Lista de Exercícios 1 – Processamento Gráfico

Introdução à OpenGL Moderna – Shaders & Buffers

2. *O que é a GLSL? Quais os dois tipos de shaders são obrigatórios no pipeline programável da versão atual que trabalhamos em aula e o que eles processam?*

O GLSL é uma linguagem de shaders para utilizar com o OpenGL. Os dois tipos de shaders obrigatórios são Fragment Shader e Vertex Shader. Um Fragment Shader é o estágio de Shader que processará um fragmento gerado pela Rasterização em um conjunto de cores e um único valor de profundidade. O Vertex Shader é o estágio de Shader programável no pipeline de renderização que lida com o processamento de vértices individuais.

3. *O que são primitivas gráficas? Como fazemos o armazenamento dos vértices na OpenGL?*

O OpenGL possui diversas primitivas de desenho, ou seja, formas de interpretar informação geométrica. Um objeto 2D é definido por seus vértices (geometria) e arestas ou faces (topologia). Em OpenGL, a partir de um conjunto de vértices é possível desenhar primitivas baseadas em pontos e linhas (GL_POINTS, GL_LINES etc.) e primitivas baseadas em faces (GL_QUADS, GL_TRIANGLES etc.). A diferença é que por definição, as primitivas baseadas em linhas apenas geram um traçado, enquanto as baseadas em faces apenas PINTAM o conteúdo (não traçam).

4. *Explique o que é VBO, VAO e EBO, e como se relacionam (se achar mais fácil, pode fazer um gráfico representando a relação entre eles).*

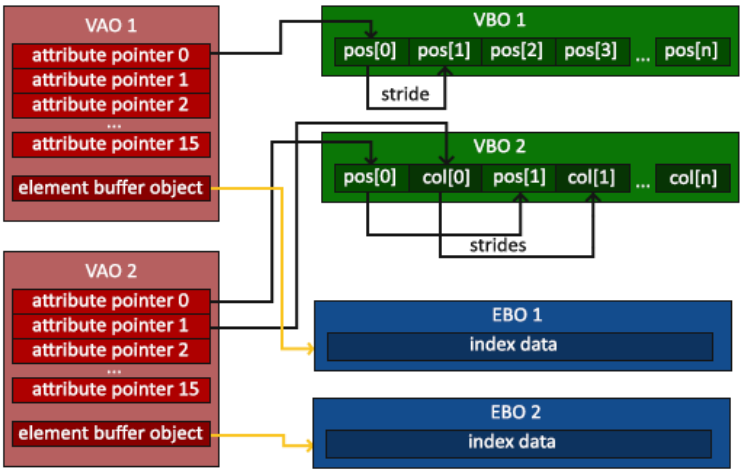
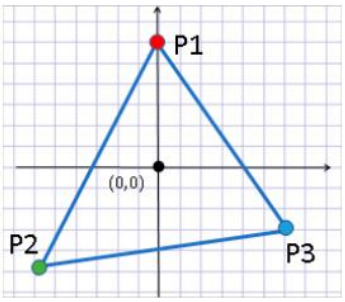


Imagem que melhor representa os conceitos de VBO, VAO e EBO e como se relacionam. Imagem extraída do site: LearnOpenGL.

5. *Considerando o seguinte triângulo abaixo, formado pelos vértices P1, P2 e P3, respectivamente com as cores vermelho, verde e azul.*



a. Descreva uma possível configuração dos buffers (VBO, VAO e EBO) para representá-lo.

No caso da figura, cada vértice deve conter 2 atributos, posição e cor, ocupando 24 bytes de memória de forma contigua. O atributo POSIÇÃO possui 3 valores e ocupa 12 bytes. Para encontrar este atributo precisamos nos deslocar 0 bytes (é o primeiro). O atributo COR possui 3 valores e ocupa 12 bytes. Para encontrar este atributo precisamos nos deslocar 12 bytes (3 * sizeof (GLfloat), na sequência.)

VBO

```
GLuint vVBO;
glGenBuffers(1, &vVBO);
glBindBuffer(GL_ARRAY_BUFFER,
vVBO);
glBufferData(GL_ARRAY_BUFFER, 18 *
sizeof(GLfloat), vertices,
GL_STATIC_DRAW);
```

```
GLuint VAO;
glGenVertexArrays(1, &VAO);
glBindVertexArray(VAO);

glBindBuffer(GL_ARRAY_BUFFER, vVBO);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 *
sizeof(GLfloat), (GLvoid*)0);
glEnableVertexAttribArray(0);
```

VAO

```
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 *
sizeof(GLfloat), (GLvoid*)(3 * sizeof(GLfloat)));
glEnableVertexAttribArray(1);
```

Nas imagens, alguns dos exemplos de aula.

b. Como estes atributos seriam identificados no vertex shader?

