# 2.1 HOMEWORK

## 1.彼此相连的三角形

### 说明

添加更多顶点到数据中，使用glDrawArrays，尝试绘制两个彼此相连的三角形

### 解答

#### 1.修改顶点数组，新增三个新的顶点

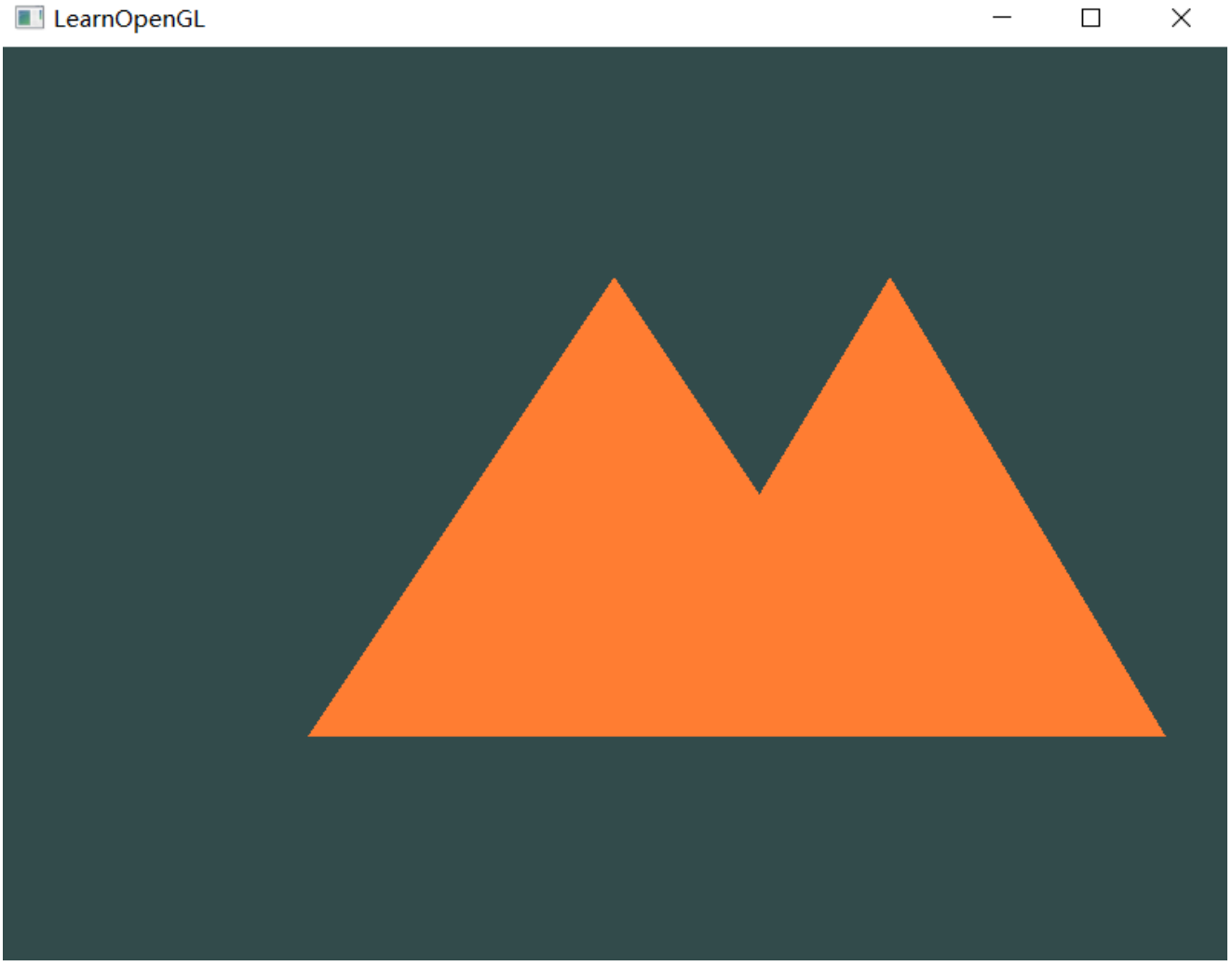
```
float vertices[] = {
    //第一个三角形
    -0.5f, -0.5f, 0.0f, // left
     0.5f, -0.5f, 0.0f, // right
     0.0f,  0.5f, 0.0f,  // top

     //第二个三角形
    0.0f, -0.5f, 0.0f,  // left
    0.9f, -0.5f, 0.0f,  // right
    0.45f, 0.5f, 0.0f   // top
};
```

