Lista de Exercícios 1 – Processamento Gráfico

Introdução à OpenGL Moderna – Shaders & Buffers

0. Leitura OBRIGATÓRIA para começar:

https://learnopengl.com/#!Getting-started/Hello-Triangle

https://learnopengl.com/#!Getting-started/Shaders

http://antongerdelan.net/opengl/hellotriangle.html

Sugere-se ainda a leitura:

- Capítulo 2 do livro *Real Time Rendering* (pdf no Canvas, gerado pelo próprio ebscohost)
- Seção 5.1 (Etapas da Renderização) do livro *Computação Gráfica - Teoria e Prática: Geração de Imagens*

1. 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?

GLSL é a OpenGL Shading Language, **vertex** e **fragment** shaders o estágio de **Tesselation** e **Geometry** são opcionais.

Vertex: Processa cada vértice separadamente, descrevem como tratar um vértice(posições 2d, coordenadas de textura, cor)

Fragment: Processa cada fragmento separadamente, descrevem como tratar uma área (cor, Z-depth, Alpha value)

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

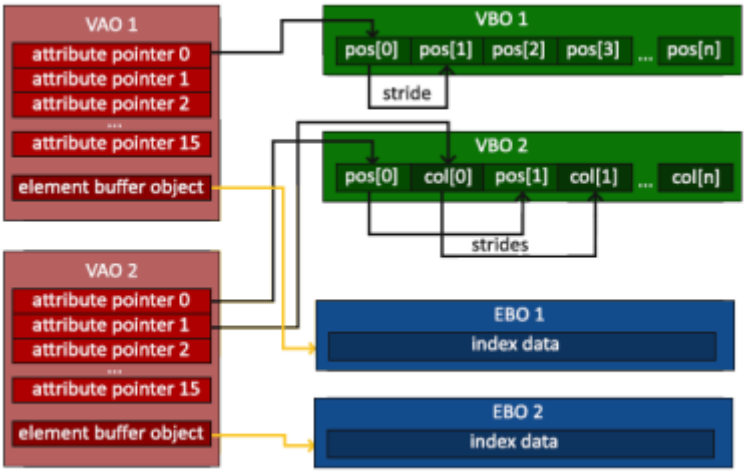
As primitivas são os elementos gráficos mais simples que podem ser criados dentro de uma aplicação. Armazenamos através dos VBOs(Vertex Buffer Objects) que são array de dados, geralmente float e servem para enviar dados dos vértices a GPU como posição, vetores normais , cores e etc.

3. 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).

VBO: Armazena os vértices

VAO: Armazena atributos pertinente a um vértice

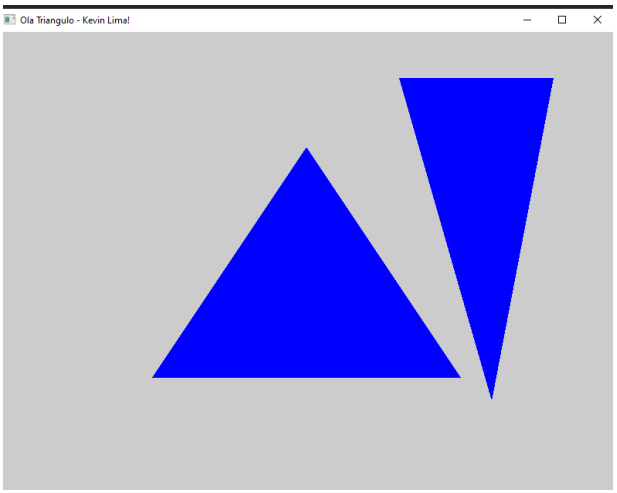
EBO: É como um índice e evita que tenhamos vértices replicados no VBO.



