# 计算机图形学 Homework2

## Basic

### 实现思路

1. 使用OpenGL3.3+GLFW画一个简单的三角形
    1. 初始化glfw，设置版本号和核心模式

```
glfwInit();
glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
```

    2. 创建窗口，并使用glad管理窗口指针

```
GLFWwindow* window = glfwCreateWindow(500, 400, "Hello GLFW", NULL, NULL);
if (window  ==  NULL)       // 错误处理
{
    std::cout << "error" << std::endl;
    glfwTerminate();
    return -1;
}
glfwMakeContextCurrent(window);      // 设置window当前内容
glfwSetFramebufferSizeCallback(window, framebuffer_size_callback);          //
注册回调函数
glfwSwapInterval(1);

if (!gladLoadGLLoader((GLADloadproc)glfwGetProcAddress))
{...}
```

    3. VBO, VAO对象处理
        - **顶点输入**：顶点满足标准化设备坐标，即(0,0)处于图像的中心（而不是左上角）且3D坐标在3个轴（x、y和z）上都为-1.0到1.0的范围内；注意因为这里是一个2D三角形，设置深度为0；

```
float vertices[] = {
    -0.5f, -0.5f, 0.0f,     //右下
     0.5f, -0.5f, 0.0f,     //左下
     0.0f,  0.5f, 0.0f      //顶部
};
```

        - 初始化VBO, VAO对象并绑定缓冲

```
unsigned int VBO, VAO;
// 绑定缓冲到VAO的缓冲类型上
glGenVertexArrays(1, &VAO);
glBindVertexArray(VAO);

// 绑定缓冲到VBO的缓冲类型上
glGenBuffers(1, &VBO);
glBindBuffer(GL_ARRAY_BUFFER, VBO);

// 将定义的顶点数据复制到当前绑定的缓冲内存中
glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
GL_STATIC_DRAW);
```

- 链接并启用顶点属性指针

```
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float),
(void*)0);
glEnableVertexAttribArray(0);

glBindBuffer(GL_ARRAY_BUFFER, 0);
glBindVertexArray(0);   // 解绑？
```

4. 着色程序的编译和链接

- 定义全局字符串对象，内容为GLSL语言编写的着色器源代码

```
// 顶点着色器 code
const char *vertexShaderSource = "#version 330 core\n"
"layout(location = 0) in vec3 aPos;\n"
"void main()\n"
"{\n"
"gl_Position = vec4(aPos.x, aPos.y, aPos.z, 1.0);\n"
"}\n";

// 片段着色器 code
const char *fragmentShaderSource = "#version 330 core\n"
"out vec4 FragColor;\n"
"void main()\n"
"{\n"
"   FragColor = vec4(0.1f, 1.0f, 1.0f, 1.0f);\n"
"}\n\0";

```

- 编译顶点着色器

```
//创建着色器对象，类型为顶点
unsigned int vertexShader = glCreateShader(GL_VERTEX_SHADER);
// 将顶点着色器源码附加到着色器对象上
glShaderSource(vertexShader, 1, &vertexShaderSource, NULL);
// 编译
glCompileShader(vertexShader);
```

> 使用*glGetShaderiv* 和*glGetShaderInfoLog*进行编译异常处理

- 编译片段着色器，类似顶点着色器

```
unsigned int fragmentShader = glCreateShader(GL_FRAGMENT_SHADER);
glShaderSource(fragmentShader, 1, &fragmentShaderSource, NULL);
glCompileShader(fragmentShader);
```

> 使用*glGetShaderiv* 和*glGetShaderInfoLog*进行编译异常处理

- 链接多个着色器为程序对象

```
// 创建程序
unsigned int shaderProgram = glCreateProgram();
//将顶点着色器attach到程序上
glAttachShader(shaderProgram, vertexShader);
//将片段着色器attach到程序上
glAttachShader(shaderProgram, fragmentShader);
//链接程序对象
glLinkProgram(shaderProgram);
```

> 使用*glGetProgramiv* 和*glGetProgramInfoLog*进行编译异常处理

5. 渲染循环

```
    while (!glfwWindowShouldClose(window))
    {
        //输入
        processImput(window);

        //渲染指令
        glClearColor(0.2f, 0.3f, 0.3f, 1.0f);
        glClear(GL_COLOR_BUFFER_BIT);

        glUseProgram(shaderProgram);       // 激活程序对象
        glBindVertexArray(VAO);            // 绑定顶点数组
        glDrawArrays(GL_TRIANGLES, 0, 3); // 绘制三角形

        glfwSwapBuffers(window);  //交换颜色缓冲（绘制窗口输出）
        glfwPollEvents();         //检查触发事件并调用对应的回调函数
    }
```

