练 习 题 报 告

课程名称 计算机图形学

项目名称 OFF 格式的模型显示

学　　院 计算机与软件学院

专　　业 软件工程（腾班）

指导教师 熊卫丹

报 告 人 洪子敬 学号 2022155033

一、 练习目的

1. 了解和熟悉 OFF 模型文件格式。
2. 掌握读取 OFF 模型文件。
3. 了解基本 3D 图元的绘制。
4. 了解深度测试技术。
5. 了解面剔除技术。

二．练习完成过程及主要代码说明

实验要求：主要是设计并实现读取 OFF 文件的接口函数 void read_off(const std::string filename)。修改本实验提供的 main.cpp 文件，按照下面顺序完成实验：

1. 创建工作项目

在所给代码用 cmd 打开终端，使用 cmake 构建工程，用 VS2022 打开项目。

2. 完成 read_off 函数读取 OFF 文件，并存储信息到外部变量中

代码中定义了一个 vec3i 结构体，以及 4 个向量容器：vertices、faces、points 和 colors 用于存储顶点下标。Vertices 容器是存储所有顶点坐标，faces 容器存储每个面片的顶点下标；而 points 是根据 vertices 和 faces 数据进一步获得的所有面片上顶点的数据，colors 容器则是和 points 的顶点一一对应，保存这个顶点的颜色，这里我们使用 main.cpp 开头定义的一些颜色。

3. 在 read_off()中将读取的顶点数据保存到 vertices 和 faces 中

代码中需要我们补写顶点坐标的读取并存放在 vertices 以及面片下标的读取存放在 faces，这里我们只需要根据读入的 off 文件格式确定读入的方式即可。**不过要注意 vertices 中存放的类型是 glm::vec3 格式，faces 中存放的类型是自定义结构类型 vec3i**。具体代码如下：

```
// 根据顶点数，循环读取每个顶点坐标，将其保存到vertices
float x1, x2, x3;
for (int i = 0; i < nVertices; i++) {
    fin >> x1 >> x2 >> x3;
    vertices.push_back(glm::vec3(x1, x2, x3));
}

// 根据面片数，循环读取每个面片信息，并用构建的vec3i结构体保存到faces
int n, y1, y2, y3;
for (int i = 0; i < nFaces; i++) {
    fin >> n >> y1 >> y2 >> y3;
    vec3i tmp(y1, y2, y3);
    faces.push_back(tmp);
}
```

洪子敬
2022155033

4. 完成 storeFacesPoints 函数，存储用于着色器中顶点和颜色信息

代码要求我们补写 points 和 colors 的填充，在前面我们知道 points 是每个面片上的顶点坐标集合，colors 是对应每个顶点的颜色。此处要注意，**实验中读入缓冲的时候已经设置每次是 3 个 1 面片，所以不需要我们把每个面片上的顶点放在一起以免弄混淆，我们只需要依此添加每个面片的顶点和颜色分别到 points 和 colors 中即可**。即遍历每个面片，把对应的三个下标对应成顶点和颜色加入到 points 和 colors 中，具体代码如下所示：

```
void storeFacesPoints()
{
    points.clear();
    colors.clear();
    // @TODO: Task1:修改此函数在points和colors容器中存储每个三角面片的各个点和颜色信息
    // 在points容器中，依次添加每个面片的顶点，并在colors容器中，添加该点的颜色信息
    // 比如一个正方形由两个三角形构成，那么vertices会由4个顶点的数据构成，faces会记录两个三角形的顶点下标，
    // 而points就是记录这2个三角形的顶点，总共6个顶点的数据。
    // colors容器则是和points的顶点一一对应，保存这个顶点的颜色，这里我们可以使用顶点坐标或者自己设定的颜色赋值。

    //存储point和colors
    for (int i = 0; i < nFaces; i++) {
        vec3i index = faces[i];
        points.push_back(vertices[index.a]);
        points.push_back(vertices[index.b]);
        points.push_back(vertices[index.c]);
        colors.push_back(vertex_colors[index.a]);
        colors.push_back(vertex_colors[index.b]);
        colors.push_back(vertex_colors[index.c]);
    }
}
```