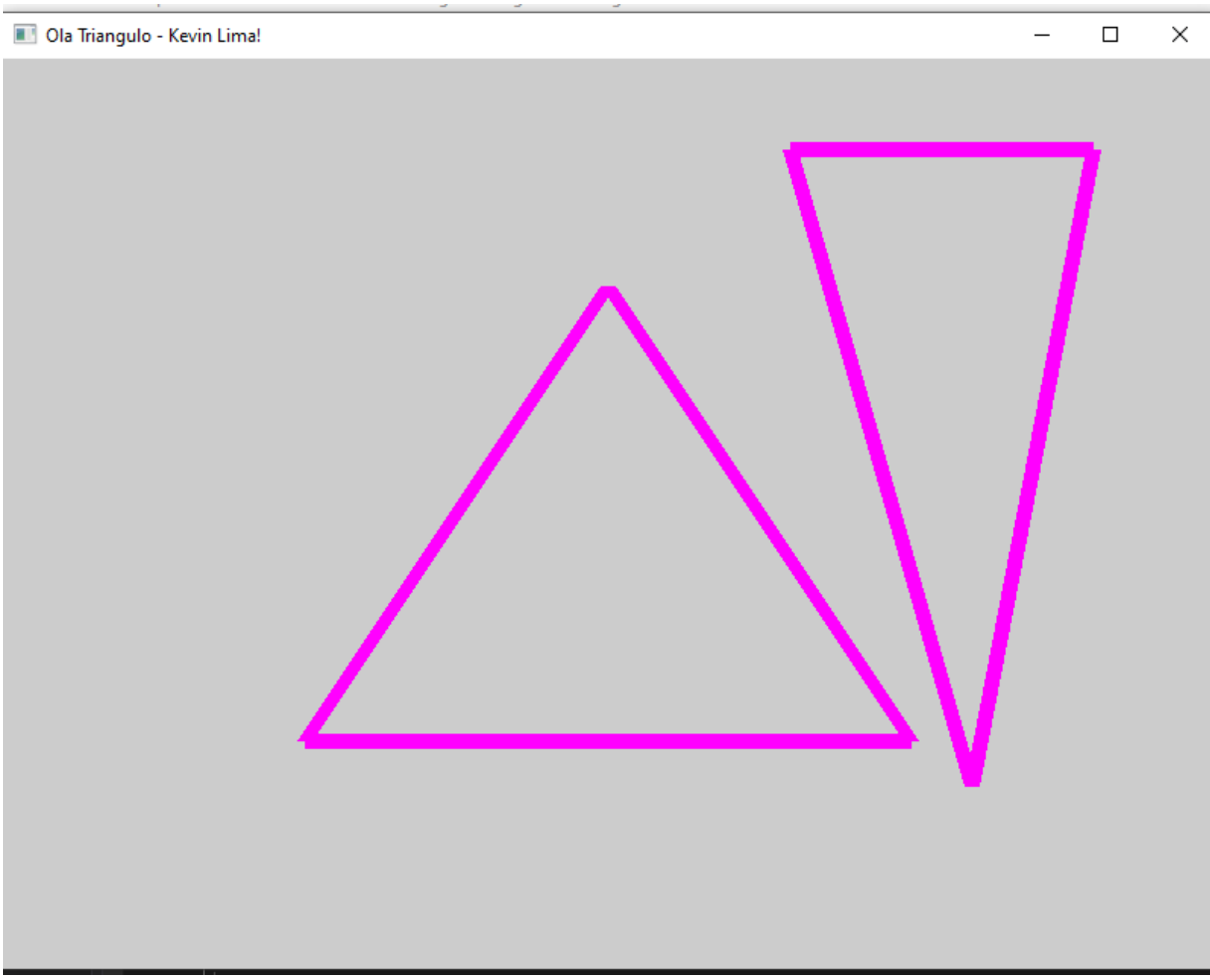
4. Analise o código fonte do projeto Hello Triangle. Localize e relacione os conceitos de shaders, VBOs e VAO apresentados até então. Não precisa entregar nada neste exercício.

