浙江大学实验报告

课程名称：____计算机图形学____ 指导老师：_____成绩：_____
实验名称：____OpenGL 消隐和光照_____ 实验类型：_____基础实验_____同组学生姓名：_____

装

订

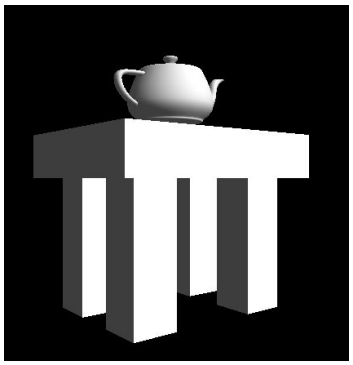
线

一、实验目的和要求

在 OpenGL 观察实验的基础上，通过实现实验内容，掌握 OpenGL 中消隐和光照的设置，并验证课程中消隐和光照的内容。

二、实验内容和原理

使用 Visual Studio C++编译已有项目工程。



模型尺寸参见 OpenGL 观察实验。要求修改代码达到以下要求：

1. 通过设置材料使得桌面和四条腿的颜色各不相同，分别为：(1, 0, 0), (0, 1, 0), (1, 1, 0), (0, 1, 1), (0, 0, 1)；
2. 通过设置材料使得茶壶为金黄色；
3. 添加按键处理，移动场景中的光源，并能改变光源的颜色；
4. 修改茶壶的镜面反射系数，使之对光源呈现高光；
5. 在场景中添加一个聚光光源，其照射区域正好覆盖茶壶，并能调整改聚光光源的照射角度和朝向。

三、主要仪器设备

Visual Studio C++
glut-3.7.6-bin.zip
模板工程

四、操作方法和实验步骤

1.通过设置材料使得桌面和四条腿的颜色各不相同

指定桌面与桌腿的材质属性：

```
GLfloat mat_desktop[] = {1.0, 0.0, 0.0, 0.0};     // 桌面材质
GLfloat mat_leg1[] = {0.0, 1.0, 0.0, 0.0};     // 桌腿1材质
GLfloat mat_leg2[] = {1.0, 1.0, 0.0, 0.0};     // 桌腿2材质
```

```
GLfloat mat_leg3[] = {0.0, 1.0, 1.0, 0.0};    // 桌腿3材质
GLfloat mat_leg4[] = {0.0, 0.0, 1.0, 0.0};    // 桌腿 4 材质
```

在绘图过程中设置物体材质：

```
    // 桌面材质
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_desktop);
    glutSolidCube(1.0);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(1.5, 1, 1.5);
    // 桌腿1材质
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_leg1);
    Draw_Leg();
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-1.5, 1, 1.5);
    // 桌腿2材质
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_leg2);
    Draw_Leg();
    glPopMatrix();

    glPushMatrix();
    glTranslatef(1.5, -1, 1.5);
    // 桌腿3材质
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_leg3);
    Draw_Leg();
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-1.5, -1, 1.5);
    // 桌腿4材质
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_leg4);
    Draw_Leg();
    glPopMatrix();
```

2.通过设置材料使得茶壶为金黄色

定义茶壶材质：

```
GLfloat mat_Teapot[] = {0.6, 0.5, 0.0, 0.0};  // 茶壶材质
```

设置茶壶材质：

```
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_Teapot);
```

3.添加按键处理，移动场景中的光源，并能改变光源的颜色

添加按键响应，通过 j,k,l,i 键移动光源（左、右、前、后）：

```
        // 改变光源位置
    case 'i':
        light_pos[2] += 0.5;
        break;
    case 'k':
        light_pos[2] -= 0.5;
        break;
    case 'j':
        light_pos[0] -= 0.5;
        break;
    case 'l':
        light_pos[0] += 0.5;
        break;
```

