.1.1 所示图片。


**图 4.1.1 初始图像**

将输入方式改为英文输入，大小写都可以。点击“h”或者“H”改变环境光的颜色如图 4.1.2 所示。点击“y”或者“Y”场景可以开始或者停止旋转，图片难以展示效果，具体见运行结果；点击“o”或者“O”可先将图片进行实心和线性之间的转换，线性框图如图 4.1.3 所示。

图 4.1.2 改变环境光后的图像

图 4.1.3 线性框图展示

按键“u”“i”（不区分大小写，下同）纹理分别如图 4.1.4 和图 4.1.5 所示。

图 4.1.4 纹理转换 u

图 4.1.5 纹理转换 i

依次按键“p”“a”“s”“d”“f”“g”实现环境光的上下左右前后的移动，效果图给分别如图 4.1.6-图 4.1.11 所示。

图 4.1.6 环境光上移

图 4.1.7 环境光下移

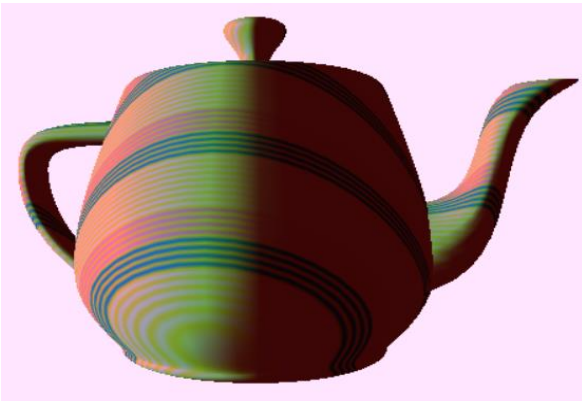

图 4.1.8 环境光左移

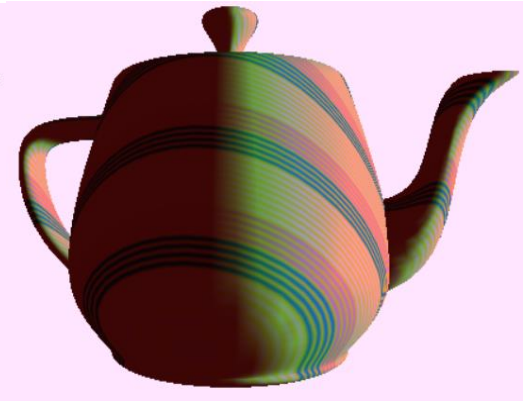

图 4.1.9 环境光右移

图 4.1.10 环境光前移

图 4.1.11 环境光后移

依次按键“j”、“k”、“l”、“z”、“x”、“c”实现聚光灯的上下左右前后移动，效果图见图 4.1.12-4.1.17 所示（为了方便观察，环境光均在茶壶正后方），可以观察茶壶的上下左右端的光线变换。