洪子敬
2022155033

5. 在 init()和 key_callback ()中启用被注释的代码

在修改完上述代码后我们再启动下面被注释的代码，否则会报错。

```
// @TODO: Task1:修改完成后再打开下面注释，否则程序会报错
// 分别读取数据
 glBufferSubData(GL_ARRAY_BUFFER, 0, points.size() * sizeof(glm::vec3), &points[0]);
 glBufferSubData(GL_ARRAY_BUFFER, points.size() * sizeof(glm::vec3), colors.size() * sizeof(glm::vec3), &colors[0]);
```

HZJ

6. 参考前面提到函数控制深度测试、面剔除和绘制模式的切换

（1）在 display 函数中：需要清理窗口，包括颜色缓存和深度缓存

```
void display(void)
{
    // @TODO: Task2:清理窗口，包括颜色缓存和深度缓存
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

    glDrawArrays(GL_TRIANGLES, 0, points.size());
}
```

HZJ

（2）在 key_callback 函数中：需要对各分支的模式进行开启关闭，分别通过以下语句实现：

启动深度测试：$glEnable(GL\_DEPTH\_TEST)$;

关闭深度测试：$glDisable(GL\_DEPTH\_TEST)$;

启动反面剔除：$glEnable(GL\_CULL\_FACE)$; $glCullFace(GL\_BACK)$;

关闭反面剔除：$glDisable(GL\_CULL\_FACE)$;

启动正面剔除：$glEnable(GL\_CULL\_FACE)$; $glCullFace(GL\_FRONT)$;

关闭正面剔除：$glDisable(GL\_CULL\_FACE)$;

启动线绘制模式：$glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE)$;

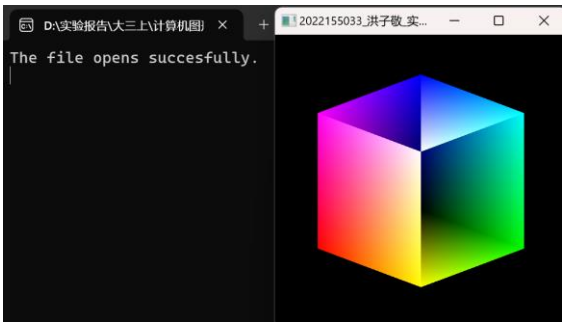
关闭线绘制模式：$glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL)$;

由于代码很多都重复，这里只展示一小部分：

```
// @TODO: Task5:启用线绘制模式
else if (key == GLFW_KEY_4 && action == GLFW_PRESS && mode == 0x0000)
{
    cout << "line mode: enable" << endl;
    //改为线模式
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
}                                                                    HZJ
// @TODO: Task5:关闭线绘制模式
else if (key == GLFW_KEY_4 && action == GLFW_PRESS && mode == GLFW_MOD_SHIFT)
{
    cout << "line mode: disable" << endl;
    //改为填充模式
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
}                                                                    HZJ
```

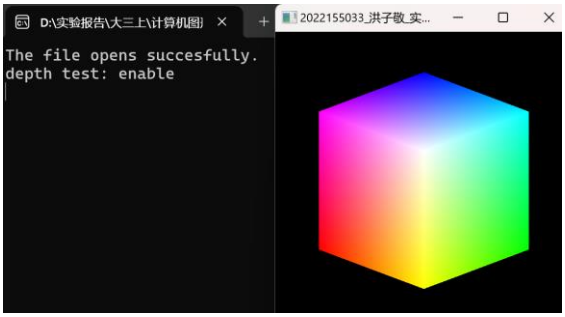
7. 结果测试

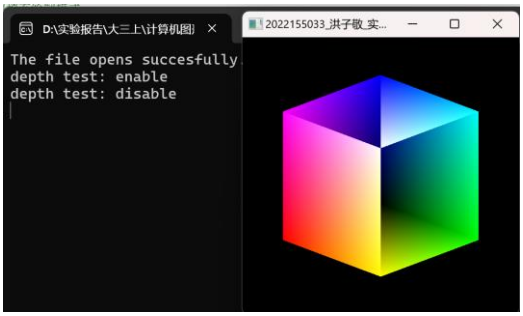
通过前面步骤，我们已经完成了对代码的编写，运行 main.sln，我们可以得到一下效果图：



