浙江大学实验报告

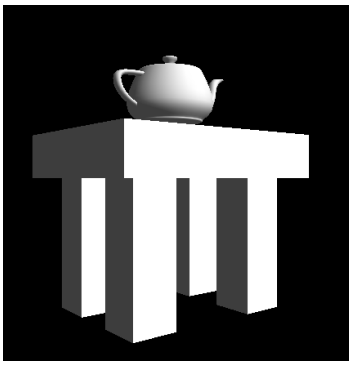
课程名称：____计算机图形学___ 指导老师：_____成绩：_____
实验名称：____OpenGL 消隐和光照_____ 实验类型：_____基础实验_____同组学生姓名：_____

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。



装

订

线

模型尺寸不做具体要求。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（在两种颜色间切换，颜色自己定义）；
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.zip
模板工程

四、实验原理及过程分析

1.设置桌子的材料及颜色

这里我们用到的是 glMaterial*()函数，其函数原型为 void glMaterial{if}(GLenum face, GLenum pname, TYPE param) 和 void glMaterial{if}v(GLenum face, GLenum pname, TYPE *param)，指定用于光照计算的当前材质属性。参数 face 的取值可以是 GL_FRONT（正向面）、GL_BACK（背相面）或 GL_FRONT_AND_BACK（正向面和背向面），指出材质属性将应用于物体的哪面。pname 指出要设置的哪种材质属性。param 为要设置的属性值，是一个指向数组的指针（向量版本）或一个数值（非向量版本）。只有设置参数值是 GL_SHININESS 时，才能使用非向量版本。其中在程序中使用的

GL_AMBIENT_AND_DIFFUSE 能够同时设置材质的环境颜色和散射颜色，并将它们设置为相同的 RGBA 值。下面的表格中列举出了在程序中参数 pname 的取值。

参数值	默认值	意义
GL_AMBIENT	（0.2，0.2，0.2，1.0）	材质的环境颜色
GL_DIFFUSE	（0.8，0.8，0.8，1.0）	材质的散射颜色
GL_AMBIENT_AND_DIFFUSE		材质的环境颜色和散射颜色
GL_SPECULAR	（0.0，0.0，0.0，1.0）	材质的镜面反射颜色
GL_SHININESS	0.0	镜面反射指数

其中，根据实验要求，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)，通过设置材料使得茶壶为金黄色（0.85, 0.65, 0.2, 1.0）。

与此相关的源代码如下：

```
1.  void Draw_Table() // Draw a table with RGB colors
2.  {
3.      GLfloat mat_specular[] = { 0.6f , 0.6f , 0.6f , 1.0f };
4.      GLfloat mat_diffuse[] = { 0.85f , 0.65f , 0.2f , 1.0f };  // This color is gold
5.
6.      // The color of table
7.      GLfloat color0[] = { 1.0f , 0.0f , 0.0f };
8.      GLfloat color1[] = { 0.0f , 1.0f , 0.0f };
9.      GLfloat color2[] = { 1.0f , 1.0f , 0.0f };
10.     GLfloat color3[] = { 0.0f , 1.0f , 1.0f };
11.     GLfloat color4[] = { 0.0f , 0.0f , 1.0f };
12.
13.     // The teapot
14.     glPushMatrix();
15.     glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);  // Set the specular color
16.     glMateriali(GL_FRONT_AND_BACK, GL_SHININESS, 50);  // Set specular reflection index
17.     glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, mat_diffuse);  // Set the ambient and
    diffuse property
18.     glTranslatef(0, 0, 4 + 1);
19.     glRotatef(90, 1, 0, 0);
20.     glutSolidTeapot(1);
21.     glPopMatrix();
22.
23.     glPushMatrix();
24.     glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, color0);  // Set the specular property
25.     glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE, color0);  // Set the ambient
    and diffuse property
26.     glTranslatef(0, 0, 3.5);
27.     glScalef(5, 4, 1);
```

```
    glutSolidCube(1.0);
    glPopMatrix();

    glPushMatrix();
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, color1);  // Set the specular property
    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE, color1);  // Set the ambient and diffuse property
    glTranslatef(1.5, 1, 1.5);
    Draw_Leg();
    glPopMatrix();

    glPushMatrix();
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, color2);  // Set the specular property
    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE, color2);  // Set the ambient and diffuse property
    glTranslatef(-1.5, 1, 1.5);
    Draw_Leg();
    glPopMatrix();

    glPushMatrix();
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, color3);  // Set the specular property
    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE, color3);  // Set the ambient and diffuse property
    glTranslatef(1.5, -1, 1.5);
    Draw_Leg();
    glPopMatrix();

    glPushMatrix();
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, color4);  // Set the specular property
    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE, color4);  // Set the ambient and diffuse property
    glTranslatef(-1.5, -1, 1.5);
    Draw_Leg();
    glPopMatrix();

}
```

2.光源

添加光照效果，更容易表现三维的物体，在前面的实验中提到，OpenGL 可以设置至少 8 种光源，它们的标号为 GL_LIGHT0,GL_LIGHT1,GL_LIGHT2……。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。在这里我们运用的创造光源的函数是 void glLightfv (GLenum light, GLenum pname, const GLfloat *params)。第一个参数 light 指定所创建的光源号，如 GL_LIGHT0、GL_LIGHT1、...、GL_LIGHT7；第二个参数 pname 制定光源特性；第三个参数设置相应的光源特性值。