#### 2.修改渲染循环中的绘制函数的参数

从3改为6个顶点

```
glDrawArrays(GL_TRIANGLES, 0, 6)
```

### 结果




---

# 2.两个三角形使用不同的VAO和VBO

## 说明

建相同的两个三角形，但对它们的数据使用不同的VAO和VBO

## 解答

### 1.分别声明两个三角形顶点

```
float firstTriangle[] = {
    -0.9f, -0.5f, 0.0f,  // left
    -0.0f, -0.5f, 0.0f,  // right
    -0.45f, 0.5f, 0.0f,  // top
};
float secondTriangle[] = {
    0.0f, -0.5f, 0.0f,  // left
    0.9f, -0.5f, 0.0f,  // right
    0.45f, 0.5f, 0.0f   // top
};
```

因为VBO的绑定时以数组为单位的，要分别使用VBO则需要拆成两个数组
如果使用一个数组，则需要在第二个VBO设置偏移量

### 2.分别声明和绑定VAO和VBO

```
unsigned int VBOs[2], VAOs[2];
glGenVertexArrays(2, VAOs); //同时生成两个VAO
glGenBuffers(2, VBOs);
```

```
//先绑定第一个三角形
glBindVertexArray(VAOs[0]);

glBindBuffer(GL_ARRAY_BUFFER, VBOs[0]);
glBufferData(GL_ARRAY_BUFFER, sizeof(firstTriangle), firstTriangle,
GL_STATIC_DRAW);

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

### 3.在渲染循环中分别绘制两个三角形

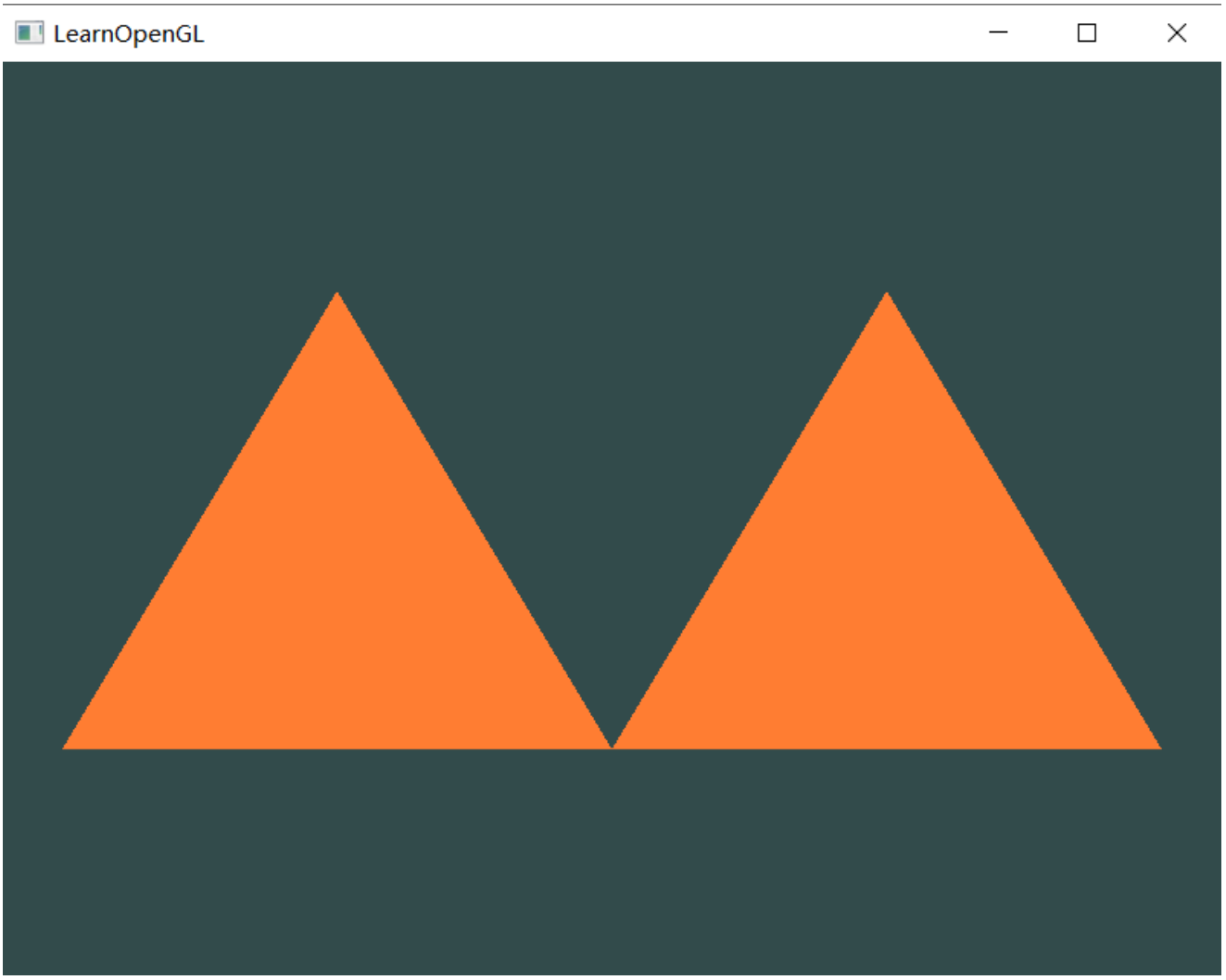
```
//绘制第一个三角形
glBindVertexArray(VAOs[0]);
glDrawArrays(GL_TRIANGLES, 0, 3);
```

### 4.释放VAO数组和VBO数组

