Lista 1

Feito por: Bruno Bueno Flores

Questões teóricas:

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?**

   R: GLSL (OpenGL Shading Language) é uma linguagem de programação usada para escrever shaders no contexto da OpenGL. Shaders são pequenos programas que são executados diretamente na GPU (unidade de processamento gráfico) e são responsáveis por realizar operações gráficas, como manipulação de vértices e pixels para renderizar imagens de maneira eficiente.

   No pipeline programável da OpenGL, existem dois tipos de shaders que são obrigatórios:

   1. Vertex Shader: Processa cada vértice individualmente. Ele transforma as coordenadas dos vértices de um objeto tridimensional (no espaço do mundo) para o espaço de tela, aplicando transformações como rotação, escala e translação. Além disso, ele pode calcular propriedades dos vértices, como cores e coordenadas de textura.

   2. Fragment Shader (ou Pixel Shader): É responsável por processar cada fragmento (basicamente um "pré-pixel") gerado pelo rasterizador após o processamento dos vértices. O fragment shader calcula a cor final de cada pixel na tela, aplicando efeitos como iluminação, texturas e sombras.

   Esses dois shaders são essenciais para a renderização gráfica moderna em OpenGL.

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

   R: Primitivas gráficas são os elementos básicos que a OpenGL usa para desenhar formas geométricas, como pontos, linhas e triângulos.

   Para armazenar vértices na OpenGL, usamos **Vertex Buffer Objects (VBOs)**. O processo envolve:
   1. Criar o VBO com “glGenBuffers()”.
   2. Ligar o VBO ao contexto de buffer com “glBindBuffer()”.
   3. Copiar os dados de vértices para o VBO com “glBufferData()”.
   4. Configurar e habilitar os atributos de vértices com “glVertexAttribPointer()” e “glEnableVertexAttribArray()”.

   Isso permite à OpenGL acessar os dados diretamente na GPU para renderizar primitivas.

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).**

   R:Vamos explicar cada um desses elementos e como eles se relacionam no pipeline de renderização da OpenGL:

   - **VBO (Vertex Buffer Object)**

   - O **VBO** armazena os dados dos vértices (como posições, cores, normais, coordenadas de textura) na memória da GPU. Esses dados são enviados uma vez e podem ser reutilizados várias vezes, permitindo uma renderização mais eficiente. O VBO é ligado com `glBindBuffer(GL_ARRAY_BUFFER, vbo)` e configurado com `glBufferData()`.

   - **VAO (Vertex Array Object)**

   - O **VAO** é uma estrutura que armazena a configuração do estado de como os dados dos vértices serão interpretados. Ele guarda as chamadas de configuração dos atributos de vértice feitas por meio dos VBOs, como quais buffers estão ligados e como os dados são interpretados (por exemplo, com `glVertexAttribPointer()`). O VAO permite que a OpenGL "lembre" como acessar os dados de vértices de forma organizada e facilita o desenho de objetos com menos chamadas de configuração repetitivas.

   - **EBO (Element Buffer Object)**

   - O **EBO** (também chamado de **Index Buffer**) armazena índices que referenciam vértices em um VBO, permitindo a reutilização dos mesmos vértices para formar várias primitivas gráficas (como triângulos). Isso reduz a duplicação de dados no VBO e melhora o desempenho ao desenhar formas complexas.