图 4.1.12 聚光灯上移

图 4.1.13 聚光灯下移

图 4.1.14 聚光灯左移

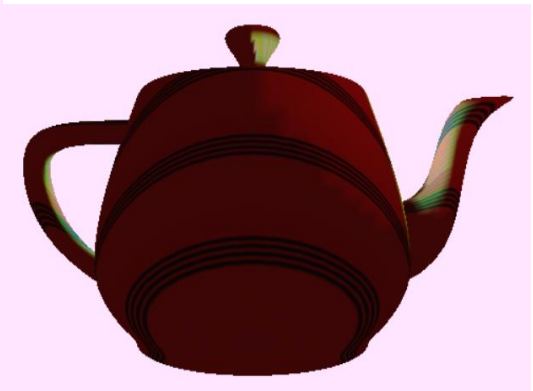

图 4.1.15 聚光灯右移

图 4.1.16 聚光灯前移

图 4.1.17 聚光灯后移

在图 4.1.17 的基础上按键“v”、“b”分别使聚光灯角度变大，变小，分别如图 4.1.18 和 4.1.19 所示。

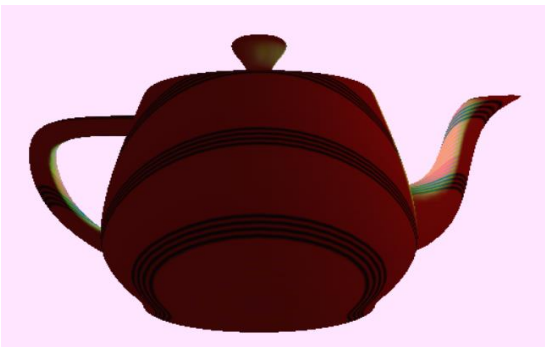

图 4.1.18 聚光灯角度变大

图 4.1.19 聚光灯角度变小

分别按键“1”、“2”……会改变视野，看到最外面茶壶内部的现象。视野由内到外变化看到的图像如图 4.1.20-4.1.25 所示，其中为了使部分图像更加清晰会附加线框图来展现。

