Lista de exercícios 1 – Processamento Gráfico Aluna: Isadora Ghisleni

Introdução à OpenGL Moderna – *Shaders & Buffers*

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?

   A GLSL é uma linguagem de programação que é usada no desenvolvimento de shaders dentro do pipeline da OpenGL. Existem dois tipos de shaders que são obrigatórios. Um deles é o **Vertex Shader**, que processa separadamente cada vértive e descreve onde cada posição desse vértice estará na tela. O outro é o **Fragment Shader**, que faz o processo de cada fragmento separadamente e ainda descreve o tratamento dos pixels entre os vértices.

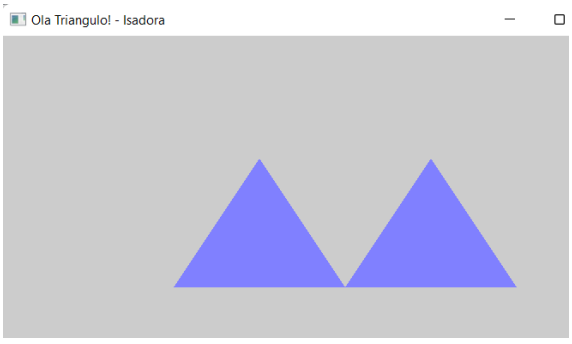
2. O que são primitivas gráficas? Como fazemos o armazenamento dos vértices na OpenGL?

   As primitivas gráficas são os elementos mais básicos que podem ser criados na programação gráfica. Exemplos são pontos que, quando conectados, formam polígonos para então formar um objeto. Na Open GL, os vértices são armazenados pelo buffer (VBO), que é um array de dados que envia os dados dos vértices para a GPU, onde os dados são alocados na própria memória da GPU.

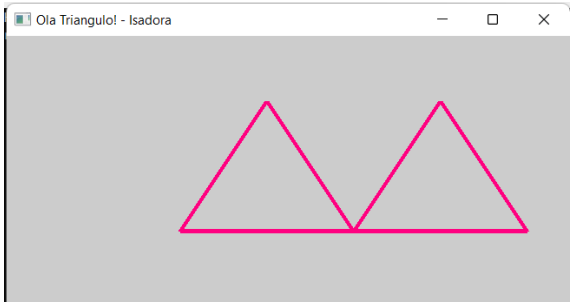
3. Explique o que é VBO, VAO e EBO, e como se relacionam (se achar mais fácil, pode fazer um gráfico representando a relação entre eles).

   O **VBO** é um Buffer que guarda um array de dados e manda informação dos vértices para a GPU. O **VAO** faz a conexão dos atributos de um vértice e é quem gerencia posição, cores ou normais. Ele descreve a forma como esses atributos dos vértices são armazenados em um VBO. E o **EBO** é um Buffer que permite armazenar índices que a OpenGL usa para decidir quais vértices desenhar.

4. .
5. Faça o desenho de 2 triângulos na tela. Desenhe eles:
   a. Apenas com o polígono preenchido.

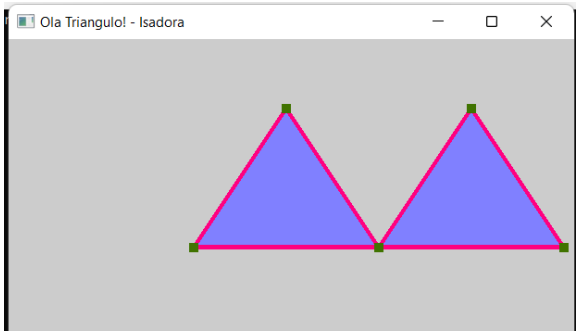


   b. Apenas com contorno.

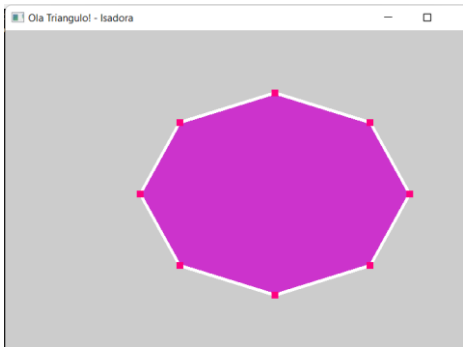


c. Apenas como pontos.