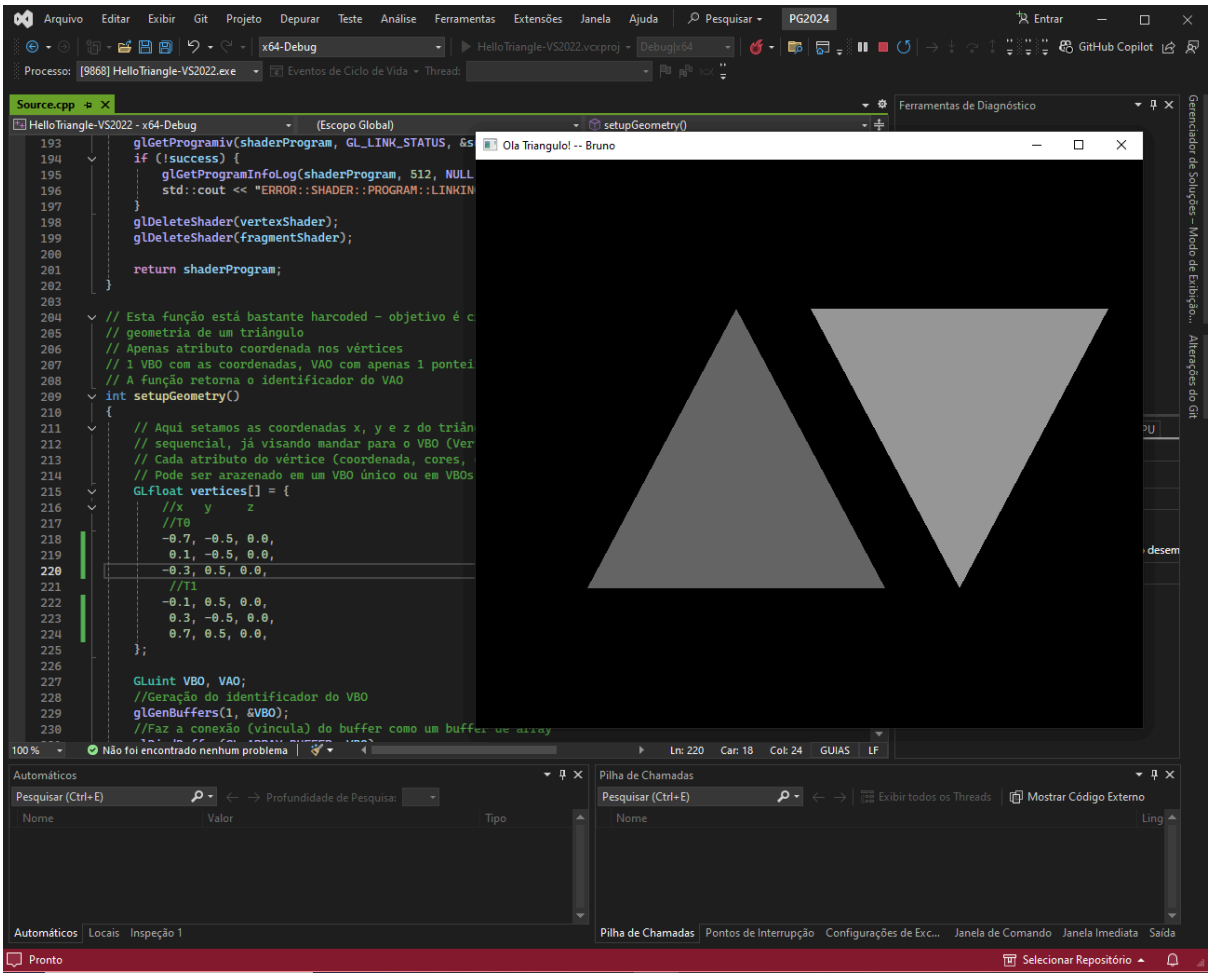
   - **Relação entre VBO, VAO e EBO**

   - O VBO contém os dados dos vértices.
   - O EBO contém os índices que referenciam os vértices do VBO.
   - O VAO armazena o estado de como esses buffers (VBO e EBO) estão configurados e conectados entre si, facilitando o processo de renderização.