每次改变后重设光源位置：

```
glLightfv(GL_LIGHT0, GL_POSITION, light_pos);
```

添加按键响应，通过 r、g、b 键改变管光源颜色：

```
        //  改变光源颜色
    case 'r':
        white[0] -= 0.02;
        if (white[0] <= 0.0) white[0] = 1.0;
        break;
    case 'g':
        white[1] -= 0.02;
        if (white[1] <= 0.0) white[1] = 1.0;
        break;
    case 'b':
        white[2] -= 0.02;
        if (white[2] <= 0.0) white[2] = 1.0;
        break;
```

每次改变后重设光源颜色：

```
glLightfv(GL_LIGHT0, GL_AMBIENT, white);
```

4.修改茶壶的镜面反射系数，使之对光源呈现高光

添加按键响应，通过 h 键使茶壶对光源呈现高光：

```
bool bSpec = false;    // 是否启用高光
......
GLfloat mat_spectular[] = {1.0, 1.0, 1.0, 1.0};  // 高光材质
......
        // 高光效果
```

```
    case 'h':
        bSpec = !bSpec;
        break;
......
    if (bSpec)
    {
        glMaterialfv(GL_FRONT, GL_SPECULAR, mat_spectular);  // 设置高光效果
        glMaterialf(GL_FRONT, GL_SHININESS, 65.0);   // 设置镜面指数
    }
```

5.在场景中添加一个聚光光源，其照射区域正好覆盖茶壶，并能调整改聚光光源的照射角度和朝向。

定义聚光光源的属性，使之在茶壶的正上方，照射方向为 y 轴负方向，且照射角度刚好覆盖茶壶，并通过按键't'响应是否开启聚光光源，按键'm'、'n'改变聚光光源的方向，按键'x'改变聚光光源的照射角度。

```
// 聚光光源属性
float direction[] = {0, -1, 0};     // 照射方向
float spotCutOff = 4.0;     // 裁剪角度

// 初始化聚光光源
void initSpot()
{
    // 所有光源的属性
    float noAmbient[] = {1.0, 1.0, 1.0, 1.0};  // 环境光
    float diffuse[] = {0.0, 0.0, 1.0, 1.0};     // 漫射光
    float position[] = {0.0, 4.0, 0.0, 1.0};    // 位置

    // 设置光源属性
    glLightfv(GL_LIGHT1, GL_AMBIENT, noAmbient);
    glLightfv(GL_LIGHT1, GL_DIFFUSE, diffuse);
    glLightfv(GL_LIGHT1, GL_POSITION, position);

    // 聚光光源属性
    // 更新聚光光源方向和夹角
    updateSpotProperties();

    // 定义光线汇聚程度
    glLightf(GL_LIGHT1, GL_SPOT_EXPONENT, 15.0);

    // 定义衰减（这里使用了缺省值，不随距离衰减）
    glLightf(GL_LIGHT1, GL_CONSTANT_ATTENUATION, 1.0);
    glLightf(GL_LIGHT1, GL_LINEAR_ATTENUATION, 0.0);
    glLightf(GL_LIGHT1, GL_QUADRATIC_ATTENUATION, 0.0);
}
```

```
void updateSpotProperties()
{
    // 聚光光源方向
    glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION, direction);
    // 聚光光源夹角
    glLightf(GL_LIGHT1, GL_SPOT_CUTOFF, spotCutOff);
}
```

补充说明：

在 OpenGL 中须开启光源才能显示光照效果，具体为在绘制函数 redraw 中添加如下代码：

```
void redraw()
{
    ......
    glEnable(GL_LIGHTING);  // 开启光照
    glEnable(GL_NORMALIZE); // 启用法向归一化
    glEnable(GL_DEPTH_TEST); // 开启深度测试

    if (bLight0)  // 0号光源
    {
        // 光源0属性
        glLightfv(GL_LIGHT0, GL_POSITION, light_pos);
        glLightfv(GL_LIGHT0, GL_AMBIENT, white);
        glEnable(GL_LIGHT0);  // 开启光源0
    }
    else
    {
        glDisable(GL_LIGHT0);
    }

    // 聚光光源
    if (bSpot)
    {
        initSpot();
        glEnable(GL_LIGHT1);
    }
    else
    {
        glDisable(GL_LIGHT1);
    }
    ......
}
```