```
glDeleteVertexArrays(2, VAOs);
glDeleteBuffers(2, VBOs);
```

释放的是数组，则无需&取址符号

## 结果




---

# 3.创建两个着色器程序，第二个程序使用不同的片段着色器

## 说明

创建两个着色器程序，第二个程序使用一个不同的片段着色器，输出黄色；再次绘制这两个三角形，让其中一个输出为黄色

## 解答

### 1.声明黄色的片段着色器

```
//黄色片段着色器GLSL
const char* fragmentShader2Source = "#version 330 core\n"
"out vec4 FragColor;\n"
"void main()\n"
"{\n"
"   FragColor = vec4(1.0f, 1.0f, 0.0f, 1.0f);\n"
"}\n\0";
```

### 2.分别生成对应的片段着色器

```
//创建黄色片段着色器
unsigned int fragmentShaderYellow = glCreateShader(GL_FRAGMENT_SHADER);
glShaderSource(fragmentShaderYellow, 1,
&fragmentShader2Source, NULL);
glCompileShader(fragmentShaderYellow);
```

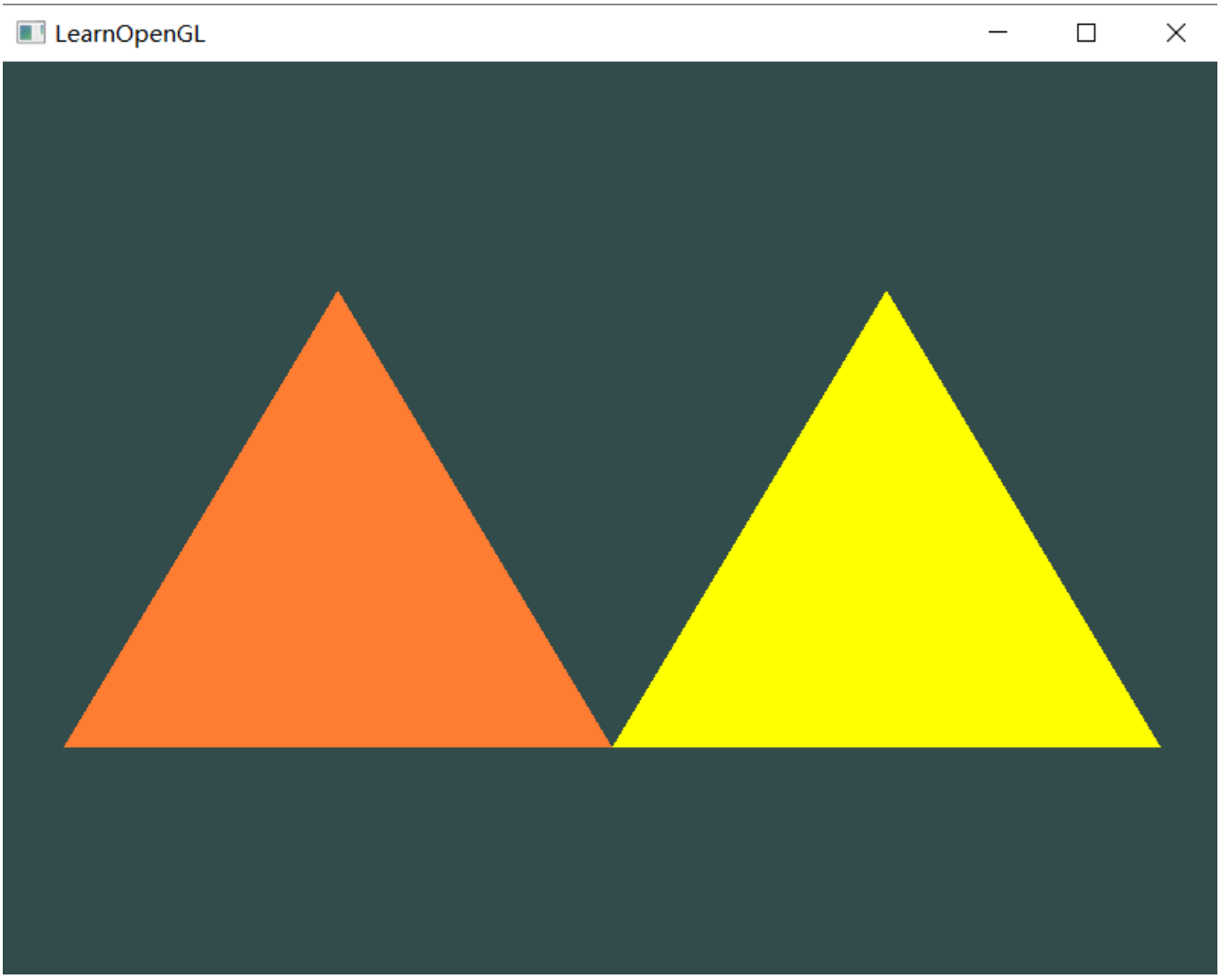
### 3.分别LINK到对应的着色器程序

```
unsigned int shaderProgramYellow = glCreateProgram();
glAttachShader(shaderProgramYellow, vertexShader);
glAttachShader(shaderProgramYellow, fragmentShaderYellow);
glLinkProgram(shaderProgramYellow);
```

### 4.渲染循环中，对不同的三角形使用不同的着色器程序

```
// 用第一个着色器绘制第一个三角形
glUseProgram(shaderProgramOrange);
//绑定和绘制第一个三角形
glBindVertexArray(VAOs[0]);
glDrawArrays(GL_TRIANGLES, 0, 3);

// 用第二个着色器绘制第二个三角形
glUseProgram(shaderProgramYellow);
//绑定和绘制第二个三角形
glBindVertexArray(VAOs[1]);
glDrawArrays(GL_TRIANGLES, 0, 3);
```

## 结果


LearnOpenGL