图 4.1.20 按“1”后的图像

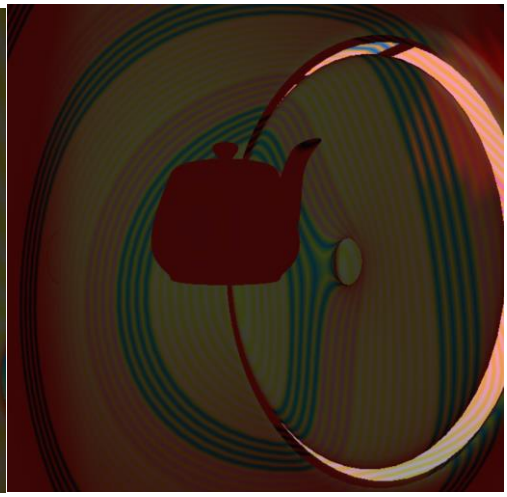

图 4.1.21 按“2”后的图像

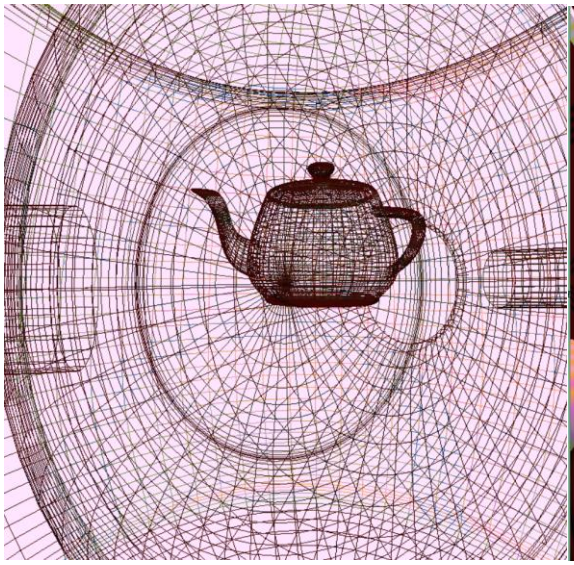

图 4.1.22 按“2”后的线框图

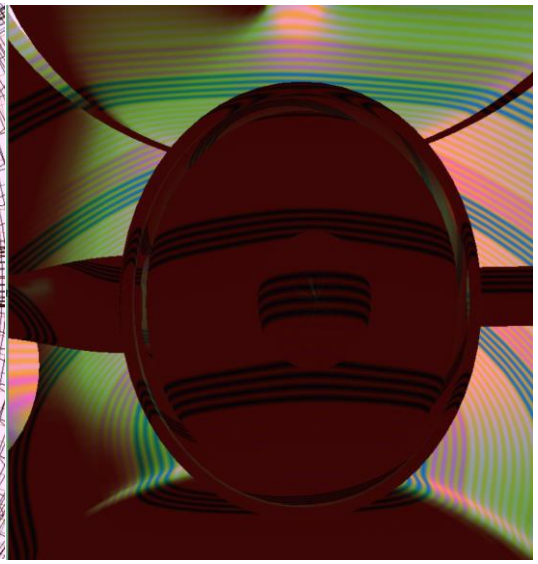

图 4.1.23 按“3”后的图像

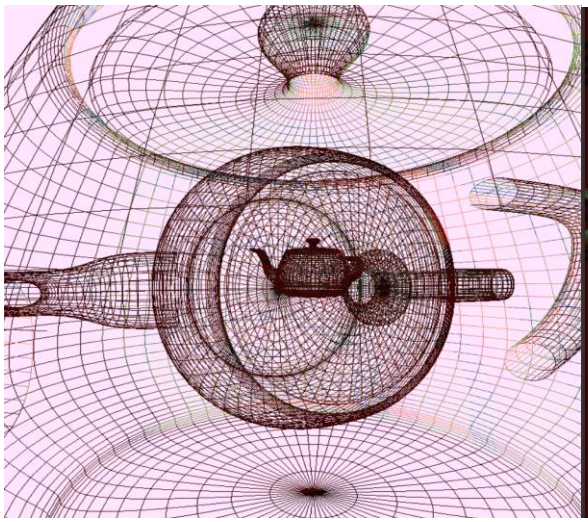

图 4.1.24 按“3”后的线框图

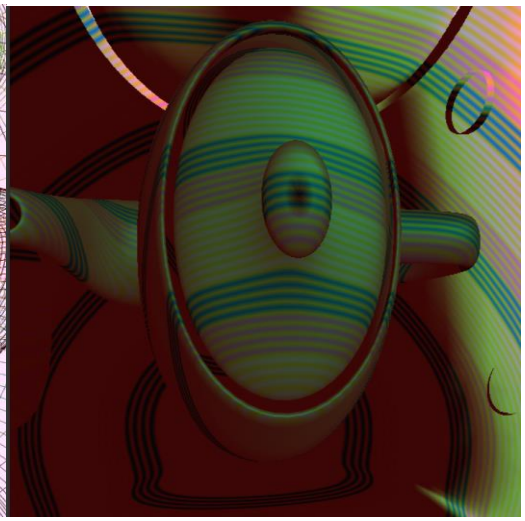

图 4.1.25 按“4”后的图像

在显示图像的过程中，可以通过按键“w”、“e”、“r”、“t”、“o”来对图像进行上下左右的平移变换和实心线框图的转换来辅助观察图像。

### 4.2 分析结果

由 4.1 的结果图可知，本程序实现了光照和纹理的渲染，并且可以通过平移旋转以及线框图的变换较好的观察到光照和纹理渲染实现了本实验的目的，加强了对计算机图形学知识的认识和理解。

### 4.3 存在的问题

通过实验结果可以发现本程序在纹理的实现方面还有明显的欠缺，即本程序中的纹理是由计算机利用颜色生成的，并不是贴上了需要的纹理图片，普适性较低，还有较大的提升空间。

## 五．实验体会及小结

### 5.1 运用到的图形学知识

#### 5.1.1 常见回调函数

(1) void glutDisplayFunc(void (*func)(void));
注册当前窗口的显示回调函数
参数:
func:形为 void func()的函数,完成具体的绘制操作
这个函数告诉 GLUT 当窗口内容必须被绘制时,那个函数将被调用.当窗口改变大小或者从被覆盖的状态中恢复,或者由于调用 glutPostRedisplay()函数要求 GLUT 更新时,执行 func 参数指定的函数.

(2) void glutReshapeFunc(void (*func)(int width, int height));
指定当窗口的大小改变时调用的函数

参数:

func:形如 void func(int width, int height)

处理窗口大小改变的函数.

width,height:为窗口改变后长宽.

这个函数确定一个回调函数,每当窗口的大小或形状改变时(包括窗口第一次创建),GLUT 将会调用这个函数,这个回调函数接受这个窗口新的长宽作为输入参数.

(3) void glutIdleFunc(void (*func)(void));

设置空闲回调函数

参数:

func:形如 void func(void);

当系统空闲时调用.