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。

（1）环境光

我们通过全局变量来控制光源的位置及颜色，相关代码如下：

```
bool light_color = true;
GLfloat color[] = { 1.0 , 1.0 , 1.0 , 1.0 };  // Define the color of light

// The position of ambient light
GLfloat ambient_x = 0.0f;
GLfloat ambient_y = 0.0f;
GLfloat ambient_z = 0.0f;
```

在对环境光的设置中，通过 light_color 来控制环境光颜色。我们所用到的 glLightfv(GL_LIGHT0, GL_POSITION, ambient_pos)用于设置 0 号光源的位置属性，glLightfv(GL_LIGHT0, GL_SPECULAR, white)用于设置 0 号光源的镜面反射光照颜色，glLightfv(GL_LIGHT0, GL_DIFFUSE, white) 用于设置 0 号光源的漫反射光照颜色，glLightfv(GL_LIGHT0, GL_AMBIENT, color) 用于设置 0 号光源的环境光颜色。void glEnable(GLenum cap)函数用来启动各种功能，其中 cap 是一个参数值，每个参数值有不一样的功能，这里我们用 glEnable(GL_LIGHT0)来启动 0 号光源。相关代码如下：

```
GLfloat white[] = { 1.0, 1.0, 1.0, 1.0 };  // The color of white
GLfloat ambient_pos[] = { 5 + ambient_x , 5 + ambient_y , 5 + ambient_z , 1 };  // The po
sition of ambient light
```

```
if (light_color) {
    color[0] = 1.0f, color[1] = 1.0f, color[2] = 1.0f, color[3] = 1.0f;  // The color of
white
}
else {
    color[0] = 0.0f, color[1] = 1.0f, color[2] = 0.0f, color[3] = 1.0f;  // Another color

}

glLightfv(GL_LIGHT0, GL_POSITION, ambient_pos);  // Set the illumination position of the
0th light source
glLightfv(GL_LIGHT0, GL_SPECULAR, white);  // Set the specular lighting color
glLightfv(GL_LIGHT0, GL_DIFFUSE, white);  // Set the diffuse lighting color
glLightfv(GL_LIGHT0, GL_AMBIENT, color);  // Set the illumination color after the multipl
e reflection of the No. 0 light source (ambient light color)
glEnable(GL_LIGHT0);  // Turn on the 0th light source
```

（2）聚光灯

我们通过全局变量来控制聚光灯的方向及照射角度，相关代码如下：

```
//The direction and angle of  spot light
```

```
GLfloat spotdir_x = 0.0f;
GLfloat spotdir_y = 0.0f;
GLfloat spotdir_z = 0.0f;
GLfloat spot_angle = 5.0f;
```

在对聚光灯的设置中，我们所用到的 glLightfv(GL_LIGHT1, GL_POSITION, ambient_pos)用于设置 1 号光源的位置属性，glLightfv(GL_LIGHT1, GL_SPECULAR, white)用于设置 1 号光源的镜面反射光照颜色，glLightfv(GL_LIGHT1, GL_DIFFUSE, white) 用于设置 1 号光源的漫反射光照颜色，glLightfv(GL_LIGHT1, GL_AMBIENT, color) 用于设置 1 号光源的环境光颜色，glLightfv(GL_LIGHT1, GL_POSITION, spot_pos)用于设置 1 号光源位置，glLightf(GL_LIGHT1, GL_SPOT_CUTOFF, spot_angle)用于设置 1 号光源的裁剪角度，glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION, spot_direction) 用于设置 1 号光源的光源方向，glLightf(GL_LIGHT1, GL_SPOT_EXPONENT, 2.) 用于设置 1 号光源的聚集度，聚光灯有光源位置，也会随着传播距离增加而衰减，还有照射方向，另外聚光灯增加的特性是，它的照射范围在一个圆锥内，类似探照灯的效果。我们对位置性光源的形状加以限制，使它的发射范围限于一个椎体之内，类似聚光灯的效果。为此，需要确定光锥的发射范围。为了指定光锥轴和光锥边缘之间的角度，这里使用 GL_SPOT_CUTOFF 参数。使得光锥的最大角度是这个值的两倍。glEnable(GL_LIGHT1)用于启动 1 号光源。相关代码如下：

```
GLfloat spot_pos[] = { 0.0f , 5.0f , 0.0f , 1.0f };  // The position of spot light
GLfloat spot_direction[] = { 0.0f + spotdir_x , -1.0f + spotdir_y , 0.0f + spotdir_z };
// The direction of spot light
```

```
glLightfv(GL_LIGHT1, GL_AMBIENT, color);  // Set ambient light composition
glLightfv(GL_LIGHT1, GL_SPECULAR, white);  // Set specular light composition
glLightfv(GL_LIGHT1, GL_DIFFUSE, white);  // Set diffuse light composition

glLightfv(GL_LIGHT1, GL_POSITION, spot_pos);
glLightf(GL_LIGHT1, GL_SPOT_CUTOFF, spot_angle);  // Cut angle
glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION, spot_direction);  // Light source direction
glLightf(GL_LIGHT1, GL_SPOT_EXPONENT, 2.);  // Aggregation
glEnable(GL_LIGHT1);  // Turn on the first light source
```