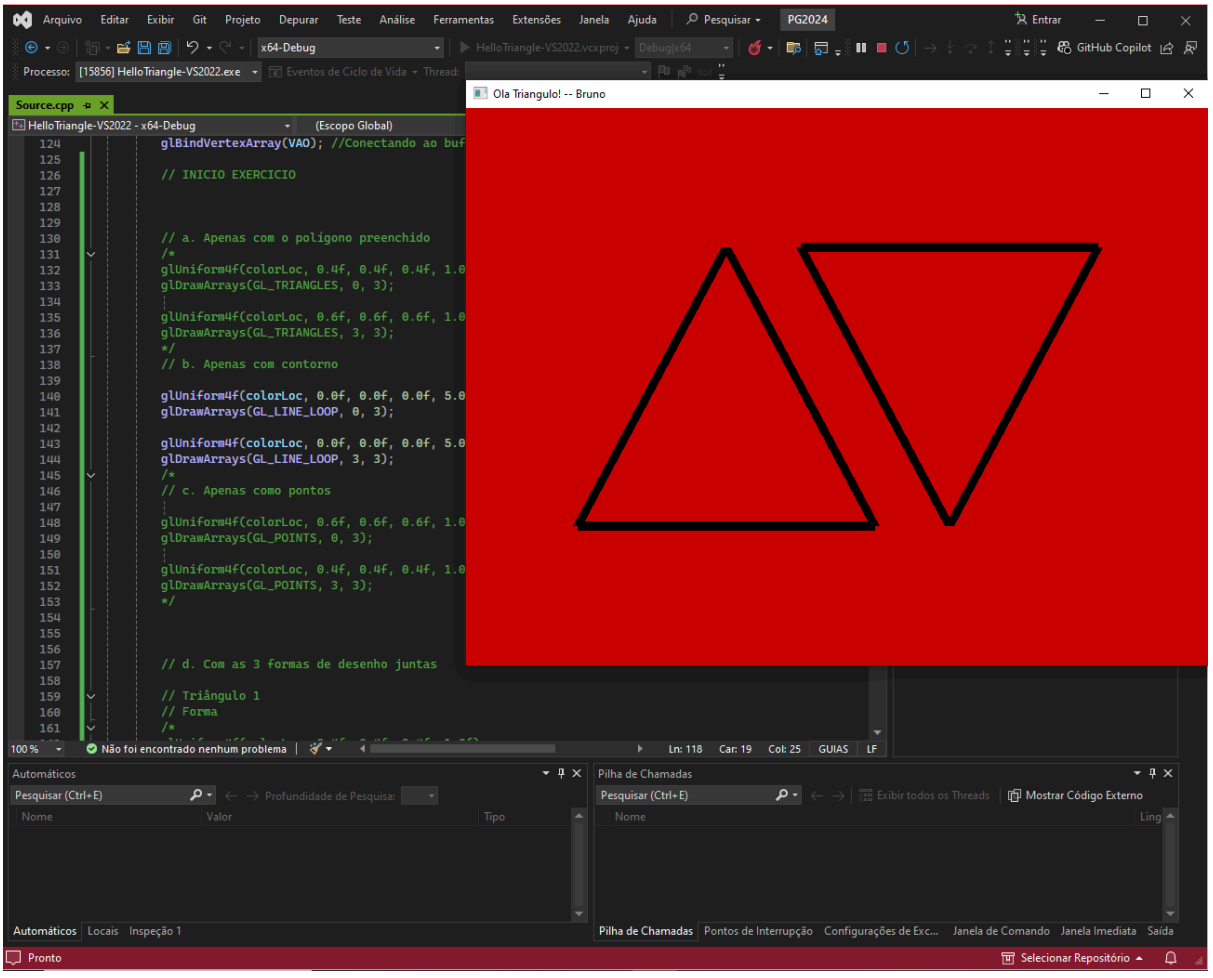
4. **Analise o código fonte do projeto Hello Triangle. Localize e relacione os conceitos de shaders, VBOs e VAO apresentados até então. Não precisa entregar nada neste exercício.**