6. 关闭glfw，删除着色器对象

```
glfwTerminate();
glDeleteShader(vertexShader);
glDeleteShader(fragmentShader);
```

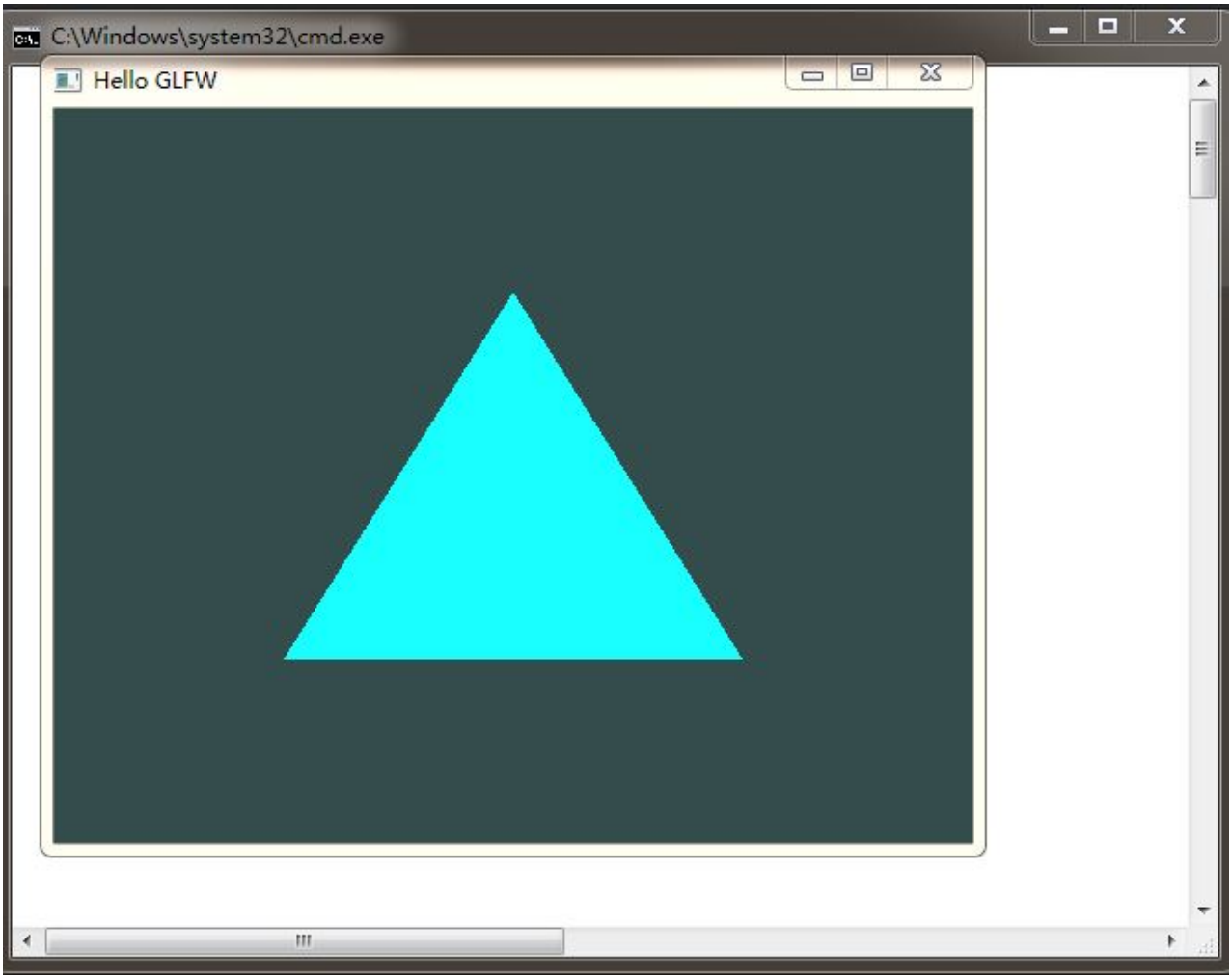
2. 将三角形三个顶点分别改为红绿蓝，并解释为什么会出现这样的结果
   - 这一问的解答中将第一问中**着色程序的编译和链接**和Render Loop中的glUseProgram函数封装在**Shader**类中，使用文件IO来读取着色器code；
   - 解释：三个顶点分别为红绿蓝，渲染三角形时，光栅化阶段通常会比原来的指定顶点更多的片段，并决定这些片段的相对位置。片段着色器运行时会基于相对位置进行颜色的线性插值。
3. 给工程添加一个GUI
   - 下载IMGUI的source和gl3w的source（运行给定的python程序下载），将得到的源文件加入项目，将头文件的地址加入IDE的包含文件路径配置中。
   - 初始化IMGUI

```
    gl3wInit();
    IMGUI_CHECKVERSION();
    ImGui::CreateContext();  // context
    ImGuiIO& io = ImGui::GetIO(); (void)io;

    ImGui::StyleColorsDark();  // style
    ImGui_ImplGlfw_InitForOpenGL(window, true); // platform/randererbinging
    ImGui_ImplOpenGL3_Init(glsl_version);
```

