科目	计算机图形学
学号	16340294
姓名	张星
邮箱	1401046908@qq.com

Basic:

题目一

实验要求

> 使用OpenGL(3.3及以上)+GLFW或freeglut画一个简单的三角形。

实验思路

> 根据TA所给的学习网站和参考资料，我上周已经配好了环境，而且上周已经可以弹出OpenGL的窗口了，所以这周的主要任务是画三角形。首先创建一个数组，用于存储三角形顶点坐标。然后创建顶点缓冲对象和数组对象，这也可以一次向GPU发送多个数据，节省时间。

- *glGenBuffers(1, &VBO)*：使用了一个ID，生成一个VBO对象。
- *glBindBuffer(GL_ARRAY_BUFFER, VBO)*：把新创建的VBO绑定到GL_ARRAY_BUFFER目标上，之后任何GL_ARRAY_BUFFER调用都会配置当前的VBO对象。
- *glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW)*：专门用来把用户定义的数据复制到当前绑定缓冲。

```
float vertices[] = {
    -0.5f, -0.5f, 0.0f,
    0.5f, -0.5f, 0.0f,
    0.0f, 0.5f, 0.0f
};
unsigned int VBO;
glGenBuffers(1, &VBO);
glBindBuffer(GL_ARRAY_BUFFER, VBO);
glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);

unsigned int VAO;
glGenVertexArrays(1, &VAO);
glBindVertexArray(VAO);
```

> 然后创建顶点着色器和片段着色器，编译，其中编译内容是由名为GLSL的类C语言编写。在这个阶段，我按照教程的教学打下来，可以出来三角形，但是颜色是纯白色的，然后经过调试，发现每次编译GLSL语言的结果都是错误，最后仔细对比了我的和教程的代码，我没有在GLSL最后加\0，导致了编译失败。

- *glCreateShader(GL_VERTEX_SHADER)*：创建一个顶点着色器并返回ID。
- *glShaderSource(vertexShader, 1, &vertexSource, NULL)*：把着色器源码附加到上一步创建的顶点着色器上。
- *glCompileShader(vertexShader)*：编译着色器源码，生成了一个着色器。

```
unsigned int vertexShader;
vertexShader = glCreateShader(GL_VERTEX_SHADER);
glShaderSource(vertexShader, 1, &vertexSource, NULL);
glCompileShader(vertexShader);
unsigned int fragmentShader;
fragmentShader = glCreateShader(GL_FRAGMENT_SHADER);
glShaderSource(fragmentShader, 1, &fragmentSource, NULL);
glCompileShader(fragmentShader);
```

将上面两个着色器程序链接起来，返回一个新的，可调用的着色器程序对象。之后就可以释放掉原本的着色器了。

```
unsigned int shaderProgram;
shaderProgram = glCreateProgram();
glAttachShader(shaderProgram, vertexShader);
glAttachShader(shaderProgram, fragmentShader);
glLinkProgram(shaderProgram);
```

接着链接顶点属性，根据之前的顶点数组大小设置步长，然后在`while`循环里面调用，就可以生成三角形了。

- *glVertexAttribPointer(0,3,GL_FLOAT,GL_FALSE,3*sizeof(float),(void*)0)*：告知OpenGL如何解析数据，如第二个参数，表示每个顶点数据有三个数字。
- *glEnableVertexAttribArray(0)*：以顶点属性位置值作为参数，启用顶点属性。
- *glClearColor(1.0f, 0.0f, 0.0f, 1.0f)*：设定背景框颜色。
- *glUseProgram(shaderProgram)*：激活之前生成的着色器程序。
- *glBindVertexArray(VAO)*：使用着色器，按照VAO中的属性配置三角形。
- *glDrawArrays(GL_TRIANGLES, 0, 3)*：绘制三角形，最后的参数表示绘制三个顶点。

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

while (!glfwWindowShouldClose(window)) {
     processInput(window);
     glClearColor(1.0f, 0.0f, 0.0f, 1.0f);
     glClear(GL_COLOR_BUFFER_BIT);
     glUseProgram(shaderProgram);
     glBindVertexArray(VAO);
     glDrawArrays(GL_TRIANGLES, 0, 3);
     glfwSwapBuffers(window);
     glfwPollEvents();
}
```

实验结果

见hw1.gif。

题目二

实验要求

> 对三角形的三个顶点分别改为红绿蓝，像下面这样，并解释为什么会出现这样的结果。

实验思路

> 题目二与上一题差别不大，就是更改了每个顶点的颜色属性，然后重新链接顶点属性即可。首先为三个顶点后面附上其颜色属性，为RGB，然后传入GLSL，编译即可获得效果。每行顶点数据的后三个表示顶点的RGB数据。再多加两行，告诉OpenGL如何解析数据即可，需要注意的是，由于有两个顶点属性：位置和颜色，所以第二个属性需要计算步长，然后再指定偏移量才可以。
> 为何会出现这样的结果：因为vertices中有两个属性，一个是位置，另一个是颜色，这两个属性通过GLSL语言编译，然后颜色属性的变量又作为输入传递给片段着色器的GLSL源码中，然后就可以渲染每个顶点的颜色了。