3.按键设置

Q：退出

P：切换投影方式（正投影与透视投影）

空格键：启动与暂停旋转（桌子与茶壶一起绕桌子中心轴旋转）

O 切换渲染方式（填充模式与线框模式）

WASDZC：控制相机上下左右前后移动

IKJLRY：控制环境光上下左右前后移动

X：环境光颜色转换

TGFHVB：控制聚光灯上下左右前后移动

NM：控制聚光灯角度大小

相关代码如下：

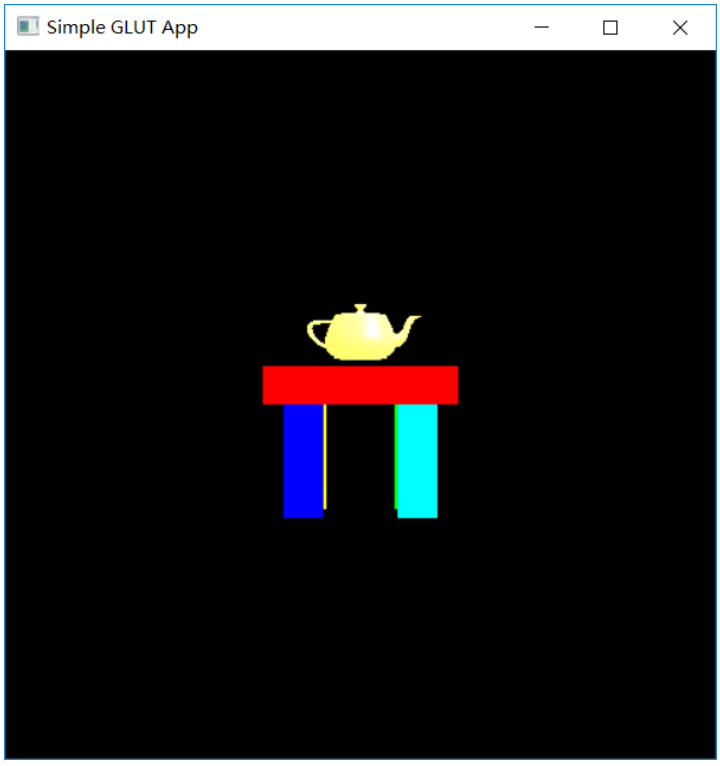
```
1.  void key(unsigned char k, int x, int y)
2.  {
3.      switch (k)
4.      {
5.      case 27:
6.      case 'q': {exit(0); break; }  // exit
7.      case 'p': {bPersp = !bPersp; updateView(wHeight, wWidth); break; }  //Switch orthogra
    phic projection and perspective projection
8.
9.      case ' ': {bAnim = !bAnim; break; }  // Switch the rotation mode
10.     case 'o': {bWire = !bWire; break; }  // Switch the rendering mode
11.
12.     case 'a': {// Move left
13.         eye[0] += 0.2f;
14.         center[0] += 0.2f;
15.         break;
16.     }
17.     case 'd': {// Move right
18.         eye[0] -= 0.2f;
19.         center[0] -= 0.2f;
20.         break;
21.     }
22.     case 'w': {// Move up
23.         eye[1] -= 0.2f;
24.         center[1] -= 0.2f;
25.         break;
26.     }
27.     case 's': {// Move down
28.         eye[1] += 0.2f;
29.         center[1] += 0.2f;
30.         break;
31.     }
32.     case 'z': {// Move forward
33.         eye[2] -= 0.2f;
34.         center[2] -= 0.2f;
35.         break;
36.     }
37.     case 'c': {// Move backwards
38.         eye[2] += 0.2f;
39.         center[2] += 0.2f;
40.         break;
41.     }
42.     case 'j': {// Ambient light moves left
```

```
43.         ambient_x = ambient_x - 0.2f;
44.         break;
45.     }
46.     case 'l': {// Ambient light moves right
47.         ambient_x = ambient_x + 0.2f;
48.         break;
49.     }
50.     case 'i': {// Ambient light moves up
51.         ambient_y = ambient_y + 0.2f;
52.         break;
53.     }
54.     case 'k': {// Ambient light moves down
55.         ambient_y = ambient_y - 0.2f;
56.         break;
57.     }
58.     case 'r': {// Ambient light moves forward
59.         ambient_z = ambient_z + 0.2f;
60.         break;
61.     }
62.     case 'y': {// Ambient light moves backwards
63.         ambient_z = ambient_z - 0.2f;
64.         break;
65.     }
66.     case 'x': {// Ambient light color switching
67.         light_color = !light_color;
68.         break;
69.     }
70.     case 'f': {// Spot light moves left
71.         spotdir_x = spotdir_x - 0.05f;
72.         break;
73.     }
74.     case 'h': {// Spot light moves right
75.         spotdir_x = spotdir_x + 0.05f;
76.         break;
77.     }
78.     case 't': {// Spot light moves up
79.         spotdir_y = spotdir_y - 0.05f;
80.         break;
81.     }
82.     case 'g': {// Spot light moves down
83.         spotdir_y = spotdir_y + 0.05f;
84.         break;
85.     }
86.     case 'v': {// Spot light moves forward
```