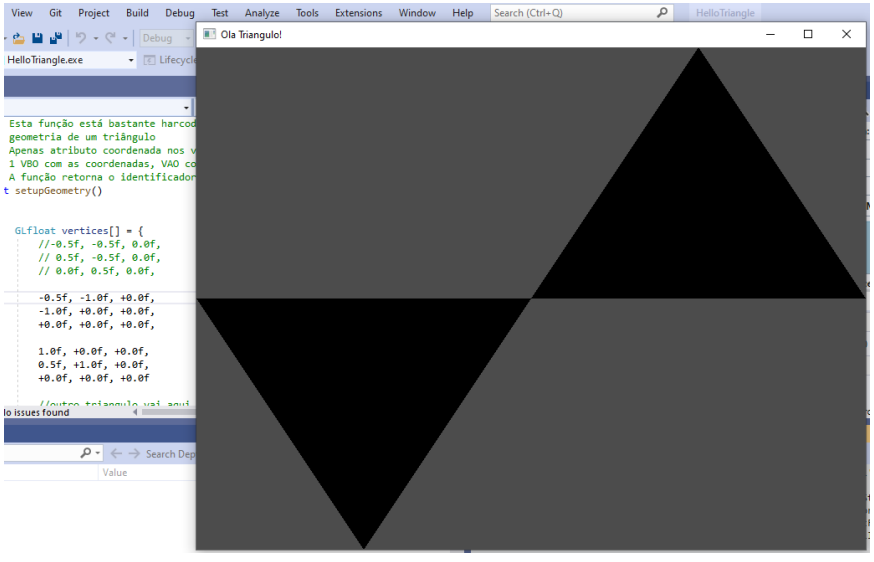
No vertex shader, indicaremos a localização de cada atributo, como na imagem de exemplo:

```
layout (location = 0) in vec3 position;
layout (location = 1) in vec3 color;
```

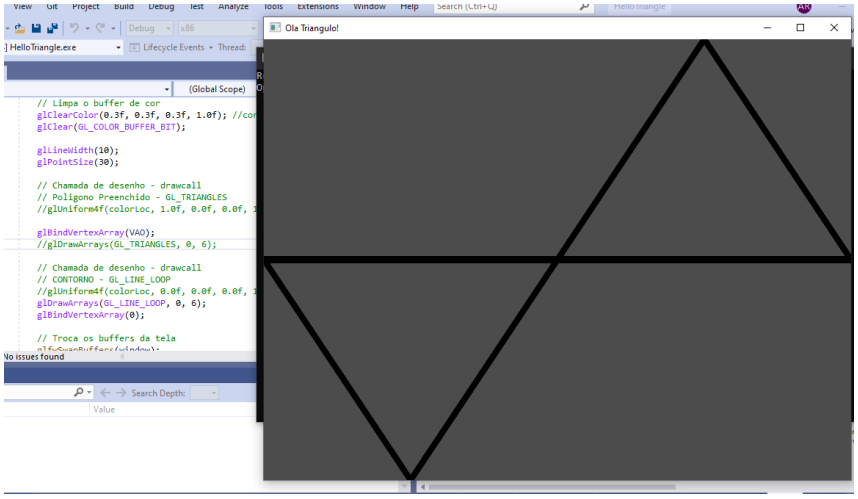
Imagem retirado do exemplo de aula.

6. *Faça o desenho de 2 triângulos na tela. Desenhe eles:*

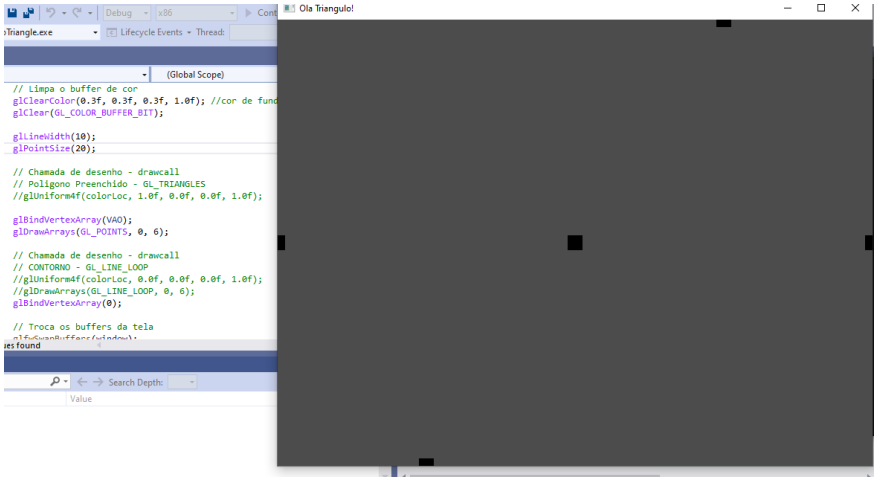
a. Apenas com o polígono preenchido;



b. Apenas com contorno;



c. Apenas como pontos;



d. Com as 3 formas de desenho juntas.