

操作系统原理实验报告

计算机图形学 HW2

院 (系) 名 称: 数据科学与计算机学院

专 业 名 称: 计算机科学与技术

学 生 姓 名: 王永锋

学 生 学 号: 16337237

指 导 教 师: 苏卓

二〇一八年四月三十日

目 录

1	配置开发环境	1
1.1	开发环境	1
1.2	依赖第三方库	1
2	图 1 效果实现	3
2.1	相关文件说明	3
2.2	部分设计细节	3
2.2.1	细节 1	3
2.3	运行截图	4
3	图 2 效果实现	5
3.1	相关文件说明	5
3.2	部分设计细节	5
3.2.1	坐标转换	5
3.2.2	正方体与轮廓线同时显示	6
3.3	运行截图	6
4	作业感想	8

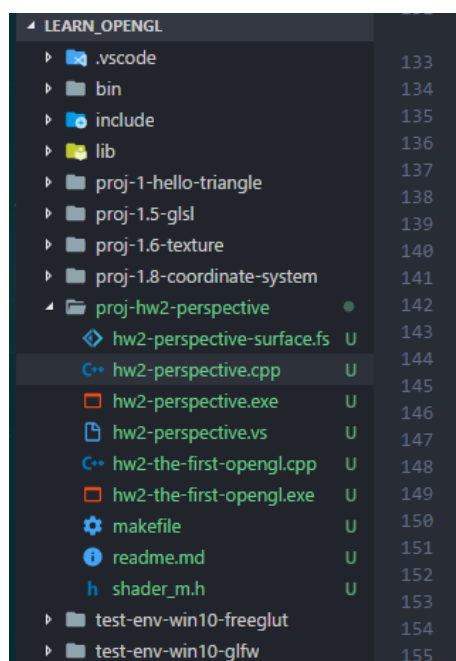
1 配置开发环境

1.1 开发环境

本作业在 Window10 下使用基于 MinGW 的 gcc 编译器编译运行。

1.2 依赖第三方库

参考自教程 [1] 的环境，在第三方的 OpenGL 窗口管理程序库中，我选择了 glfw 并配置在我的项目文件夹中。除此之外，为了得到 OpenGL 程序中函数的具体位置，我还使用了 GLAD 来进行函数的定位。在实现图 2 的过程中，我还用到了 glm 用于计算旋转、平移、投影矩阵。



LEARN_OPENGL	
.vscode	133
bin	134
include	135
lib	136
proj-1-hello-triangle	137
proj-1.5-glsl	138
proj-1.6-texture	139
proj-1.8-coordinate-system	140
proj-hw2-perspective	141
hw2-perspective-surface.fs	142
hw2-perspective.cpp	143
hw2-perspective.exe	144
hw2-perspective.vs	145
hw2-the-first-opengl.cpp	146
hw2-the-first-opengl.exe	147
makefile	148
readme.md	149
shader_m.h	150
test-env-win10-freeglut	151
test-env-win10-glfw	152

图 1.1 开发环境图示

2 图 1 效果实现

2.1 相关文件说明

实现效果的源代码文件为“hw2-the-first-opengl.cpp”，可执行文件为“hw2-the-first-opengl.exe”。由于通过静态链接库的方式进行编译，因此可直接运行。

2.2 部分设计细节

在实现这一个矩形的过程中，我参考了一部分来自教程 [1] 的代码。

代码主要分为这几个部分：

- 使用 glfw 创建窗口
- 使用 glad 导入 opengl 函数
- 编译着色器（着色器的编写）
- 绑定顶点缓冲对象，顶点数组对象
- 渲染循环

2.2.1 细节 1

需要注意的一个细节是，OpenGL 中的绘制对象只有三种：点，线，三角形。那要如何绘制矩形呢？教程 [1] 中给出了一个解决方法：使用两个三角形来拼出一个矩形。¹。

因此在初始化顶点数组的时候，便初始化了两个三角形。

```
1  float vertices_white[] = {  
2      // 第一个三角形  
3      0.5f, 0.5f, 0.0f, // 右上角  
4      0.5f, -0.5f, 0.0f, // 右下角  
5      -0.5f, 0.5f, 0.0f, // 左上角  
6      // 第二个三角形  
7      0.5f, -0.5f, 0.0f, // 右下角  
8      -0.5f, -0.5f, 0.0f, // 左下角  
9      -0.5f, 0.5f, 0.0f // 左上角
```

