#### 实验环境

* **操作系统**：Windows 10
* **开发工具**：Python 3.10
* **OpenGL库**：PyOpenGL, GLFW
* **显卡**：支持 OpenGL 3.3 或更高版本的显卡
* **其他库**：NumPy

#### 实验内容

1. 使用 PyOpenGL 和 GLFW 创建 OpenGL 窗口。
2. 编写顶点着色器和片段着色器，渲染一个简单的三角形。
3. 掌握着色器的编写方式，调试 OpenGL 绘制流程。

#### 实验原理

OpenGL 是一个强大的图形渲染库，用于生成 2D 和 3D 图形。在 OpenGL 中，图形的渲染是通过 GPU 来完成的。为了将图形数据传递给 GPU，OpenGL 采用了着色器（Shaders）来处理顶点、颜色等数据。以下是绘制一个三角形的基本步骤：

1. **顶点数据（Vertex Data）**：我们需要给定三角形的三个顶点坐标。
2. **着色器（Shaders）**：用来计算和渲染图形的程序。每个着色器有特定的任务：
   * 顶点着色器：负责计算和传递顶点位置。
   * 片段着色器：负责处理颜色和像素。
3. **绘制命令（Draw Calls）**：通过 glDrawArrays 或 glDrawElements 命令将顶点数据传递给 GPU 进行绘制。

#### 实验步骤

**初始化和创建窗口**： 使用 GLFW 库创建一个 OpenGL 窗口，并进行初始化，设置 OpenGL 上下文。

**编写着色器**： 使用 GLSL 语言编写顶点着色器和片段着色器，并通过 PyOpenGL 的 compileShader 和 compileProgram 函数进行编译和链接。

**设置顶点数据**： 定义一个三角形的三个顶点坐标，并将其传递到 GPU。使用 glGenVertexArrays 和 glGenBuffers 创建 VAO（顶点数组对象）和 VBO（顶点缓冲对象）

**渲染循环**： 在渲染循环中，通过 glUseProgram 激活着色器程序，使用 glDrawArrays 绘制三角形。每一帧都会清除屏幕，并重新绘制三角形。

**资源清理**： 程序结束时，释放所有分配的 OpenGL 资源。

#### 代码实现

import glfwfrom OpenGL.GL

import \*from OpenGL.GL.shaders

import compileProgram, compileShader

import numpy as np

# 顶点着色器代码

vertex\_shader = """

# version 330 core

layout(location = 0) in vec3 aPos;

void main()

{

gl\_Position = vec4(aPos, 1.0);

}

"""

# 片段着色器代码

fragment\_shader = """

# version 330 core

out vec4 FragColor;

void main()

{

FragColor = vec4(1.0, 0.0, 0.0, 1.0); // 红色

}

"""

# 初始化GLFWif not glfw.init():

raise Exception("GLFW 初始化失败")

# 创建窗口

window = glfw.create\_window(800, 600, "OpenGL 基础图形", None, None)if not window:

glfw.terminate()

raise Exception("创建窗口失败")

glfw.make\_context\_current(window)

# 编译着色器程序

shader = compileProgram(

compileShader(vertex\_shader, GL\_VERTEX\_SHADER),

compileShader(fragment\_shader, GL\_FRAGMENT\_SHADER)

)

# 定义三角形的顶点

vertices = np.array([

0.0, 0.5, 0.0, # 顶点1

-0.5, -0.5, 0.0, # 顶点2

0.5, -0.5, 0.0 # 顶点3

], dtype=np.float32)

# 创建VBO和VAO

VAO = glGenVertexArrays(1)

VBO = glGenBuffers(1)

glBindVertexArray(VAO)

glBindBuffer(GL\_ARRAY\_BUFFER, VBO)

glBufferData(GL\_ARRAY\_BUFFER, vertices.nbytes, vertices, GL\_STATIC\_DRAW)

glVertexAttribPointer(0, 3, GL\_FLOAT, GL\_FALSE, 3 \* vertices.itemsize, None)

glEnableVertexAttribArray(0)

glBindBuffer(GL\_ARRAY\_BUFFER, 0)

glBindVertexArray(0)

# 渲染循环while not glfw.window\_should\_close(window):

# 处理输入

glfw.poll\_events()

# 清屏

glClearColor(0.2, 0.3, 0.3, 1.0)

glClear(GL\_COLOR\_BUFFER\_BIT)

# 使用着色器程序

glUseProgram(shader)

# 绘制三角形

glBindVertexArray(VAO)

glDrawArrays(GL\_TRIANGLES, 0, 3)

glBindVertexArray(0)

# 交换缓冲区

glfw.swap\_buffers(window)

# 清理资源

glDeleteVertexArrays(1, [VAO])

glDeleteBuffers(1, [VBO])

# 终止GLFW

glfw.terminate()

#### 实验结果

程序运行时，会弹出一个 OpenGL 窗口，渲染出一个红色的三角形。