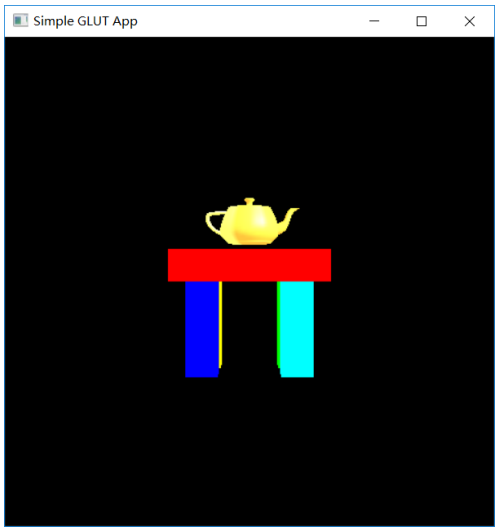
```
87.         spotdir_z = spotdir_z + 0.05f;
88.         break;
89.     }
90.     case 'b': {// Spot light moves backwards
91.         spotdir_z = spotdir_z - 0.05f;
92.         break;
93.     }
94.     case 'n': {// Spotlight angle becomes larger
95.         if (spot_angle <= 89.0f)
96.             spot_angle = spot_angle + 0.2f;
97.         break;
98.     }
99.     case 'm': {// Spotlight angle becomes smaller
100.         if (spot_angle >= 1.0f)
101.             spot_angle = spot_angle - 0.2f;
102.         break;
103.     }
104.     }
105.
106.     updateView(wHeight, wWidth);
107. }
```