### 5.1.2 用于键盘操作的回调函数

(1) void glutKeyboardFunc(void (*func)(unsigned char key, int x, int y));注册当前窗口的键盘回调函数

参数:

func:形如 void func(unsigned char key, int x, int y)

key:按键的 ASCII 码

x,y:当按下键时鼠标的坐标,相对于窗口左上角,以像素为单位

当敲击键盘按键时调用. ( 除了特殊按键,即 glutSpecialFunc()中处理的按键,详见 glutSpecialFunc() )

(2)void glutSpecialFunc(void (*func)(int key, int x, int y));

设置当前窗口的键盘特定键的回调函数

参数:

Func:形如 void func(int key, int x, int y);

key:按下的特定键,为以下定义的常量

GLUT_KEY_F1:F1 功能键;GLUT_KEY_F2:F2 功能键； GLUT_KEY_F3:F3 功能键；

GLUT_KEY_F4:F4 功能键;GLUT_KEY_F5:F5 功能键； GLUT_KEY_F6:F6 功能键；

GLUT_KEY_F7:F7 功能键;GLUT_KEY_F8:F8 功能键； GLUT_KEY_F9:F9 功能键；

GLUT_KEY_F10:F10 功能键；GLUT_KEY_F11:F11 功能键；

GLUT_KEY_F12:F12 功能键;GLUT_KEY_LEFT:左方向键;GLUT_KEY_UP:上方向键；

GLUT_KEY_RIGHT:右方向键；GLUT_KEY_DOWN:下方向键；

GLUT_KEY_PAGE_UP:PageUp 键；GLUT_KEY_PAGE_DOWN:PageDown 键；

GLUT_KEY_HOME:Home 键； GLUT_KEY_END:End 键;GLUT_KEY_INSERT:Insert 键

注意特别：上述键盘和鼠标的回调函数中，都返回光标的当前坐标 x 和 y，注意这个坐标的原点是在窗口左上角，向右为 X 轴的正方向，向下为 Y 轴的正方向。而 glut 中的函数是把窗口的左下角设为原点，向右为 X 轴的正方向，向上为 Y 轴的正方向。所以假设窗口的高为 winHeight，那么回调函数中光标（x，y）的位置换算到 glut 中实际上是（x，winHeight - y)。

### 5.1.3 基本变换

世界坐标系（WC)：正对绘图窗口，右方是世界坐标系的 x 轴，上方是 y 轴，从屏幕指向我们的是 z 轴。

模型坐标系（MC)：使用 glVertex()之类的函数定义图元时用的坐标系。

初始时，MC 和 WC 是重合的。

旋转变换：旋转需要选定一个向量以及一个角度，该向量指模型坐标系下的一个向量，模型坐标系围绕该向量旋转，相应的函数为 glRotate（)。

平移变换：平移时 MC 和 WC 的位置不变，但是 MC 里面所有的坐标都将在原来的基础上加上偏移量（dx，dy，dz)，相应的函数为 glTranslatef（）[5]。

### 5.1.4 光照渲染

设置光照时需要考虑 3 种光：环境反射光，镜面反射光，漫反射光。其中环境反射光是光源多次反射后的光，可以理解为背景光，镜面反射和漫反射反映了物体表面的粗糙/光滑程度，两者的和是一定的。

### 5.1.5 缓冲区

```
glutInitDisplayMode( GLUT_RGB | GLUT_DEPTH | GLUT_DOUBLE);//初始化显示模式:RGB 颜色模型，深度测试，使用双缓冲窗口
```

单缓冲：将所有的绘图指令在窗口上执行，就是直接在窗口上绘图，这样的绘图效率是比较慢的，如果使用单缓冲，而电脑比较慢，屏幕会发生闪烁。一般只用于显示单独的一副非动态的图像。

双缓冲： 实际上的绘图指令是在一个缓冲区完成，这里的绘图非常的快，在绘图指令完成之后，再通过交换指令把完成的图形立即显示在屏幕上，这就避免了出现绘图的不完整，同时效率很高。一般用于生成动画效果。

