Homework 2 - GUI and Draw simple graphics

Homework 2 - GUI and Draw simple graphics
- 作业要求
- 绘制三角形
- 改变三角形的三个顶点颜色
- GUI

作业要求

Basic:

1. 使用OpenGL(3.3及以上)+GLFW或freeglut画一个简单的三角形。
2. 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。
3. 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。

绘制三角形

开始绘制图形之前，我们必须先给OpenGL输入一些顶点数据。OpenGL是一个3D图形库，所以我们在OpenGL中指定的所有坐标都是3D坐标（x、y和z）。OpenGL不是简单地把**所有的**3D坐标变换为屏幕上的2D像素；OpenGL仅当3D坐标在3个轴（x、y和z）上都为-1.0到1.0的范围内时才处理它。所有在所谓的标准化设备坐标(Normalized Device Coordinates)范围内的坐标才会最终呈现在屏幕上（在这个范围以外的坐标都不会显示）。

由于我们希望渲染一个三角形，我们一共要指定三个顶点，每个顶点都有一个3D位置。我们会将它们以标准化设备坐标的形式（OpenGL的可见区域）定义为一个`float`数组。

定义这样的顶点数据以后，我们会把它作为输入发送给图形渲染管线的第一个处理阶段：顶点着色器。它会在GPU上创建内存用于储存我们的顶点数据，还要配置OpenGL如何解释这些内存，并且指定其如何发送给显卡。顶点着色器接着会处理我们在内存中指定数量的顶点。

我们通过顶点缓冲对象(Vertex Buffer Objects, VBO)管理这个内存，它会在GPU内存（通常被称为显存）中储存大量顶点。使用这些缓冲对象的好处是我们可以一次性的发送一大批数据到显卡上，而不是每个顶点发送一次。从CPU把数据发送到显卡相对较慢，所以只要可能我们都要尝试尽量一次性发送尽可能多的数据。当数据发送至显卡的内存中后，顶点着色器几乎能立即访问顶点，这是个非常快的过程。

要想绘制我们想要的物体，OpenGL给我们提供了glDrawArrays函数，它使用当前激活的着色器，之前定义的顶点属性配置，和VBO的顶点数据（通过VAO间接绑定）来绘制图元。

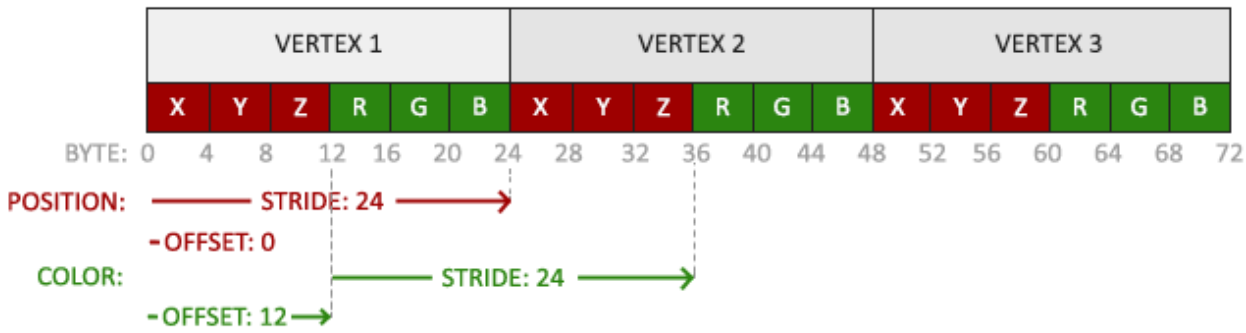
```
    unsigned int VBO, VAO;
    glGenVertexArrays(1, &VAO);
    glGenBuffers(1, &VBO);
```