五、实验结果与分析

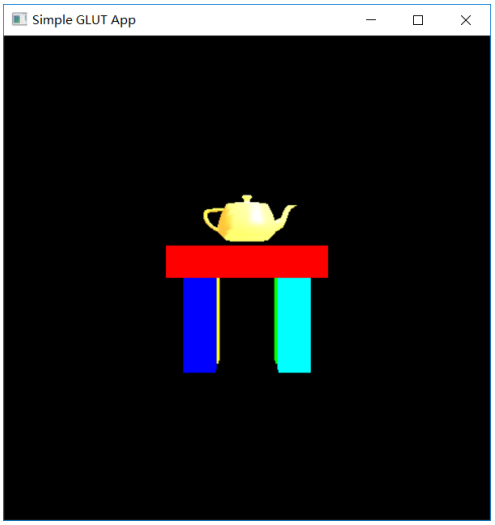
初始：



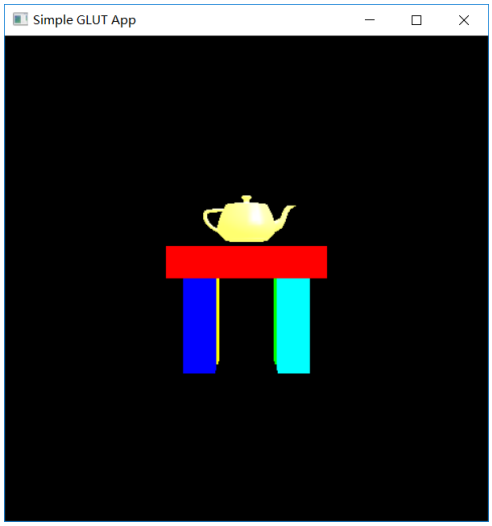
1.聚光灯



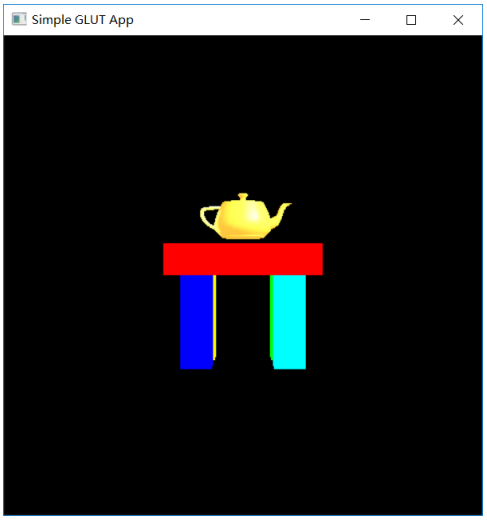
聚光灯左移



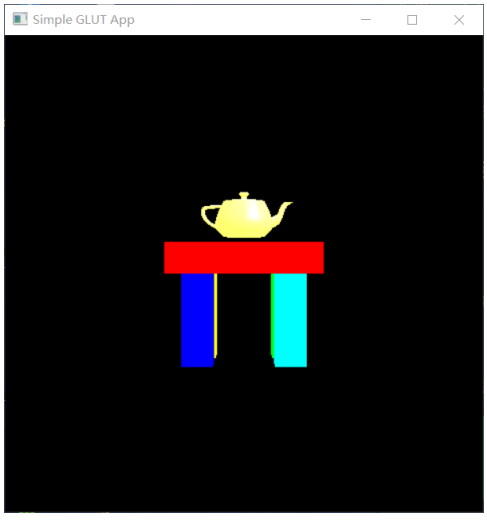
聚光灯右移



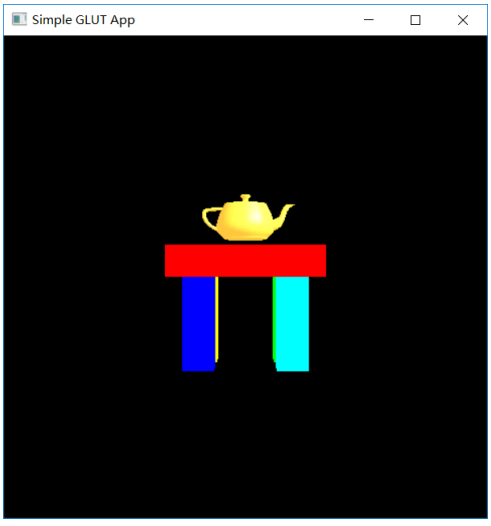
聚光灯上移



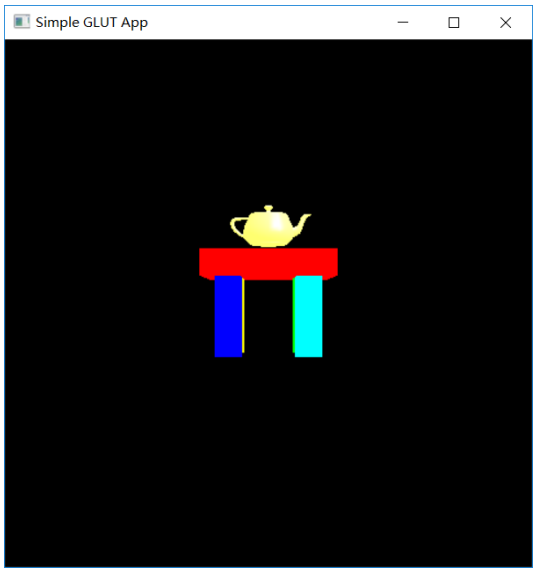
聚光灯下移



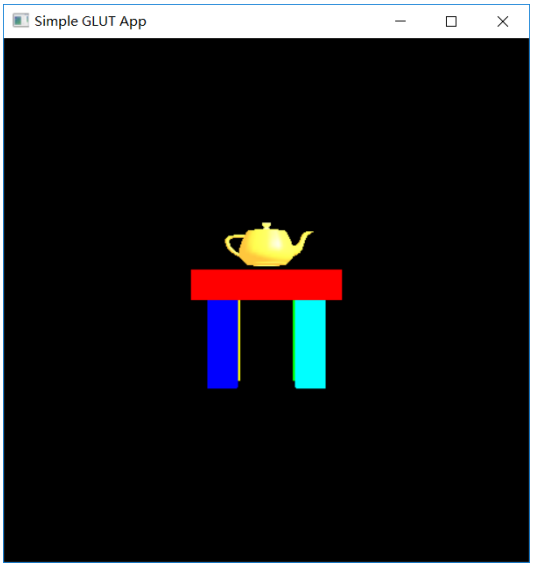
聚光灯前移



聚光灯后移

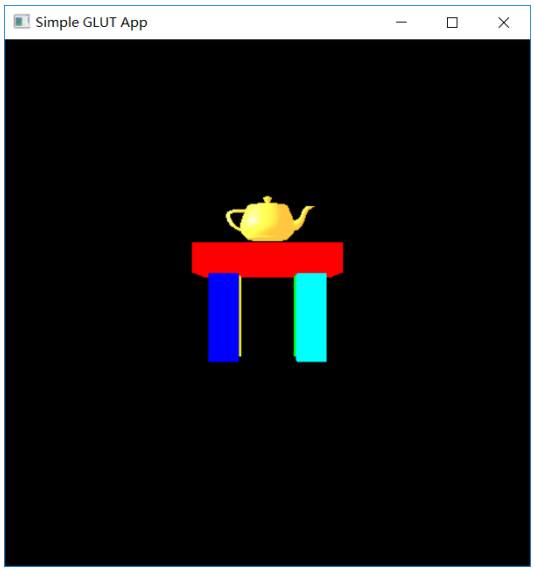


聚光灯角度变大

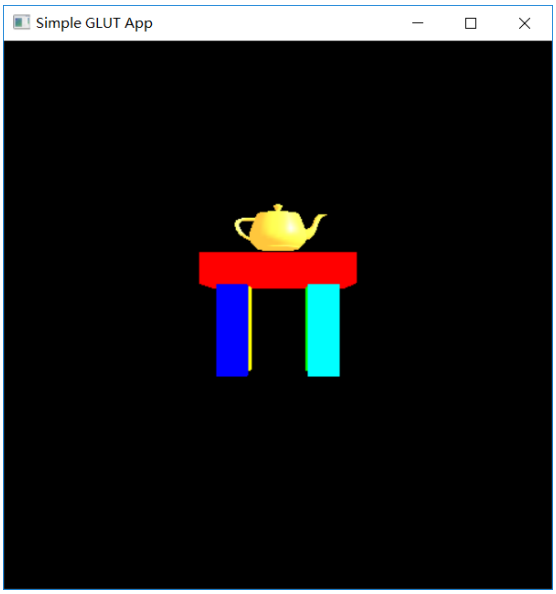


聚光灯角度变小

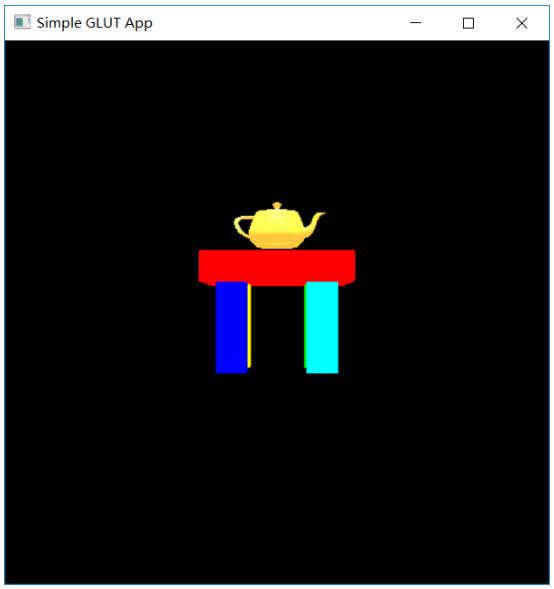
2.环境光



环境光左移

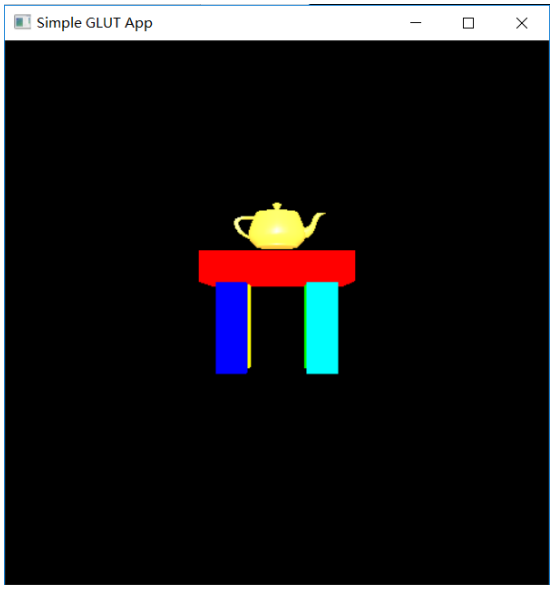


环境光右移

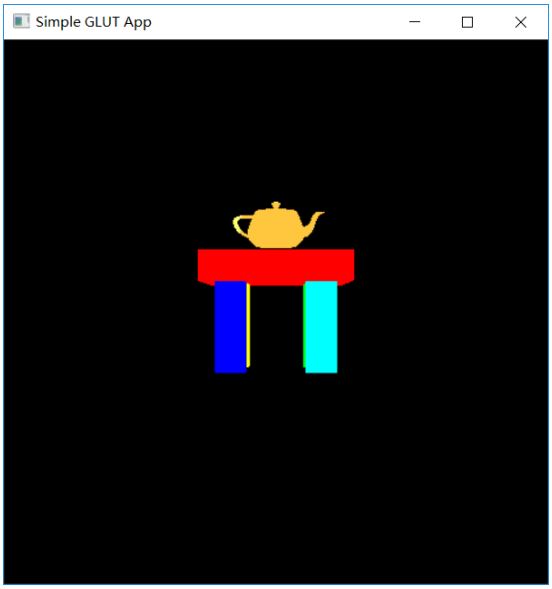


环境光上移

环境光下移



环境光前移



环境光后移

环境光转换颜色

由实验结果可以看出，当聚光灯或者环境光的位置发生变化时，对于物体的光照效果是非常不同的。在本次实验中，通过运用 OpenGL，在观察实验的基础上和实现实验内容的过程中，学会了 OpenGL 中消隐和光照的设置，并通过实验验证了课程中消隐和光照的理论内容。

六、 源代码

```
#include <stdlib.h>
#include "GL/glut.h"

float fTranslate;
float fRotate;
float fScale = 1.0f;    // set inital scale value to 1.0f

bool bPersp = false;  // Judge whether it is a perspective projection or a orthographic p
rojection
bool bAnim = false;  // Determine if the teapot and table rotate
bool bWire = false;  // Determine if the drawing mode is linear or filled

int wHeight = 0;
int wWidth = 0;

bool light_color = true;
GLfloat color[] = { 1.0 , 1.0 , 1.0 , 1.0 };  // Define the color of light

// The position of ambient light
GLfloat ambient_x = 0.0f;
GLfloat ambient_y = 0.0f;
GLfloat ambient_z = 0.0f;

//The direction and angle of  spot light
GLfloat spotdir_x = 0.0f;
GLfloat spotdir_y = 0.0f;
GLfloat spotdir_z = 0.0f;
GLfloat spot_angle = 5.0f;

void Draw_Leg()  // Draw a leg
{
    glScalef(1, 1, 3);  // Stretch the model three times in the z direction
    glutSolidCube(1.0);  // Draw a cube with a side length of one
}

void Draw_Table() // Draw a table with RGB colors
{
    GLfloat mat_specular[] = { 0.6f , 0.6f , 0.6f , 1.0f };
    GLfloat mat_diffuse[] = { 0.85f , 0.65f , 0.2f , 1.0f };  // This color is gold

