计算机图形学
编程作业：OpenGL 格网显示

姓名：杨杰　　班级：111172　　学号：20171002157

目录

一、基本要求：.....1

二、题目要求与具体实现：.....1

（一）改变颜色 COLOR CHANGES.....1
（二）改变光源位置.....2
（三）格网装入和显示.....4
（四）自动旋转模型及颜色光滑渐变.....6
（五）实现基于 MOUSE 的相机控制.....7
（六）使用 OBV 实现高速渲染.....10

一、基本要求：

在本次作业中，需要实现一个简单的 3D 格网显示。所编写的程序除了显示物体的模型外，还要能让用户改变光的位置和物体的颜色。

二、题目要求与具体实现：

（一）改变颜色 Color Changes

实现描述：

增加全局变量 float matchange，在 display()函数中调用 glMaterialfv(GL_FRONT, GL_DIFFUSE, mat)，其中 mat 为数组存储颜色的值，在函数 keyboard()中的 switch 语句中添加键盘“c”对应的功能语句，当用户输入“c”时，修改 matchange 的值，并调用 glutPostRedisplay()函数，从而实现变色。

测试截图：

具体代码：

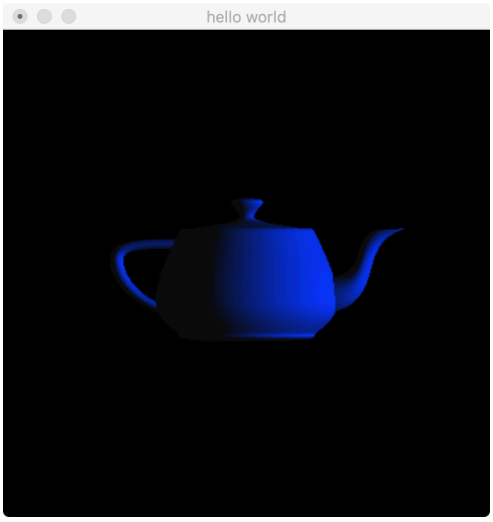
```
float matchange=0;//改变画颜色
void display(void){
    GLfloat mat[4];//颜色改变
    //……
    mat[0] = matchange; mat[1] = 0.2; mat[2] = 0.9;
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat);
    //……
}
void keyboard(unsigned char key, int x, int y){
    switch (key){
    //……
        case 99://c 切换颜色
            if(matchange<=0.9) matchange+=0.1;
            else matchange=0;
            glutPostRedisplay();
            break;
    //……
}
```

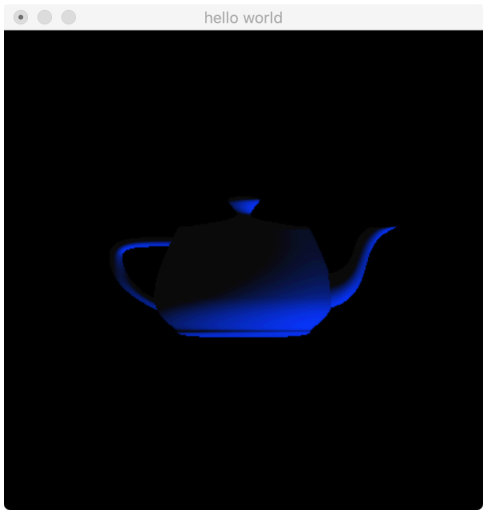
（二）改变光源位置

实现描述：

如上题实现，添加全局变量、对应函数中代码补全等。这里使用了 w、s、a、d 键改变光源位置。

测试截图：





具体代码：

```
float positionx=1;//光源 x
float positiony=1;//光源 y
void display(void){
    //……
    GLfloat position[4];//光线改变
    position[0]=positionx;position[1]=positiony;position[2]=5;position[3]=0;
    glLightfv(GL_LIGHT0, GL_POSITION, position);
    //……
}
void keyboard(unsigned char key, int x, int y){
    switch (key){
    //……
            case 119://w 方向上键
                cout<<key;
                positiony+=10;
                glutPostRedisplay();
                break;
            case 115://s 方向下键
                cout<<key;
                positiony-=10;
                glutPostRedisplay();
                break;
            case 97://a 方向左键
                cout<<key;
                positionx-=10;
                glutPostRedisplay();
                break;
            case 100://d 方向右键
```

