# C类大作业实验报告

冯志远 2024311588

## C.1 实验

### 实验内容

1. 显示旋转的三角形（参考第三讲课件）（OpenGL 5 分）
    - **a.** 三角形的颜色通过右键菜单选择绘制
    - **b.** 三角形顺时针旋转

### 实验环境

- **编程语言**：C++
- **开发环境**：Visual Studio Code
- **依赖库**：GLFW、GLAD
- **运行平台**：Windows

### 实验步骤

#### 1. 环境配置

- 配置 OpenGL 环境，安装并链接 GLFW 和 GLAD 库。
- 创建一个宽 800 像素、高 600 像素的窗口用于显示旋转三角形。

#### 2. 着色器设计

- **顶点着色器**：通过矩阵变换实现三角形绕 Z 轴的旋转，使用一个 `uniform` 变量传递实时更新的旋转角度。
- **片段着色器**：通过 `uniform` 变量传递颜色，实现三角形颜色动态更新。

#### 3. 实现三角形绘制

- 定义三角形的顶点坐标并绑定到顶点缓冲对象 (VBO)。
- 创建顶点数组对象 (VAO) 用于管理三角形的绘制数据。

#### 4. 实现旋转与颜色控制

- 在主循环中逐帧更新旋转角度，使三角形持续顺时针旋转。
- 使用 GLFW 捕获键盘事件，通过按键 `1` 、 `2` 、 `3` 切换三角形的颜色为红色、绿色、蓝色。

**5. 循环渲染**

- 清空屏幕并设置背景色。
- 绘制更新后的三角形，实时应用旋转角度和颜色变化。

**6. 资源清理**

- 程序退出前，释放 VAO、VBO 和着色器相关资源，销毁窗口并退出 GLFW。

## 实验结果

- 显示一个持续顺时针旋转的三角形。
- 按下 `1`、`2`、`3` 键，三角形颜色分别变为红色、绿色、蓝色。
- 程序运行稳定，窗口关闭后资源被正确释放。

## 实验总结

通过本实验，熟悉了 OpenGL 的基本使用方法，包括顶点缓冲区、着色器编写以及动态交互功能的实现。成功完成了一个基础的动态渲染案例，为进一步学习复杂图形渲染奠定了基础。同时，通过调试与优化代码，提升了对图形学数学模型的理解。

# C.2 实验

## 实验内容

绘制三角形和四边形并着色，同时旋转和平移（OpenGL 5 分）

- **a.** 三角形的三边长不同；
- **b.** 三角形的三个顶点的颜色不同；
- **c.** 三角形顺时针旋转；
- **d.** 四边形长度任意，四个顶点颜色不同，逆时针旋转，要求使用两种着色模式（smooth 和 flat）。

## 实验环境

1. **编程语言**：C++
2. **开发环境**：Visual Studio Code
3. **依赖库**：GLFW、GLAD
4. **运行平台**：Windows
5. **项目结构**：
    - 平面着色（Flat）：`flat` 文件夹
    - 平滑着色（Smooth）：`smooth` 文件夹

- 执行命令：
  - 平面着色： `make run dir=flat`
  - 平滑着色： `make run dir=smooth`

## 实验步骤

### 1. 环境配置

- 安装并配置 GLFW 和 GLAD 库。
- 创建 800x600 的窗口，初始化 OpenGL，并设置视口大小。

### 2. 顶点着色器设计

- **旋转与平移**：
  - 在顶点着色器中，通过矩阵变换实现三角形和四边形的旋转与平移。
  - 每帧更新旋转角度和位移，使三角形顺时针旋转，四边形逆时针旋转，并添加动态平移效果。
- **颜色属性**：
  - 为每个顶点设置颜色数据，传递给片段着色器处理。

### 3. 片段着色器设计

- **平面着色（Flat）**：
  - 使用 `flat` 修饰符传递颜色，确保片段颜色不进行插值，每个图形整体显示一个固定颜色。
- **平滑着色（Smooth）**：
  - 去除 `flat` 修饰符，片段颜色根据顶点颜色进行插值，实现渐变效果。

### 4. 顶点数据绑定

- 定义三角形和四边形的顶点数据，包括位置和颜色。
- 创建两个顶点缓冲对象（VBO）和两个顶点数组对象（VAO），分别绑定三角形和四边形的数据。

### 5. 循环渲染

- 每帧更新旋转角度和位移量：
  - 三角形顺时针旋转，水平向右平移；
  - 四边形逆时针旋转，水平向左平移。
- 切换着色模式并绘制：
  - 使用 `glUseProgram` 切换到平面着色或平滑着色的片段着色器程序；
  - 绘制三角形（ `GL_TRIANGLES` ）和四边形（ `GL_TRIANGLE_FAN` ）。

### 6. 资源清理

- 在程序退出前，释放 VBO、VAO 和着色器资源，并销毁窗口。

## 实验结果

1. **显示效果**：
   - 界面显示一个三边长不同的三角形，三个顶点颜色不同（红、绿、蓝），三角形顺时针旋转并向右平移。
   - 界面显示一个四边形，四个顶点颜色不同（红、绿、蓝、黄），四边形逆时针旋转并向左平移。
2. **着色模式**：
   - 在 `flat` 模式下，三角形和四边形均显示为固定颜色，不进行颜色插值；
   - 在 `smooth` 模式下，三角形和四边形呈现顶点颜色的渐变效果。