¹事实上，后来上网一搜，发现无论是多复杂的图形，都需要使用三角形来拼出来

2.3 运行截图

运行 **hw2-the-first-opengl.exe**，结果如图 2.1 所示。

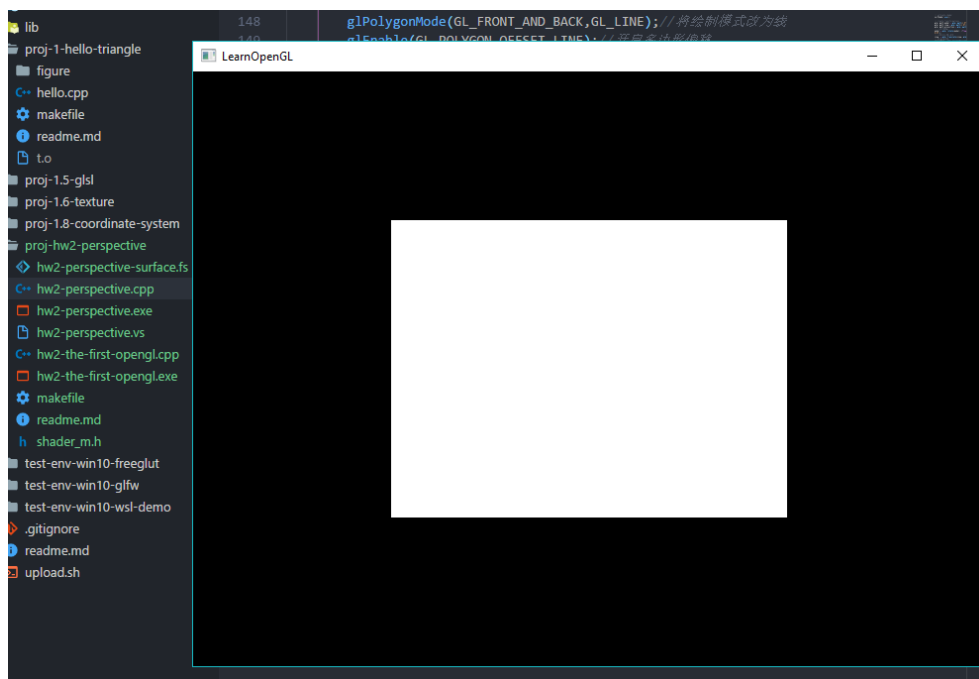


图 2.1 运行截图 1

3 图 2 效果实现

3.1 相关文件说明

实现图 2 过程中，使用了以下文件。其中可执行文件 **hw2-perspective.exe** 直接打开运行即可。

```
1      shader_m.h           // 用于读取着色器
2      hw2-perspective.cpp   // 主要源代码
3      hw2-perspective.vs    // 顶点着色器
4      hw2-perspective-surface.fs // 片段着色器
5      makefile              // 用于编译程序
6      hw2-perspective.exe   // 可执行文件
```

3.2 部分设计细节

在实现图 2 的过程中，我主要遇到了一下问题：

- 坐标转换的实现
- 在显示正方体的同时显示边缘

3.2.1 坐标转换

要将一个正方体显示在屏幕上，一个关键的步骤就是将这个物体的各个顶点转变为在屏幕上的一个个像素，这其中，就需要用到坐标变换。

在我的程序中，坐标变换的实现。

```
1      glm::mat4 model = glm::mat4(1.0f);
2      glm::mat4 view = glm::mat4(1.0f);
3      glm::mat4 projection = glm::mat4(1.0f);
4      // 旋转矩阵，实现随时间旋转
5      model = glm::rotate(model, (float)glfwGetTime(), glm::vec3(-0.5f, 0.5f, 0.0f));
6      // 将物体向前平移三个单位，方便观察
7      view = glm::translate(view, glm::vec3(0.0f, 0.0f, -3.0f));
8      // 设置投影，角度为45度
9      projection = glm::perspective(glm::radians(45.0f), (float)SCR_WIDTH / (float)
      SCR_HEIGHT, 0.1f, 100.0f);
```

这些变换矩阵，最终在顶点着色器中起作用，如以下代码第 14 行所示。

```
1  #version 330 core
2  layout (location = 0) in vec3 aPos;
3
4  out vec4 outColor;
5
6  uniform mat4 model;
7  uniform mat4 view;
8  uniform mat4 projection;
9  uniform vec4 myColor;
10
11 void main()
12 {
13     # 计算得到顶点变换后的坐标
14     gl_Position = projection * view * model * vec4(aPos, 1.0f);
15     outColor = myColor;
16 }
```