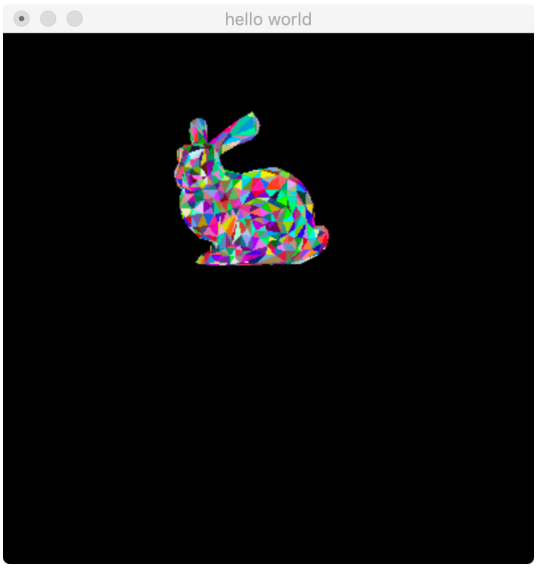
```
                cout<<key;
                positionx+=10;
                glutPostRedisplay();
                break;//……
}
```

（三）格网装入和显示

实现描述：

这里写了一个 readobj 类，通过传入文件地址填充 vector<vector<float>> varr 存放点的数组，vector<vector<int>> farr 存放面的数组。使用时从类中传出对应的数组，根据数组格式画三角形即可。这里在主函数中添加了一个 bool 的 flag 选择显示茶壶或者读取的 obj 文件。

测试截图：（主函数设置 flag=2 启动）



具体代码：

```
//readobj.h
#include <stdio.h>
#include <GLUT/GLUT.h>
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
using namespace std;
```

```
class readobj{
private:
    vector<vector<float>> varr;//存放点的数组;
    vector<vector<int>> farr;//存放面的数组;
    string filename;
public:
    readobj(string name);
    void getnums();
    void display();
    vector<vector<float>> getvarr(){return varr;}
    vector<vector<int>> getfarr(){return farr;}
};

//readobj.cpp
#include "readobj.hpp"

readobj::readobj(string name){
    filename=name;
    getnums();
}

void readobj::getnums(){
    fstream file(filename.c_str());
    if(!file.is_open())
        cout<<"Open filed!"<<endl;
    else{
        char type;
        float a,b,c;
        while(!file.eof()){
            file>>type>>a>>b>>c;
            //cout<<type<<" "<<a<<" "<<b<<" "<<c<<endl;
            if(type=='v'){
                vector<float>temp{a,b,c};
                varr.push_back(temp);
            }
            else if(type=='f'){
            vector<int>temp{(int)a,(int)b,(int)c};
            farr.push_back(temp);
            }
        }
   }
    file.close();
}
```

```
//display()函数增添
void display(void){
    //……
    if(flag==0)
        glutSolidTeapot(3.0);
    else{
        for (int i=0;i<farr.size();i++){
            glBegin(GL_TRIANGLES);
            if(allchange){
                mat[0] = rand()/double(RAND_MAX);
                mat[1] = rand()/double(RAND_MAX);
                mat[2] = rand()/double(RAND_MAX);
                glMaterialfv(GL_FRONT, GL_DIFFUSE, mat);
            }
            glVertex3f(varr[farr[i][0]-1][0], varr[farr[i][0]-1][1],
varr[farr[i][0]-1][2]);
            glVertex3f(varr[farr[i][1]-1][0], varr[farr[i][1]-1][1],
varr[farr[i][1]-1][2]);
            glVertex3f(varr[farr[i][2]-1][0], varr[farr[i][2]-1][1],
varr[farr[i][2]-1][2]);
            glEnd();
        }
    }
```