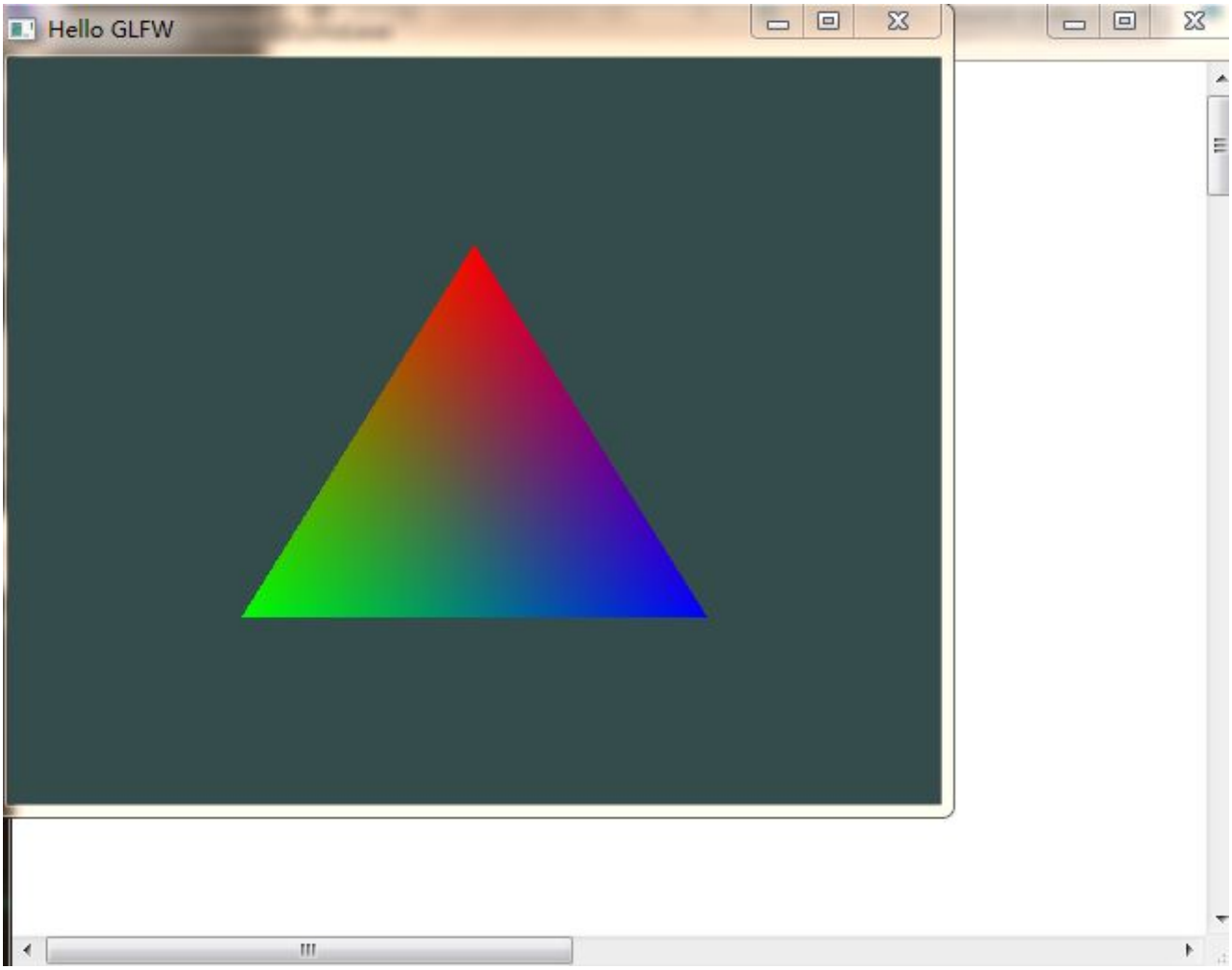
   - 渲染

```
      ImGui::Begin("COLOR");
      ImGui::SetWindowSize(ImVec2(50, 70));
      ImGui::SetWindowPos(ImVec2(-0.5f, 0));

      ImGui::Text("Change the colors");
      ImGui::ColorEdit3("RIGHT", (float*)&colors[0]); // Edit 3 floats representing a color
      ImGui::ColorEdit3("LEFT", (float*)&colors[1]); // Edit 3 floats representing a color
      ImGui::ColorEdit3("TOP", (float*)&colors[2])

      for (int i = 0; i < 3; i++)
      {
         vertices[i * 6 + 3] = colors[i].x;
         vertices[i * 6 + 4] = colors[i].y;
         vertices[i * 6 + 5] = colors[i].z;
       }
        ImGui::End();
```