3.2.2 正方体与轮廓线同时显示

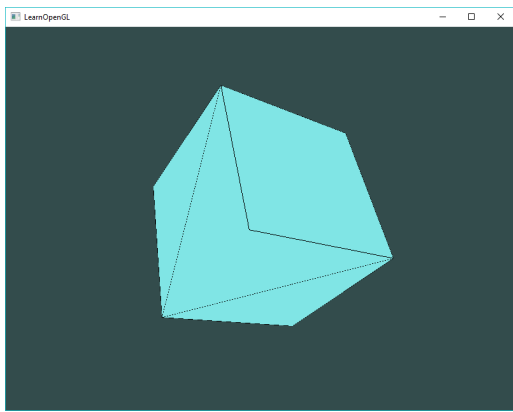
看到老师的实例图中，正方体不仅着了色，还有清晰的轮廓线。在我的实现中，我并不清楚如何将轮廓线画出来，于是便开始各种尝试。

我曾经试过通过修改着色器，让着色器对边缘着上其他颜色，结果却变成了一个具有渐变颜色的正方体，这应该是由于着色器本身就具有片段插值的功能，我暂时还不会干预着色器的这种行为，于是便寻找其他方法。

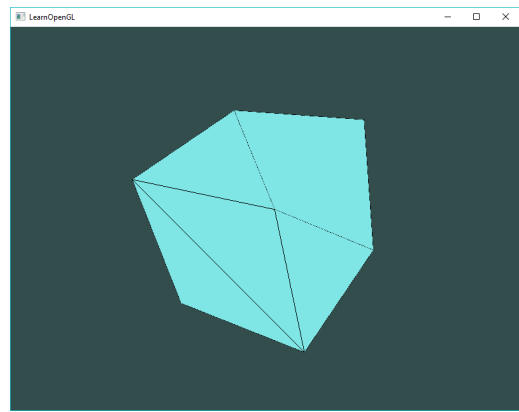
在博客 [2] 中，热心网友提出了一种方法，通过修改全局配置的方法，将绘制面改为绘制轮廓线。这样子只需要在绘制完所有图像后，修改状态位绘制轮廓线，再绘制一次图像即可。参考他的视线，我在渲染循环中增加了绘制轮廓线的过程。效果其实并不好，在正方体旋转的过程中，会出现一闪一闪的情况，而且还一定会画出正方形中间的斜线（那些我不想画出来的轮廓线）。效果如 3.1a 所示。

其实还有一种办法，就是再创建多个对象，然后使用绘制直线的命令将轮廓线画出来，再启动深度检测，可能也行，不过这种由于实现比较繁琐，就没有继续实现下去。

3.3 运行截图



(a) 运行截图 2-1



(b) 运行截图 2-2

图 3.1 运行截图 2

4 作业感想

以前由于 python 课程项目要做 3D 建模的缘故，接触过一些关于 OpenGL 的知识，不过那时候看的是关于固定渲染管线的实现，对于目前的现代 OpenGL 方法而言并不熟悉。

这一次看的 OpenGL 教程，一上来就讲固定渲染管线隐藏了太多细节，让我学现代 OpenGL，使用 OpenGL 的核心模式编程，难度曲线陡升。毕竟，连渲染一条简单的直线，我都必须要自己编写着色器，去给我的顶点着色，更别提渲染更复杂的带有纹理的立体图形了。

这一个作业中的代码，其实有挺多都有参考教程的实例代码（到现在想想，我自己可能还不能够不看示例代码就能够完整的写出这些程序），可以说这个教程对我的学习起到了莫大的帮助，现在我至少是能够看懂教程提供的示例代码，并且通过修改示例代码的方式实现自己想要的功能。

不过，这一次作业花的时间太长了，从开始看教程到完成作业，花了将近一天时间，才把 OpenGL 的核心模式看懂一部分，理解了 OpenGL 的运行逻辑。为了保证开发效率，不知道要不要转投 python 的怀抱诶。

参考文献

- [1] OpenGL 教程: <https://learnopengl-cn.github.io/>
- [2] CSDN 博客: OpenGL: 三维模型, 模型网格点与面片同时显示, 模型网格轮廓线与面片同时显示。 <https://blog.csdn.net/hw140701/article/details/78913660>