```

```
40.     // The color of table
41.     GLfloat color0[] = { 1.0f , 0.0f , 0.0f };
42.     GLfloat color1[] = { 0.0f , 1.0f , 0.0f };
43.     GLfloat color2[] = { 1.0f , 1.0f , 0.0f };
44.     GLfloat color3[] = { 0.0f , 1.0f , 1.0f };
45.     GLfloat color4[] = { 0.0f , 0.0f , 1.0f };
46.
47.     // The teapot
48.     glPushMatrix();
49.     glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);  // Set the specular color
50.     glMateriali(GL_FRONT_AND_BACK, GL_SHININESS, 50);  // Set specular reflection index
51.     glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, mat_diffuse);  // Set the ambient and
    diffuse property
52.     glTranslatef(0, 0, 4 + 1);
53.     glRotatef(90, 1, 0, 0);
54.     glutSolidTeapot(1);
55.     glPopMatrix();
56.
57.     glPushMatrix();
58.     glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, color0);  // Set the specular property
59.     glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE, color0);  // Set the ambient
    and diffuse property
60.     glTranslatef(0, 0, 3.5);
61.     glScalef(5, 4, 1);
62.     glutSolidCube(1.0);
63.     glPopMatrix();
64.
65.     glPushMatrix();
66.     glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, color1);  // Set the specular property
67.     glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE, color1);  // Set the ambient
    and diffuse property
68.     glTranslatef(1.5, 1, 1.5);
69.     Draw_Leg();
70.     glPopMatrix();
71.
72.     glPushMatrix();
73.     glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, color2);  // Set the specular property
74.     glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE, color2);  // Set the ambient
    and diffuse property
75.     glTranslatef(-1.5, 1, 1.5);
76.     Draw_Leg();
77.     glPopMatrix();
78.
79.     glPushMatrix();
```

```
80.     glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, color3);  // Set the specular property
81.     glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE, color3);  // Set the ambient
    and diffuse property
82.     glTranslatef(1.5, -1, 1.5);
83.     Draw_Leg();
84.     glPopMatrix();
85.
86.     glPushMatrix();
87.     glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, color4);  // Set the specular property
88.     glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE, color4);  // Set the ambient
    and diffuse property
89.     glTranslatef(-1.5, -1, 1.5);
90.     Draw_Leg();
91.     glPopMatrix();
92.
93. }
94.
95.
96. void updateView(int width, int height)
97. {
98.     glViewport(0, 0, width, height);                        // Reset the current viewport

99.
100.     glMatrixMode(GL_PROJECTION);                          // Select the projection matrix

101.     glLoadIdentity();                                     // Reset the projection matrix
102.
103.     float whRatio = (GLfloat)width / (GLfloat)height;  // Set the display scale
104.     if (bPersp) {
105.         gluPerspective(45.0f, whRatio, 0.1f, 100.0f);  // Perspective mode, the paramete
    rs of the function are angle of view, aspect ratio, near, far
106.         //glFrustum(-3, 3, -3, 3, 3,100);
107.     }
108.     else {
109.         glOrtho(-3, 3, -3, 3, -100, 100);
110.     }
111.
112.     glMatrixMode(GL_MODELVIEW);                           // Select the modelview matrix
113. }
114.
115. void reshape(int width, int height)
116. {
117.     if (height == 0)                                        // Prevent A divide by zero
```

```
118.     {
119.         height = 1;                                    // Make Height Equal One
120.     }
121.
122.     wHeight = height;
123.     wWidth = width;
124.
125.     updateView(wHeight, wWidth);
126. }
127.
128. void idle()
129. {
130.     glutPostRedisplay();  // Call the current drawing function
131. }
132.
133. float eye[] = { 0, 0, 8 };  // The place of the camera
134. float center[] = { 0, 0, 0 };  // The place of the object
135.
136. void key(unsigned char k, int x, int y)
137. {
138.     switch (k)
139.     {
140.     case 27:
141.     case 'q': {exit(0); break; }  // exit
142.     case 'p': {bPersp = !bPersp; updateView(wHeight, wWidth); break; }  //Switch orthogr
    aphic projection and perspective projection
143.
144.     case ' ': {bAnim = !bAnim; break; }  // Switch the rotation mode
145.     case 'o': {bWire = !bWire; break; }  // Switch the rendering mode
146.
147.     case 'a': {// Move left
148.         eye[0] += 0.2f;
149.         center[0] += 0.2f;
150.         break;
151.     }
152.     case 'd': {// Move right
153.         eye[0] -= 0.2f;
154.         center[0] -= 0.2f;
155.         break;
156.     }
157.     case 'w': {// Move up
158.         eye[1] -= 0.2f;
159.         center[1] -= 0.2f;
160.         break;
```

```
161.     }
162.     case 's': {// Move down
163.         eye[1] += 0.2f;
164.         center[1] += 0.2f;
165.         break;
166.     }
167.     case 'z': {// Move forward
168.         eye[2] -= 0.2f;
169.         center[2] -= 0.2f;
170.         break;
171.     }
172.     case 'c': {// Move backwards
173.         eye[2] += 0.2f;
174.         center[2] += 0.2f;
175.         break;
176.     }
177.     case 'j': {// Ambient light moves left
178.         ambient_x = ambient_x - 0.2f;
179.         break;
180.     }
181.     case 'l': {// Ambient light moves right
182.         ambient_x = ambient_x + 0.2f;
183.         break;
184.     }
185.     case 'i': {// Ambient light moves up
186.         ambient_y = ambient_y + 0.2f;
187.         break;
188.     }
189.     case 'k': {// Ambient light moves down
190.         ambient_y = ambient_y - 0.2f;
191.         break;
192.     }
193.     case 'r': {// Ambient light moves forward
194.         ambient_z = ambient_z + 0.2f;
195.         break;
196.     }