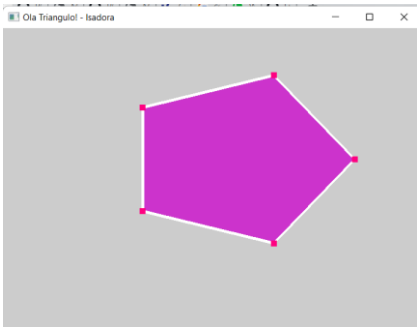
d. Com as 3 formas de desenho juntas.



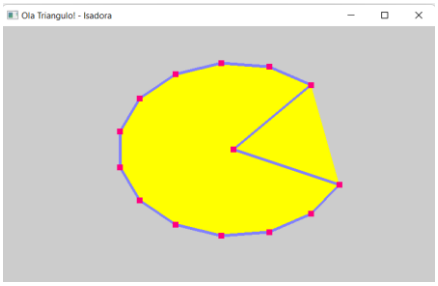
6. Faça o desenho de um círculo na tela, utilizando a equação paramétrica do círculo para gerar os vértices. Depois disso:
   a. Desenhe um octágono.



b. Desenhe um pentágono.

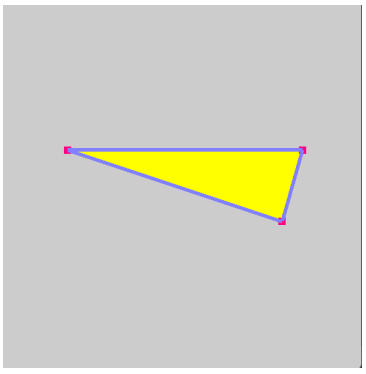


c. Desenhe um pac-man!

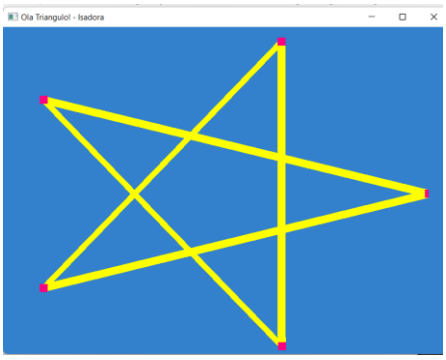


Obs.: não consegui deixar o preenchimento do desenho dentro das linhas.

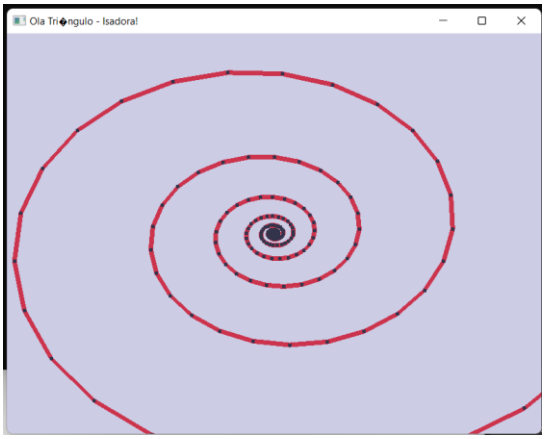
d. Desenhe uma fatia de pizza.



e. DESAFIO: desenhe uma “estrela”.



7. Desenhe uma espiral.



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.

```
// coordenadas dos vértices

GLfloat vertices[] =
{
    0.0,  0.6, 0.0,
    1.0,  0.0, 0.0,

   -0.6, -0.5, 0.0,
    0.0,  1.0, 0.0,

    0.6, -0.3, 0.0,
    0.0,  0.0, 0.1,

};
```

```
GLuint vVBO, VAO;
    //Geração do identificador do VBO
glGenBuffers(1, &vVBO);
glBindBuffer(GL_ARRAY_BUFFER, vVBO);
glBufferData(GL_ARRAY_BUFFER, 18 * sizeof(GLfloat),
vertices, GL_STATIC_DRAW);

    //Geração do identificador do 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);

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

b. Como estes atributos seriam identificados no vertex shader?

```
// Código fonte do Vertex Shader (em GLSL): ainda hardcoded
const GLchar* vertexShaderSource = "#version 430\n"
"layout (location = 0) in vec3 position;\n"
"layout (location = 1) in vec3 color;/n"

"out vec4 vertexColor;/n"
"void main()\n"
"{\n"

"gl_Position = vec4(position, 1.0);\n"
"vertexColor = vec4(color, 1.0);/n"
"}\0";
```

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.

   Utilizando o software GeoGebra, foi criado um rascunho de uma borboleta. Ao lado, a figura implementada e complementada no VisualStudio.