## 5.2 学习到的新知识

**纹理贴图**：在计算机图形学中，纹理贴图是使用图像、函数或其他数据源来改变物体表面外观的技术。

纹理（Texturing）是一种针对物体表面属性进行“建模”的高效技术。图像纹理中的像素通常被称为纹素（Texels），区别于屏幕上的像素。通过将投影方程（projector function）运用于空间中的点 ，从而得到一组称为参数空间值（parameter-spacevalues）的关于纹理的数值，再使用一个或者多个映射函数（corresponder function）将参数空间值（parameter-space values ）转换到纹理空间。这个过程就称为贴图（Mapping，也称映射 ）,也就是纹理贴图（Texture Mapping，也称纹理映射）[6]。

## 5.3 遇到的问题及解决办法

**问题 1**：当改变窗口的大小，图形会离开显示窗口；

解决办法：在 reshape 函数中只加入一个语句即 glViewport(0, 0, width, height); /*glviewport 前两个参数：以像素为单位，指定了窗口左下角的位置，后两个参数：表示视口矩形的宽度和高度，根据窗口的实时变化重回窗口。

**问题 2**：实现光照和纹理渲染，如何在图上同时展示；

解决办法：通过键盘回调函数，实现人机交互，通过用户的指令来实现相应的功能，使得程序实现更加条理清晰。

**问题 3**：本程序展现 3 个茶壶，只能通过线框图确定有 3 个茶壶但是无法近距离查看里面的茶壶；

解决办法：修改视线数组 eye 的第三个参数，就可以看到内层茶壶了，本程序可以通过按 1，2，……等键来选择。

**问题 4**：纹理贴图不光可以使用颜色自动生成，也可以将指定图像贴到物体表面，本文只实现了前者暂未实现后者。在尝试后者的时候遇到了很多问题，首先，程序找不到要贴的图像，后发现将图像放在和.cpp 文件相同的目录即可；然后，程序找到.bmp 文件后，又提示该图像不是位图文件，无法进行下一步操作。最后，尝试其余文件名后缀的图像也失败了。

## 5.4 心得体会

经过本次实验，我学会了编程实现三维图像的基本操作以及光照、纹理渲染，对计算机图形学有了一个基本的了解。

我也深刻意识到了计算机图形学的强大功能，当今热门领域 3D 电影，游戏，虚拟技术无不体现着计算机图形学的知识。显然这些先进功能的实现离不开最基本的光照渲染和纹理贴图。在真实感三维场景绘制中，为了使图像更加逼真，自然需要加入光照进行渲染。构造好图像以后，为了让图像更加形象，

可以加入纹理贴图对其进行渲染。通过以上的实验让我初步了解了当今社会流行的各种 3D 产品的实现，意识到他们并不神秘，处处体现了科学家们的智慧。

## 六. 参考文献


[1]百度. 计算机图形学[EB/OL]. [2022 年 6 月 20 日]. https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA%E5%9B%BE%E5%BD%A2%E5%AD%A6/279486.

[2] ZJU_fish1996. [OpenGL]茶壶与光照[EB/OL]. [2022 年 6 月 20 日]. https://blog.csdn.net/zju_fish1996/article/details/51365518.

[3] 凯鲁嘎吉. OpenGL 实例：纹理映射[EB/OL]. [2022 年 6 月 20 日]. https://www.cnblogs.com/kailugaji/archive/2019/05/06/10821630.html.

[4] 绯红 Tanker. gluPerspective 的具体含义[EB/OL]. [2022 年 6 月 20 日]. https://blog.csdn.net/tyxkzzf/article/details/40921713.

[5] sha256sum. OpenGL-旋转平移与缩放[EB/OL]. [2022 年 6 月 20 日]. https://blog.csdn.net/charles_neil/article/details/78664228.

[6] 鹅厂程序小哥. 3D 图形学（4）：纹理贴图[EB/OL]. [2022 年 6 月 20 日]. https://blog.csdn.net/qq826364410/article/details/88569131.


## 七. 附录

附录 1：源代码 main.cpp