197.     case 'y': {// Ambient light moves backwards
198.         ambient_z = ambient_z - 0.2f;
199.         break;
200.     }
201.     case 'x': {// Ambient light color switching
202.         light_color = !light_color;
203.         break;
204.     }
```

```
205.     case 'f': {// Spot light moves left
206.         spotdir_x = spotdir_x - 0.05f;
207.         break;
208.     }
209.     case 'h': {// Spot light moves right
210.         spotdir_x = spotdir_x + 0.05f;
211.         break;
212.     }
213.     case 't': {// Spot light moves up
214.         spotdir_y = spotdir_y - 0.05f;
215.         break;
216.     }
217.     case 'g': {// Spot light moves down
218.         spotdir_y = spotdir_y + 0.05f;
219.         break;
220.     }
221.     case 'v': {// Spot light moves forward
222.         spotdir_z = spotdir_z + 0.05f;
223.         break;
224.     }
225.     case 'b': {// Spot light moves backwards
226.         spotdir_z = spotdir_z - 0.05f;
227.         break;
228.     }
229.     case 'n': {// Spotlight angle becomes larger
230.         if (spot_angle <= 89.0f)
231.             spot_angle = spot_angle + 0.2f;
232.         break;
233.     }
234.     case 'm': {// Spotlight angle becomes smaller
235.         if (spot_angle >= 1.0f)
236.             spot_angle = spot_angle - 0.2f;
237.         break;
238.     }
239.     }
240.
241.     updateView(wHeight, wWidth);
242. }
243.
244.
245. void redraw()
246. {
247.
```

```
248.     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);  // Clear color cache and depth
    cache
249.     glLoadIdentity();                                    // Reset The Current Modelview M
    atrix
250.
251.     // The place of the camera,
252.     // the place of the object
253.     // and the observation direction
254.     gluLookAt(eye[0], eye[1], eye[2],
255.         center[0], center[1], center[2],
256.         0, 1, 0);
257.
258.     if (bWire) {
259.         glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);  // Set the polygon drawing mode: fro
    nt and back, line type
260.     }
261.     else {
262.         glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);  // Set the polygon drawing mode: fro
    nt and back, fill type
263.     }
264.
265.     glEnable(GL_DEPTH_TEST);  // Open depth test
266.     glEnable(GL_LIGHTING);  // Turn on lighting mode
267.
268.     GLfloat white[] = { 1.0, 1.0, 1.0, 1.0 };  // The color of white
269.     GLfloat ambient_pos[] = { 5 + ambient_x , 5 + ambient_y , 5 + ambient_z , 1 };  // T
    he position of ambient light
270.     GLfloat spot_pos[] = { 0.0f , 5.0f , 0.0f , 1.0f };  // The position of spot light
271.     GLfloat spot_direction[] = { 0.0f + spotdir_x , -1.0f + spotdir_y , 0.0f + spotdir_z
     };  // The direction of spot light
272.
273.     if (light_color) {
274.         color[0] = 1.0f, color[1] = 1.0f, color[2] = 1.0f, color[3] = 1.0f;  // The colo
    r of white
275.     }
276.     else {
277.         color[0] = 0.0f, color[1] = 1.0f, color[2] = 0.0f, color[3] = 1.0f;  // Another
    color
278.     }
279.
280.     glLightfv(GL_LIGHT0, GL_POSITION, ambient_pos);  // Set the illumination position of
     the 0th light source
281.     glLightfv(GL_LIGHT0, GL_SPECULAR, white);  // Set the specular lighting color
282.     glLightfv(GL_LIGHT0, GL_DIFFUSE, white);  // Set the diffuse lighting color
```

```
283.     glLightfv(GL_LIGHT0, GL_AMBIENT, color);  // Set the illumination color after the mu
    ltiple reflection of the No. 0 light source (ambient light color)
284.     glEnable(GL_LIGHT0);  // Turn on the 0th light source
285.
286.     glLightfv(GL_LIGHT1, GL_AMBIENT, color);  // Set ambient light composition
287.     glLightfv(GL_LIGHT1, GL_SPECULAR, white);  // Set specular light composition
288.     glLightfv(GL_LIGHT1, GL_DIFFUSE, white);  // Set diffuse light composition
289.
290.     glLightfv(GL_LIGHT1, GL_POSITION, spot_pos);
291.     glLightf(GL_LIGHT1, GL_SPOT_CUTOFF, spot_angle);  // Cut angle
292.     glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION, spot_direction);  // Light source direction

293.     glLightf(GL_LIGHT1, GL_SPOT_EXPONENT, 2.);  // Aggregation
294.     glEnable(GL_LIGHT1);  // Turn on the first light source
295.
296.
297.     //  glTranslatef(0.0f, 0.0f,-6.0f);         // Place the triangle at Center
298.     glRotatef(fRotate, 0, 1.0f, 0);  // Rotate around Y axis
299.     glRotatef(-90, 1, 0, 0);  // Make the table facing the camera
300.     glScalef(0.2, 0.2, 0.2);  // Scale to make the object appear in the window at the ap
    propriate size
301.     Draw_Table();  // Draw Scene
302.
303.     if (bAnim) fRotate += 0.5f;  // Rotation factor change
304.     glutSwapBuffers();  // Swap buffer
305. }
306.
307. int main(int argc, char *argv[])
308. {
309.     glutInit(&argc, argv);  // Initialize the glut library
310.     glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH | GLUT_DOUBLE);  // Specify the window di
    splay mode that the function glutCreateWindow will create. RGB mode Double buffering
311.     glutInitWindowSize(480, 480);  // Set the window position, which is the position of
    the top left corner of the window relative to the entire screen
312.     int windowHandle = glutCreateWindow("Simple GLUT App");  // Set the window title
313.
314.     glutDisplayFunc(redraw);  // Register a draw callback function that specifies the fu
    nction to call when the window content needs to be redrawn
315.     glutReshapeFunc(reshape); // The callback function when the registration window size
     changes.
316.     glutKeyboardFunc(key);  // Register key callback function
317.     glutIdleFunc(idle);  // Register global callback function : call when idle
318.
319.     glutMainLoop();  // Glut event processing loop
```

```
320.     return 0;
321. }
```