由于本次实验绘制采用的是 glut 提供的函数，无法对每个面指定法向，我采用了法向归一化

的方法为场景中所有物体计算法向：glEnable(GL_NORMALIZE);

五、实验数据记录和处理

物体材质属性：

```
GLfloat mat_desktop[] = {1.0, 0.0, 0.0, 0.0};     // 桌面材质
GLfloat mat_leg1[] = {0.0, 1.0, 0.0, 0.0};     // 桌腿1材质
GLfloat mat_leg2[] = {1.0, 1.0, 0.0, 0.0};     // 桌腿2材质
GLfloat mat_leg3[] = {0.0, 1.0, 1.0, 0.0};     // 桌腿3材质
GLfloat mat_leg4[] = {0.0, 0.0, 1.0, 0.0};     // 桌腿 4 材质
GLfloat mat_Teapot[] = {0.6, 0.5, 0.0, 0.0};  // 茶壶材质
```

光源属性：

点光源：

```
GLfloat white[] = { 1.0, 1.0, 1.0, 1.0 }; // 环境光属性（白光）
float diffuse[] = {0.0, 0.0, 1.0, 1.0};     // 漫射光
```

聚光光源：

```
float noAmbient[] = {1.0, 1.0, 1.0, 1.0};  // 环境光
float diffuse[] = {0.0, 0.0, 1.0, 1.0};     // 漫射光
```

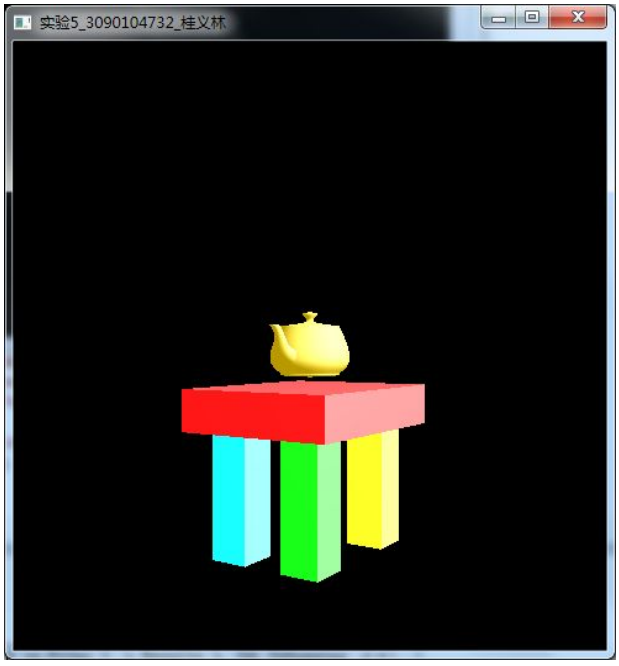
高光：

高光材质：

```
GLfloat mat_spectular[] = {1.0, 1.0, 1.0, 1.0};  // 高光材质
镜面反射系数：65.0
```