```
float vertices[] = {
     -0.5f, -0.5f, 0.0f, 0.0f, 1.0f, 0.0f,
     0.5f, -0.5f, 0.0f, 0.0f, 0.0f, 1.0f,
     0.0f, 0.5f, 0.0f, 1.0f, 0.0f, 0.0f
};
glVertexAttribPointer(0,3,GL_FLOAT,GL_FALSE,6*sizeof(float),(void*)0);
glEnableVertexAttribArray(0);
glVertexAttribPointer(1,3,GL_FLOAT,GL_FALSE,6*sizeof(float),(void*)(3*sizeof(float)));
glEnableVertexAttribArray(1);
```

实验结果

见hw2.gif。

题目三

实验要求

> 给上述工作添加一个GUI，里面有一个菜单栏，使得可以选择并改变三角形的颜色。

实验思路

> 首先将ImGui的头文件和实现cpp导入，然后在example中找到OpenGL的例子，模仿其调用函数。但是遇到的问题是，example用的不是glad，所以一开始出现了很多问题，我一直找那个库函数，但是找不到，最后将ImGui中某些cpp文件中的gl3w替换成glad，才可以正常运行函数。之后模仿其添加了一些组建，就可以正常编辑颜色了。

按照示例中写入开头，初始化，然后创建一个颜色变量TriangleColor，用于后续的编辑。

- *ImGui_ImplOpenGL3_Init(glsl_version)*：初始化OpenGL，参数为GLSL，不同版本的OpenGL对应不同，3.3对应的是130.

```
gladLoadGL();
IMGUI_CHECKVERSION();
ImGui::CreateContext();
ImGuiIO& io = ImGui::GetIO(); (void)io;
ImGui::StyleColorsDark();
ImGui_ImplGlfw_InitForOpenGL(window, true);
ImGui_ImplOpenGL3_Init(glsl_version);
ImVec4 TriangleColor = ImVec4(0.0f, 1.0f, 0.0f, 1.00f);
```

使用变量替换掉原本的常量，这样在每一次循环中，就可以重新渲染三角形，从而显示不同的颜色，需要注意的是，这一段代码需要放入while循环中。

```
float vertices[] = {
     -0.5f, 0.0f, 0.0f, TriangleColor.x, TriangleColor.y, TriangleColor.z,
     0.5f, 0.0f, 0.0f, TriangleColor.x, TriangleColor.y, TriangleColor.z,
     0.0f, 0.5f, 0.0f, TriangleColor.x, TriangleColor.y, TriangleColor.z,
     0.0f, -0.6f, 0.0f, TriangleColor.x, TriangleColor.y, TriangleColor.z,
     -0.5f, -0.1f, 0.0f, TriangleColor.x, TriangleColor.y, TriangleColor.z,
     0.5f, -0.1f, 0.0f, TriangleColor.x, TriangleColor.y, TriangleColor.z
     };
```

- *ImGui::ColorEdit3("Color", (float)*&TriangleColor)*：创建一个颜色选择器，其中数字返回到后面的参数中，用于调整颜色。
- *ImGui::Checkbox("Triangle", &showTriangle)*：选择框，选中，则后面的参数变为true。

```
ImGui_ImplOpenGL3_NewFrame();
ImGui_ImplGlfw_NewFrame();
ImGui::NewFrame();

ImGui::Begin("Edit color", &ImGui, ImGuiWindowFlags_MenuBar);
ImGui::ColorEdit3("Color", (float*)&TriangleColor);
ImGui::Checkbox("Triangle", &showTriangle);
ImGui::Checkbox("Line", &showLine);
ImGui::Checkbox("SecTriangle", &showSecTriangle);
ImGui::End();

// Rendering
ImGui::Render();
ImGui_ImplOpenGL3_RenderDrawData(ImGui::GetDrawData());
```

实验结果

见hw3.gif。

Bonus:

题目一

实验要求

> 绘制其他的图元，除了三角形，还有点、线等。

实验思路

> 其实很简单，只要将绘制三角形的那句换为线，然后使用两个顶点就可以绘制线。同理，绘点也是如此。

```
glBindVertexArray(VAO);
glDrawArrays(GL_LINE_STRIP, 0, 2);
```

实验结果

见hw3.gif。

题目二

实验要求

> 使用EBO(Element Buffer Object)绘制多个三角形。

实验思路

> 创建一个索引数组，更新顶点数组，当需要绘制第二个三角形的时候，按照索引数组中的顺序绘制顶点，这样加上原本的三角形，就有了两个三角形。需要注意的是，索引数组的顺序不能与第一个三角形顺序相同，否则就重叠了。

- *glDrawElements(GL_TRIANGLES, 3, GL_UNSIGNED_INT, 0)*：第二个参数，表示绘制三个顶点。

```
float vertices[] = {
-0.5f, 0.0f, 0.0f, TriangleColor.x, TriangleColor.y, TriangleColor.z,
0.5f, 0.0f, 0.0f, TriangleColor.x, TriangleColor.y, TriangleColor.z,
0.0f, 0.5f, 0.0f, TriangleColor.x, TriangleColor.y, TriangleColor.z,
0.0f, -0.6f, 0.0f, TriangleColor.x, TriangleColor.y, TriangleColor.z,
-0.5f, -0.1f, 0.0f, TriangleColor.x, TriangleColor.y, TriangleColor.z,
0.5f, -0.1f, 0.0f, TriangleColor.x, TriangleColor.y, TriangleColor.z
};

unsigned int indices[] = {
     3, 4, 5
};
glGenBuffers(1, &EBO);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO);
glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices), indices, GL_STATIC_DRAW);

glBindVertexArray(VAO);
glDrawElements(GL_TRIANGLES, 3, GL_UNSIGNED_INT, 0);
glBindVertexArray(0);
```

实验结果

见hw3.gif。