（四）自动旋转模型及颜色光滑渐变

实现描述：

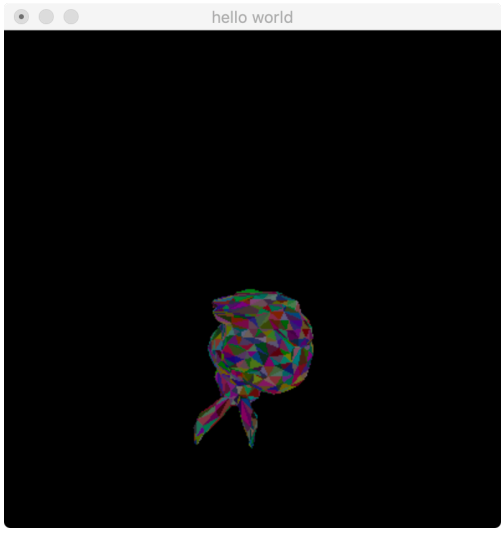
添加全局变量 float rotateangle=0;//旋转角度 float rotatex=0;//旋转 x 轴 float rotatey=0;//旋转 y 轴 float rotatez=1;//选择 z 轴。Display()添加 glRotatef(rotateangle,rotatex,rotatey,rotatez)，并添加对应键盘值，其中 r 是手动旋转，t 是改变 x 轴，y 是改变 y 轴，z 是改变 z 轴。

添加全局变量 bool rotatechange=false;//自动旋转，在 timerFunc(int value)函数中添加自动旋转，并添加启动终止键盘 e 键。

添加全局变量 bool colorchange=false，在 timerFunc(int value)函数中添加自动变色，并添加启动终止键盘 q 键。

测试截图：

（可见附 git 图片）



具体代码：

```
void timerFunc(int value){
    if(colorchange){//自动变换颜色
        if(matchange<=0.9) matchange+=0.1;
        else matchange=0;
    }
    if (rotatechange) {//自动旋转模型
        rotateangle+=5;
    }
    glutPostRedisplay();
    glutTimerFunc(33, timerFunc, 1);
}
```

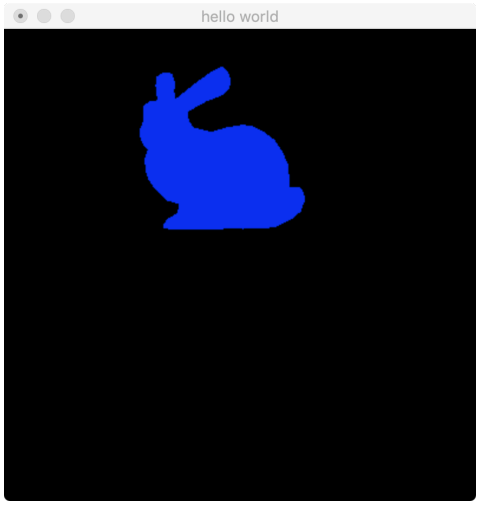
（五）实现基于 mouse 的相机控制

实现思路：

完成 void mouse(int button, int state, int x, int y)函数和 void mouseMotion(int x, int y)函数，内容为书鼠标状态的获取和修改相应的参数，主函数添加 glutMouseFunc(mouse)及 glutMotionFunc(mouseMotion);语句。