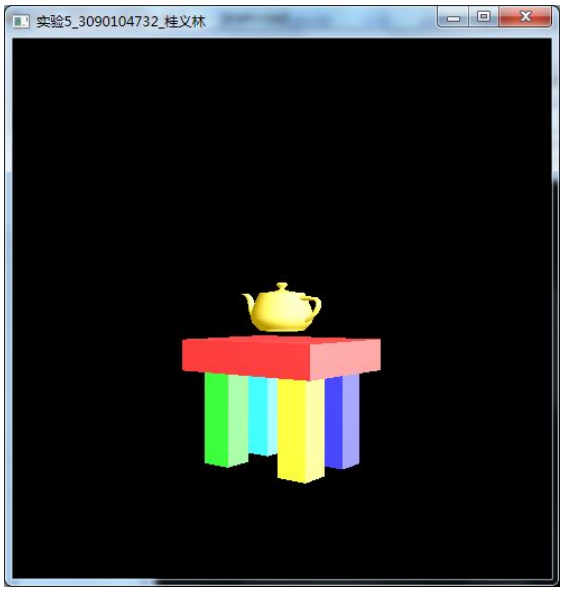
六、实验结果与分析

1.设置桌子与茶壶的材质

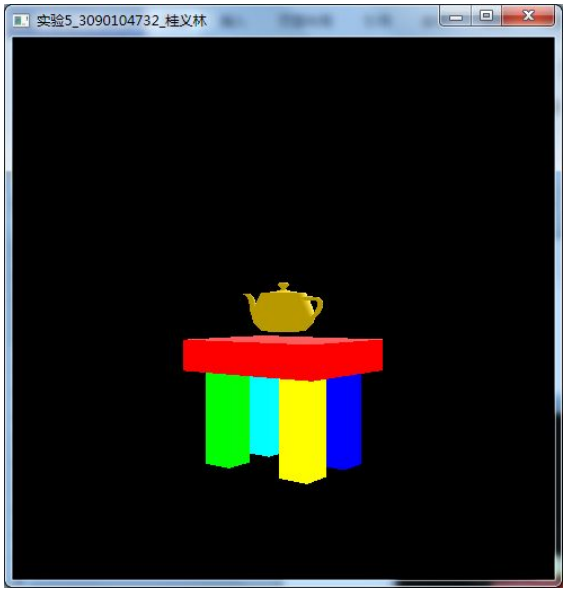


2.场景中光源的移动

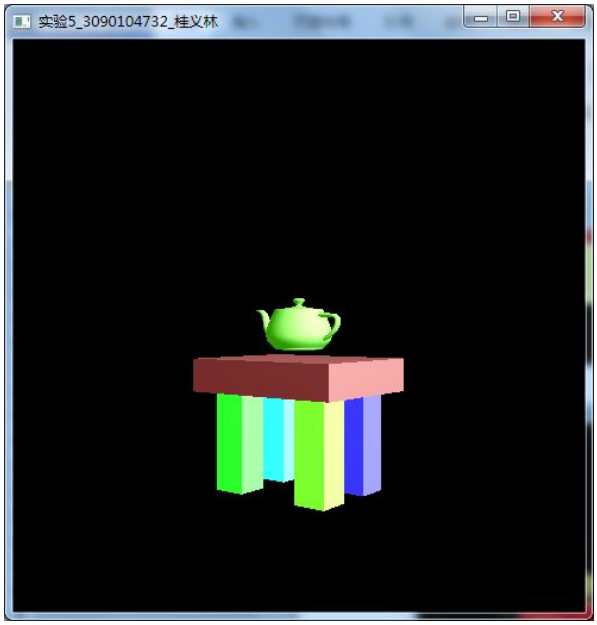
移动前：

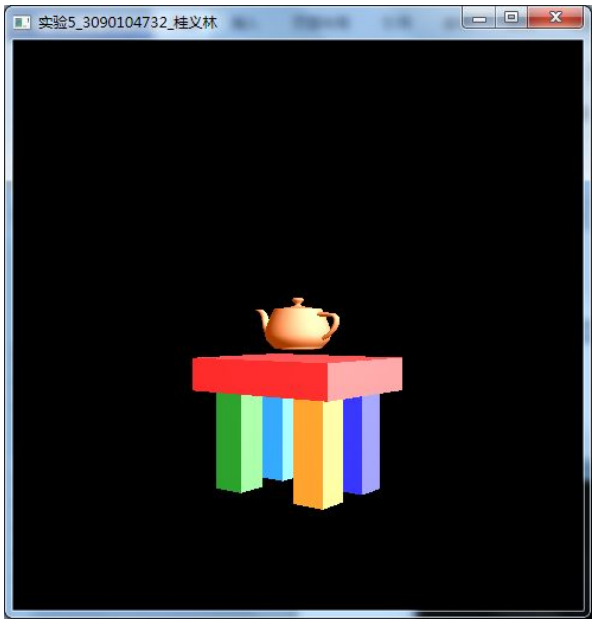


移动后：

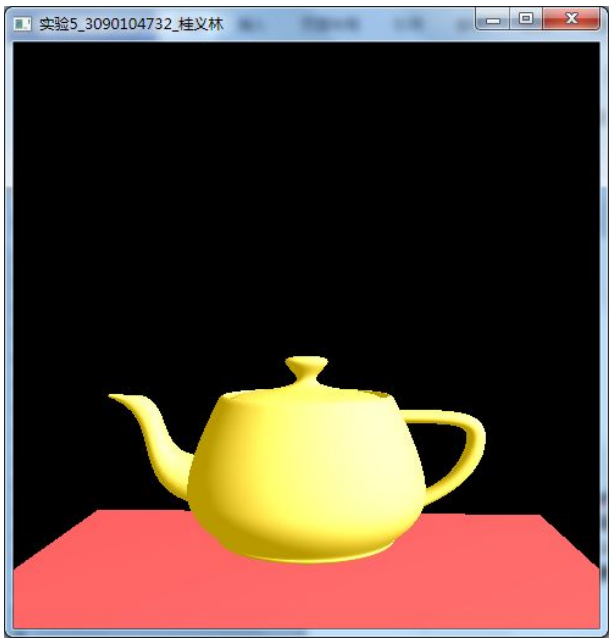


3.场景中光源颜色的改变

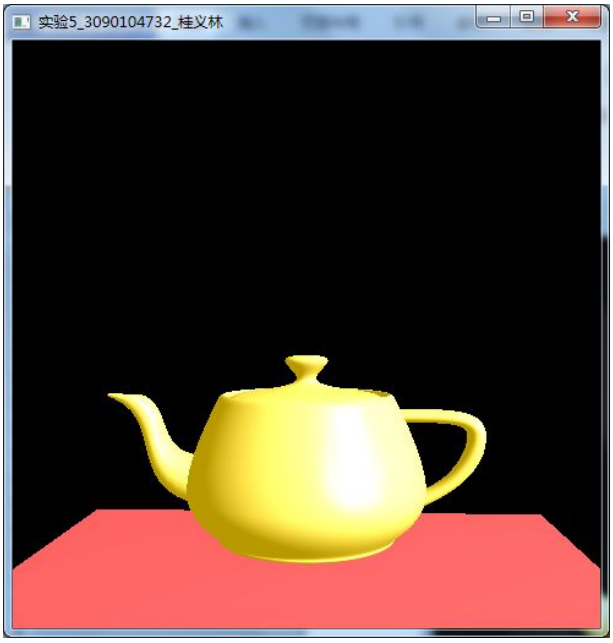