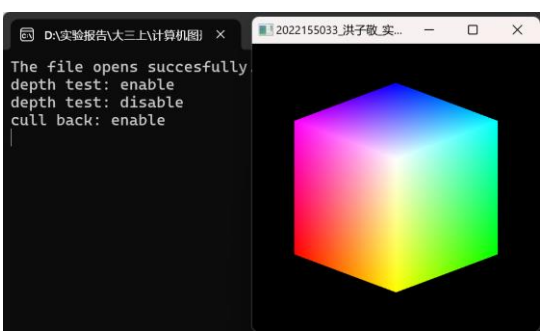
注意进行下面测试前：先按“shift”键，后按其他模式的启动键才能启动当前模式；关闭当前模式需要同时按“shift”键和该模型的启动键。

测试（1）：按键 1 和实现按键 1 和!分别控制深度测试的开启和关闭

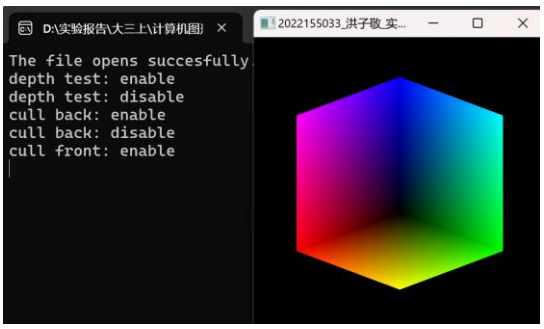




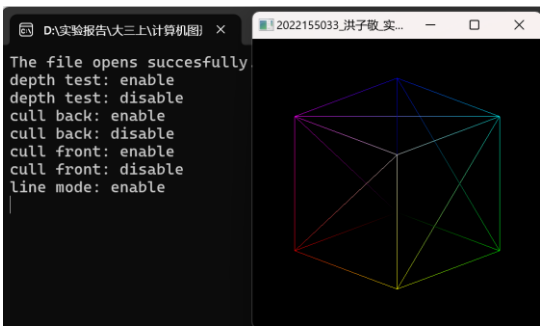
测试（2）：按键 2 和@分别控制背面面片的剔除和恢复

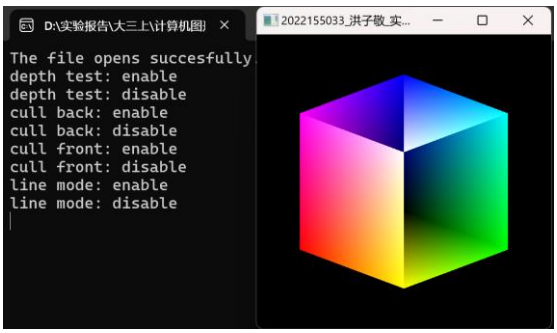


测试（3）：按键 3 和#分别控制正向面片的剔除和恢复

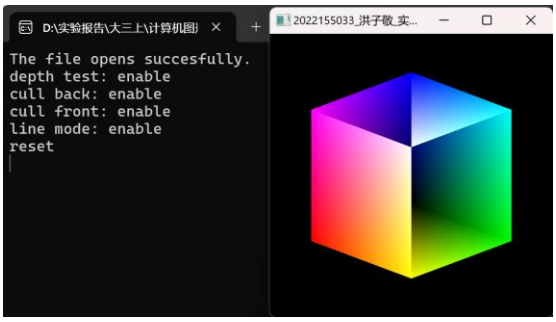


测试（4）：按键 4 和$分别控制线模式的启动和关闭





测试（5）：按键 0 实现对图形模式的重置



观察结果测试可知，代码编写和修改成功，本次练习圆满结束。