测试截图：

（可见附 git 图片）



具体代码：

```
//鼠标相机控制需要的参数
bool mouseLeftDown;//左键按下
bool mouseRightDown;//右键按下
bool mouseMiddleDown;//前一刻鼠标位置
float mouseX, mouseY;//前一刻鼠标位置
float cameraDistanceX;//右键物体平移
float cameraDistanceY;//右键物体平移
float cameraAngleX;//左键旋转角度
float cameraAngleY;//左键旋转角度

//鼠标缩放实现
void OnMouse(int button,int state,int x,int y){
    if(button==GLUT_LEFT_BUTTON&&state==GLUT_DOWN){//放大
        sizenum+=1;
        glutPostRedisplay();
    }
    else if(button==GLUT_RIGHT_BUTTON&&state==GLUT_DOWN){//缩小
        sizenum-=1;
        glutPostRedisplay();
    }
    else if(button==GLUT_MIDDLE_BUTTON&&state==GLUT_DOWN){//旋转
        rotatechange=!rotatechange;
        glutPostRedisplay();
    }
}

//鼠标状态获取
void mouse(int button, int state, int x, int y){
    mouseX = x;
    mouseY = y;
    if(button == GLUT_LEFT_BUTTON){
```

```
        if(state == GLUT_DOWN){
            mouseLeftDown = true;
        }
        else if(state == GLUT_UP)
            mouseLeftDown = false;
    }

    else if(button == GLUT_RIGHT_BUTTON){
        if(state == GLUT_DOWN){
            mouseRightDown = true;
        }
        else if(state == GLUT_UP)
            mouseRightDown = false;
    }

    //滚轮缩放模型
    else if (state == GLUT_UP && button == GLUT_MIDDLE_BUTTON)
    {
        sizenum++;
        glutPostRedisplay();
    }

    else if (state == GLUT_UP && button == GLUT_MIDDLE_BUTTON)
    {
        sizenum--;
        glutPostRedisplay();
    }
}

void mouseMotion(int x, int y){
    cameraAngleX = cameraAngleY = 0;
    cameraDistanceX = cameraDistanceY = 0;

    if (mouseLeftDown){
        cameraAngleY += (x - mouseX) * 0.1;
        cameraAngleX += (y - mouseY) * 0.1;
        mouseX = x;
        mouseY = y;
    }
    if (mouseRightDown){
        cameraDistanceX = (x - mouseX) * 0.002;
        cameraDistanceY = -(y - mouseY) * 0.002;
        mouseY = y;
        mouseX = x;
```

```
    }
    glutPostRedisplay();
}
```

（六）使用 OBV 实现高速渲染

实现描述：

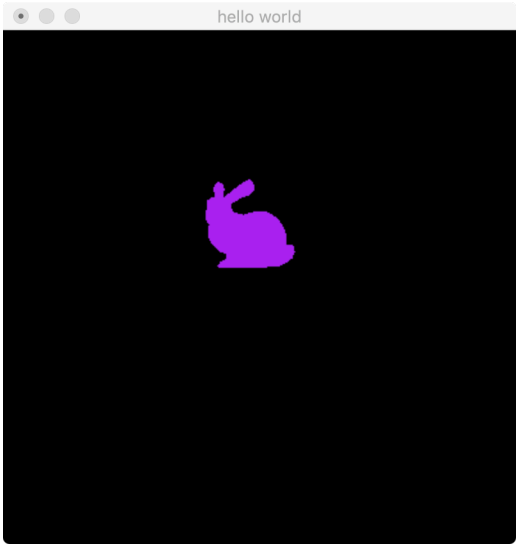
建立显示列表、调用显示列表、删除显示列表。

分配显示列表编号：OpenGL 中用正整数来区分不同的显示列表，为防止重复定义已经存在的显示列表号，使用 glGenLists 函数来自动分配一个没有被使用过的显示列表编号。

创建显示列表：声明了把哪些 OpenGL 语句装入到当前显示列表中。使用 glNewList 开始装入，使用 glEndList 结束装入。

调用显示列表：只需要在需要调用的地方插入 glCallList(id)即可，入参 id 表示了要调用的显示列表的编号。另外也可以使用 glCallLists 一次性调用一组显示列表。

测试截图：　（主函数设置 flag=3 启动）



具体代码：

```
void CreateDisplayLists(){
    glNewList(1,GL_COMPILE);
    for (int i=0;i<farr.size();i++){
        glBegin(GL_TRIANGLES);
        glVertex3f(varr[farr[i][0]-1][0], varr[farr[i][0]-1][1],
varr[farr[i][0]-1][2]);
        glVertex3f(varr[farr[i][1]-1][0], varr[farr[i][1]-1][1],
varr[farr[i][1]-1][2]);
        glVertex3f(varr[farr[i][2]-1][0], varr[farr[i][2]-1][1],
varr[farr[i][2]-1][2]);
        glEnd();
        }
        glEndList();
}
```