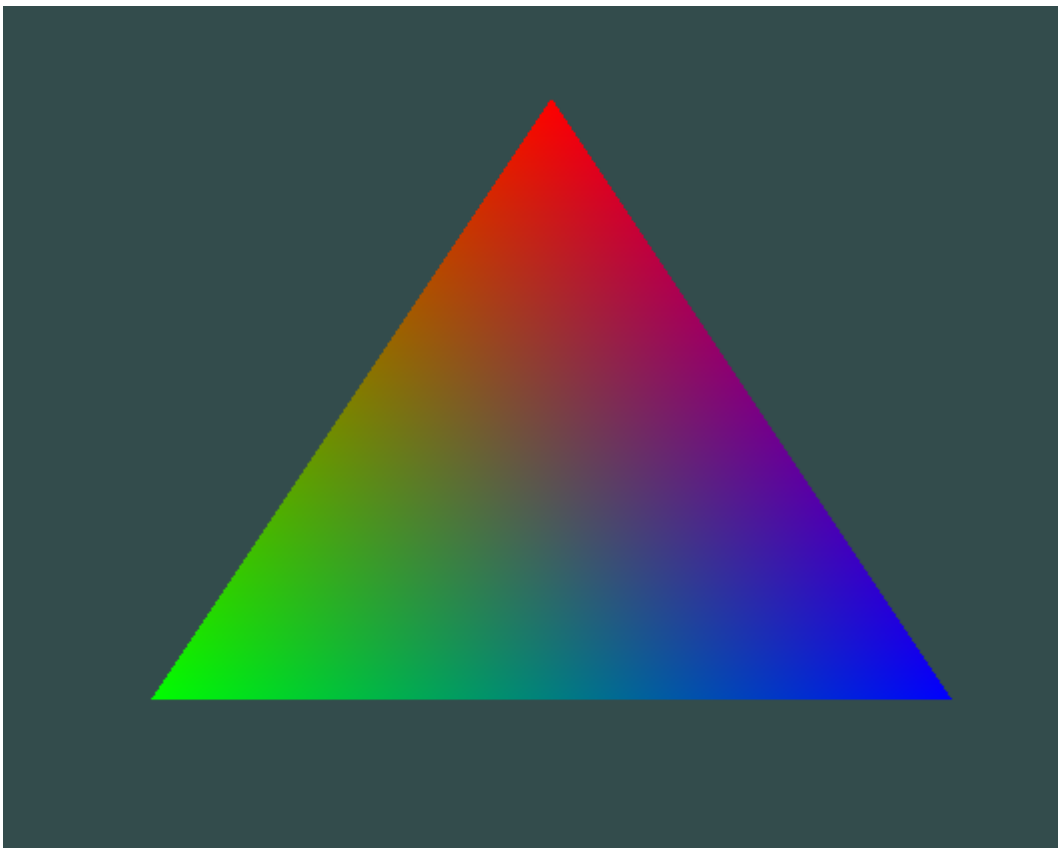
```
    // bind the Vertex Array Object first, then bind and set vertex buffer(s), and then
configure vertex attributes(s).
    glBindVertexArray(VAO);

    glBindBuffer(GL_ARRAY_BUFFER, VBO);
    glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);

    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);

    // note that this is allowed, the call to glVertexAttribPointer registered VBO as
the vertex attribute's bound vertex buffer object so afterwards we can safely unbind
    glBindBuffer(GL_ARRAY_BUFFER, 0);

    // You can unbind the VAO afterwards so other VAO calls won't accidentally modify
this VAO, but this rarely happens. Modifying other
    // VAOs requires a call to glBindVertexArray anyways so we generally don't unbind
VAOs (nor VBOs) when it's not directly necessary.
    glBindVertexArray(0);
    shader.use();

    glDrawArrays(GL_TRIANGLES, 0, 3);
    glBindVertexArray(VAO); // seeing as we only have a single VAO there's no need to
bind it every time, but we'll do so to keep things a bit more organized
    glBindBuffer(GL_ARRAY_BUFFER, VBO);
    glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);

    glDrawArrays(GL_TRIANGLES, 0, 3);
```

改变三角形的三个顶点颜色

我们需要对顶点数组加入颜色属性，因此我们将顶点数组改为

```
ImVec4 top = ImVec4(1.0f, 0.0f, 0.0f, 1.0f);
ImVec4 left = ImVec4(0.0f, 1.0f, 0.0f, 1.0f);
ImVec4 right = ImVec4(0.0f, 0.0f, 1.0f, 1.0f);

float vertices[] = {
// location        // color
0.0f,  0.5f, 0.0f, top.x, top.y, top.z,
-0.5f, -0.5f, 0.0f, left.x, left.y, left.z,
0.5f, -0.5f, 0.0f, right.x, right.y, right.z
};
```





在我们设置顶点颜色后，会发现，三角形呈现出一种混合和渐变的效果，这是因为，这是在片段着色器中进行的所谓片段插值(Fragment Interpolation)的结果。当渲染一个三角形时，光栅化(Rasterization)阶段通常会造成比原指定顶点更多的片段。光栅会根据每个片段在三角形形状上所处相对位置决定这些片段的位置。 基于这些位置，它会插值(Interpolate)所有片段着色器的输入变量。比如说，我们有一个线段，上面的端点是绿色的，下面的端点是蓝色的。如果一个片段着色器在线段的70%的位置运行，它的颜色输入属性就会是一个绿色和蓝色的线性结合；更精确地说就是30%蓝 + 70%绿。

GUI

```
        ImGui::Begin("Color Setting\n");
        ImGui::SetWindowSize(ImVec2(300, 220));

        ImGui::ColorEdit3("top", (float*)&top);
        ImGui::ColorEdit3("left", (float*)&left);
        ImGui::ColorEdit3("right", (float*)&right);

        ImGui::End();
```