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

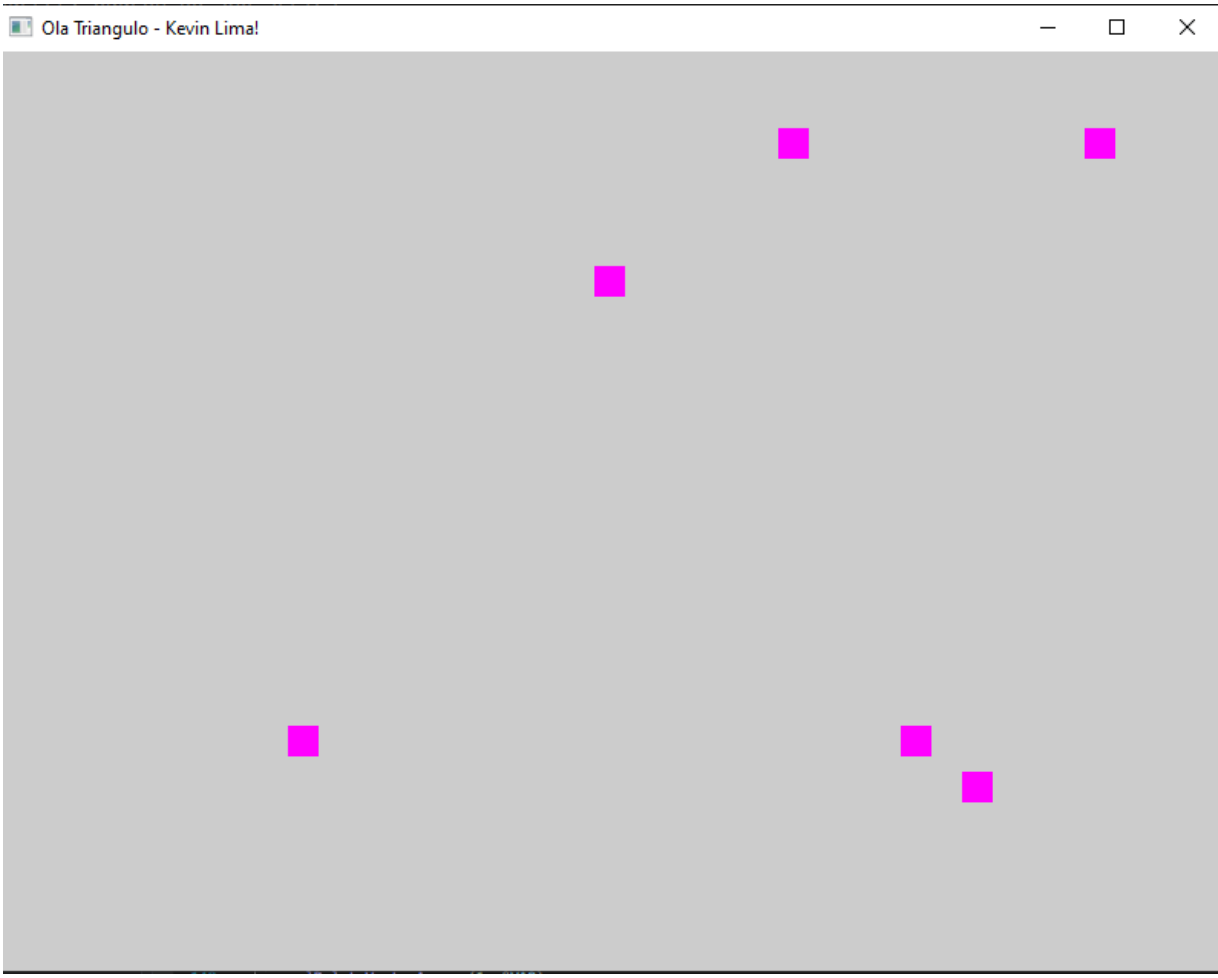
a. Apenas com o polígono preenchido



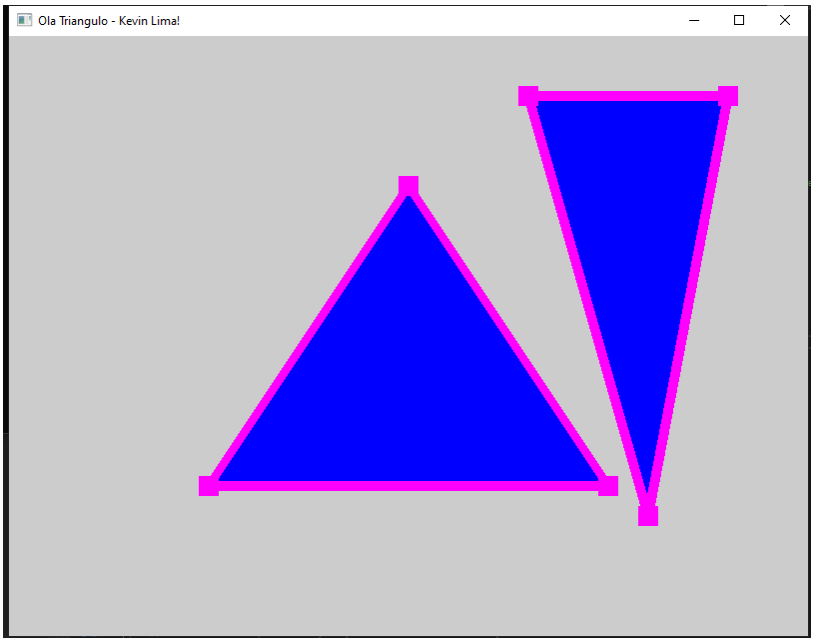
b. Apenas com contorno



c. Apenas como pontos



d. Com as 3 formas de desenho juntas



6. Faça o desenho de um círculo na tela, utilizando a equação paramétrica do círculo para

gerar os vértices. Depois disso:

a) Desenhe um octágono

b) Desenhe um pentágono

c) Desenhe um pac-man!

d) Desenhe uma fatia de pizza

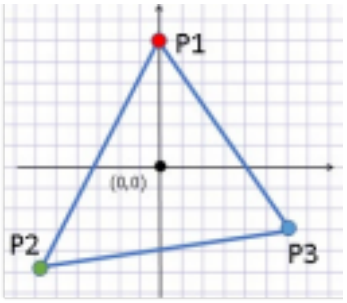
e) DESAFIO: desenhe uma “estrela”

7. Desenhe uma espiral, assim:

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

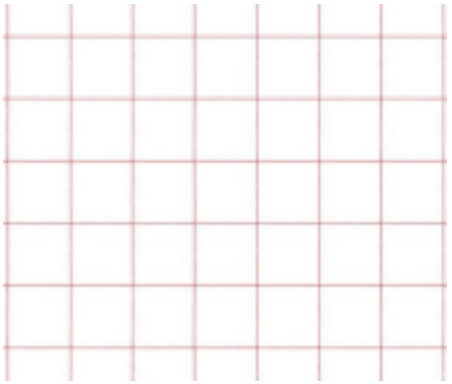
a. Descreva uma possível configuração dos buffers (VBO, VAO e EBO) para representá-lo.

b. Como estes atributos seriam identificados no *vertex shader*?



Agora implemente!

9. Faça um desenho em um papel quadriculado (pode ser no computador mesmo) e reproduza-o utilizando primitivas em OpenGL. Neste exercício você poderá criar mais de um VAO e fazer mais de uma chamada de desenho para poder utilizar primitivas diferentes, se necessário.



~~10. Implemente (pode pegar do tutorial) uma classe para tratar os *shaders* a partir de arquivos.~~ FEITO EM AULA E ATUALIZADO O REPOSITÓRIO!

Entrega individual via Moodle (consulte a data de entrega no sistema)