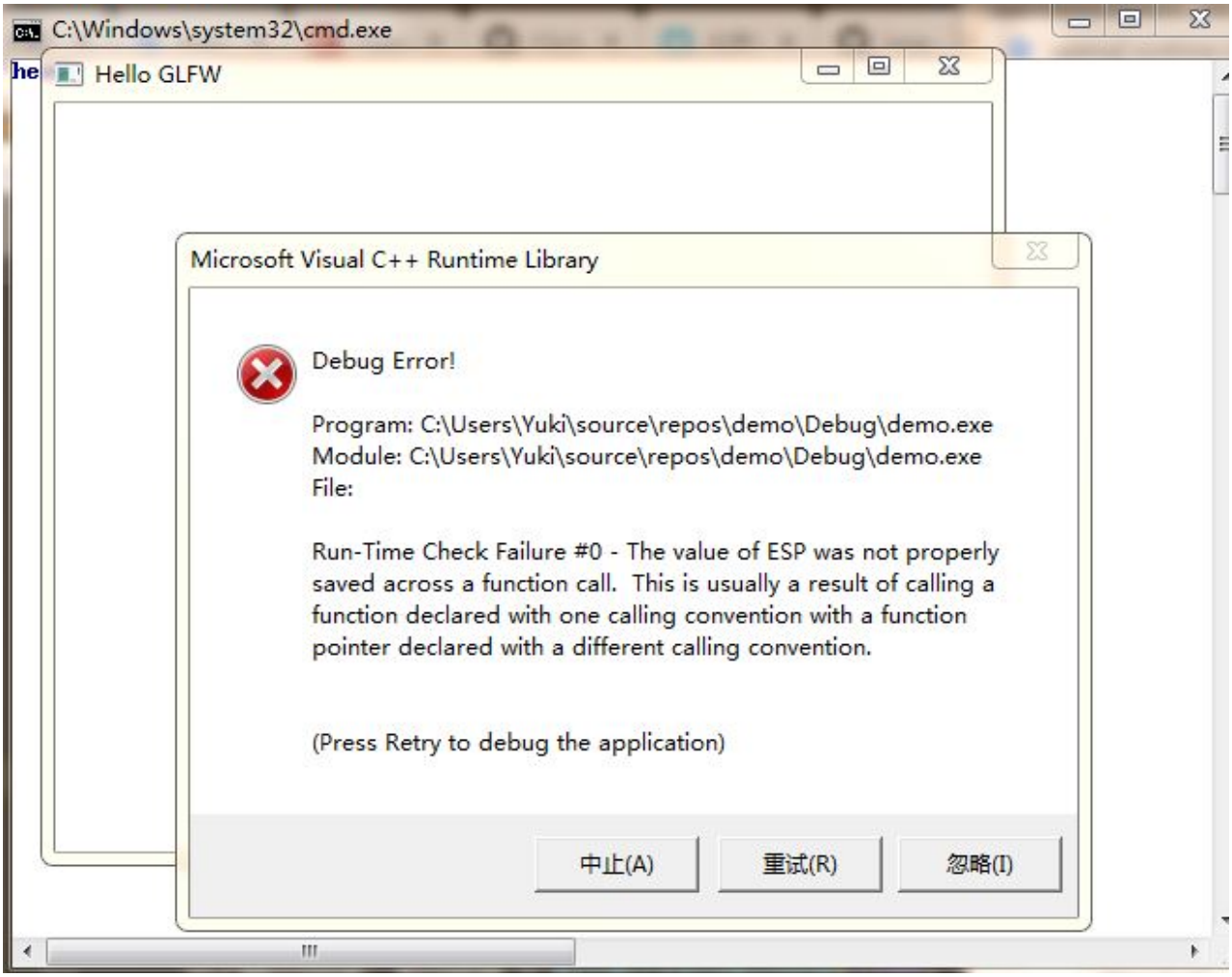
## 运行结果

1. 

2.




3. **ImGui_ImplOpenGL3_NewFrame()** 语句运行时报错，解决方案仍在排查中


C:\Windows\system32\cmd.exe
Hello GLFW
Microsoft Visual C++ Runtime Library
Debug Error!
Program: C:\Users\Yuki\source\repos\demo\Debug\demo.exe
Module: C:\Users\Yuki\source\repos\demo\Debug\demo.exe
File:
Run-Time Check Failure #0 - The value of ESP was not properly saved across a function call. This is usually a result of calling a function declared with one calling convention with a function pointer declared with a different calling convention.
(Press Retry to debug the application)
中止(A)
重试(R)
忽略(I)