## 实验总结

通过本实验，进一步掌握了 OpenGL 中的矩阵变换、着色器编写，以及顶点数据的高效管理方法。通过对比平面着色和平滑着色，加深了对片段着色器中颜色插值的理解。

# C.3 实验

## 实验内容

绘制一个彩色的四棱锥并添加光照效果（OpenGL 5 分）

- **a.** 四棱锥边长均为 2；
- **b.** 四棱锥各个顶点颜色不同；
- **c.** 四棱锥的中心为 (1, 2, 3)；
- **d.** 使用多种光（环境光、镜面光和散射光等），通过调整材质因子和光的因子实现不同的光照效果。

## 实验环境

1. **编程语言**：C++
2. **开发环境**：Visual Studio Code
3. **依赖库**：GLFW、GLAD、GLM
4. **运行平台**：Windows

## 实验步骤

### 1. 环境配置

- 配置 GLFW 和 GLAD 库，设置 OpenGL 环境为 3.3 Core Profile。

- 创建一个宽 800 像素、高 600 像素的窗口，初始化视口和深度测试功能。

**2. 着色器设计**

- **顶点着色器**：
  - 通过 `model`、`view` 和 `projection` 矩阵实现三维变换。
  - 使用法线变换矩阵计算顶点法向量，用于光照计算。
  - 将顶点颜色、法向量和片段位置传递给片段着色器。
- **片段着色器**：
  - 实现环境光、散射光和镜面光的计算。
  - 使用 `uniform` 变量设置光源位置、颜色以及材质参数（反射因子、光强度等）。

**3. 四棱锥数据定义**

- 顶点数据：定义四棱锥的底部四个顶点及其法向量，每个顶点具有独立的颜色。
- 顶点索引：使用 `EBO` 定义底部平面和四个侧面，每个侧面包含一个公共顶点和两条边。
- 法向量：为每个面单独计算法向量，并通过顶点数据传递给着色器。

**4. 渲染管线配置**

- 生成 VAO、VBO 和 EBO，绑定四棱锥的顶点和索引数据。
- 设置顶点属性指针：位置、颜色和法向量，分别绑定到着色器对应的输入变量。

**5. 光照和材质设置**

- 默认光照参数：
  - 环境光强度：0.3
  - 散射光强度：0.5
  - 镜面光强度：0.5
  - 镜面高光（shininess）：32
  - 光源位置：`(1.0, 6.0, 3.0)`
  - 光源颜色：白色 `(1.0, 1.0, 1.0)`
- 用户交互：在程序运行前通过控制台输入参数，实时调整光源强度和材质反射因子。

**6. 主循环渲染**

- 计算 `model` 矩阵，将四棱锥的中心平移到 `(1.0, 2.0, 3.0)`，同时保持旋转和缩放功能。
- 设置 `view` 矩阵，从视点 `(3.0, 6.0, 10.0)` 看向四棱锥中心 `(1.0, 2.0, 3.0)`。
- 设置 `projection` 矩阵，采用透视投影以实现三维视觉效果。
- 每帧清空颜色缓冲区和深度缓冲区，调用 `glDrawElements` 绘制四棱锥。

**7. 资源清理**

- 程序退出前，释放 VAO、VBO、EBO 和着色器程序，销毁 GLFW 窗口并退出。

## 实验结果

1. **显示效果**：
   - 屏幕显示一个边长为 2、顶点颜色各异的四棱锥，底部和四个侧面颜色明亮且独立。
   - 环境光提供基础亮度，散射光根据光源位置和表面法向量调整亮度，镜面光增强高光效果。
   - 用户可以通过调整光源参数（如环境光强度、散射光强度、镜面光强度和光源颜色）观察不同的光照效果。
2. **动态交互**：
   - 用户在运行时输入光源颜色、强度和材质参数，实时更新显示效果。

## 实验分析

### 成功点

1. 成功实现四棱锥的几何绘制和光照效果，三种光照模型（环境光、散射光和镜面光）结合表现出立体感和材质效果。
2. 通过顶点法向量的计算和传递，保证了光照效果在侧面和底部的正确性。
3. 用户交互功能增强了光照模型的可调性，为不同场景设置提供了灵活性。

### 问题与解决

1. **问题**：法向量变换不正确，导致光照效果异常。
   - **解决**：使用 `mat3(transpose(inverse(model)))` 计算法向量变换矩阵，确保其符合模型变换。
2. **问题**：底部平面亮度过高，无法清晰区分光照效果。
   - **解决**：通过调整环境光强度和散射光强度，使底面与侧面光照效果均衡。

## 实验总结

本实验通过构建光照模型，深入理解了环境光、散射光和镜面光的原理及其在 OpenGL 中的实现。使用法向量变换和材质反射因子的设置，成功展示了三维图形的立体感和真实感。

## 展望

下一步可以加入更多高级光照效果，如点光源、多光源或动态阴影，以进一步提高图形的真实感。此外，可以结合纹理映射技术为四棱锥添加表面细节，使其更加逼真。