4.茶壶的高光效果

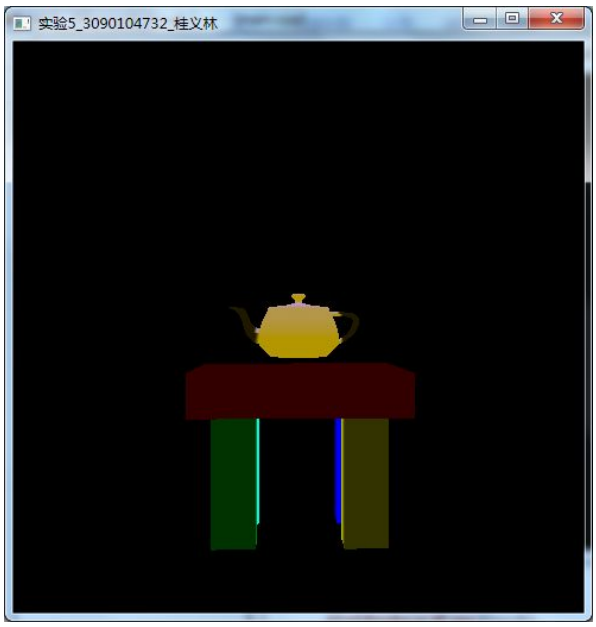


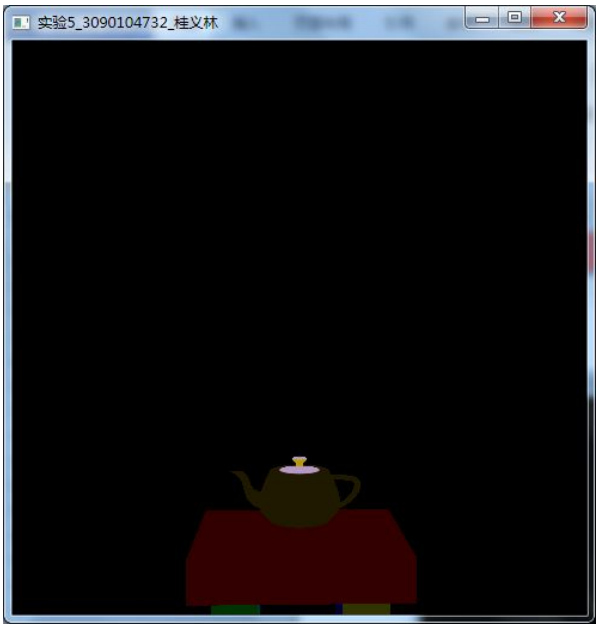
开启高光（左）



开启高光后（右）

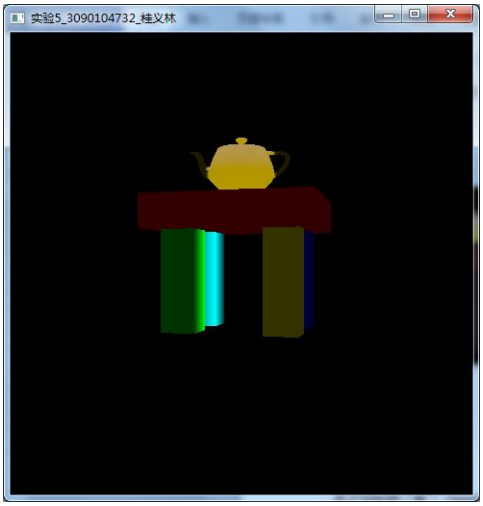
5.聚光光源





七、讨论、心得

本次实验我了解了 OpenGL 中如何进行光源、物体材质等属性的设置，从而实现光照效果。在实验中，一开始我没有对**物体的法向量**进行设置，光照效果有问题。由于本次实验并不是一个面一个面的绘制，而是使用 OpenGL 已有的 API 函数绘制（glSolidCube, glSolidTeapot），不能通过 glNormal3f()设置表面的法向，经过资料查阅我使用了**法向归一化**的方法，即 glEnable(GL_NORMALIZE);利用 opengl 系统自动对物体表面法向进行设置，但这样会降低程序的性能。此外，在设置聚光光源时，我的实验结果不仅照亮了茶壶，也照亮了桌子的 4 条腿（见下图），即没有处理好**物体遮挡关系**对光照的影响，通过调小角度可以使聚光光源仅照亮茶壶，但没有从根本上解决光线遮挡的问题，关于这个问题我认为应该物体相对光源的遮挡关系有选择的计算光照，但 OpenGL 中我没有找到相关的函数，可能在实际编程中需要根据需要自己编写判断的代码。



被照亮的桌腿