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

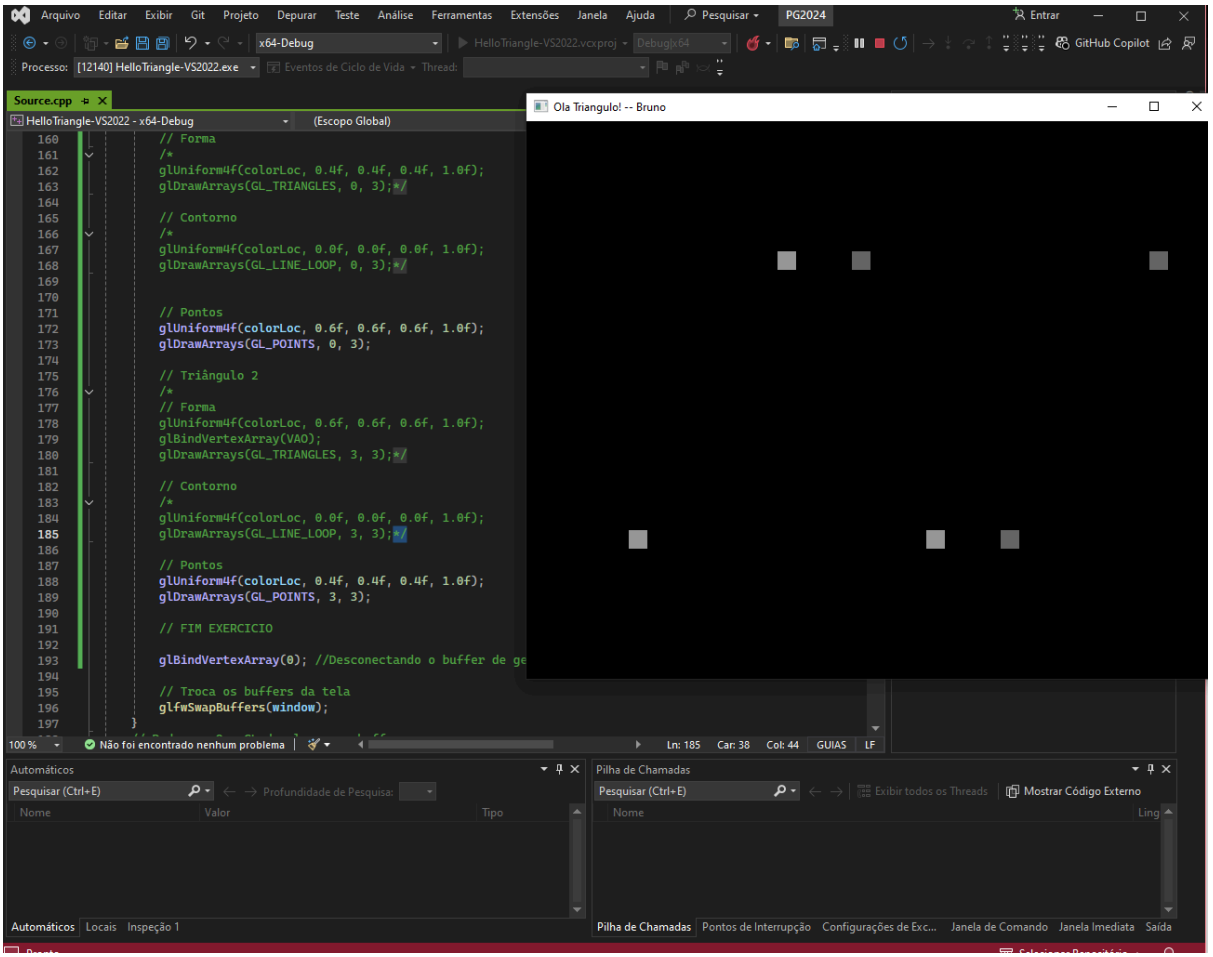
a. Apenas com o polígono preenchido



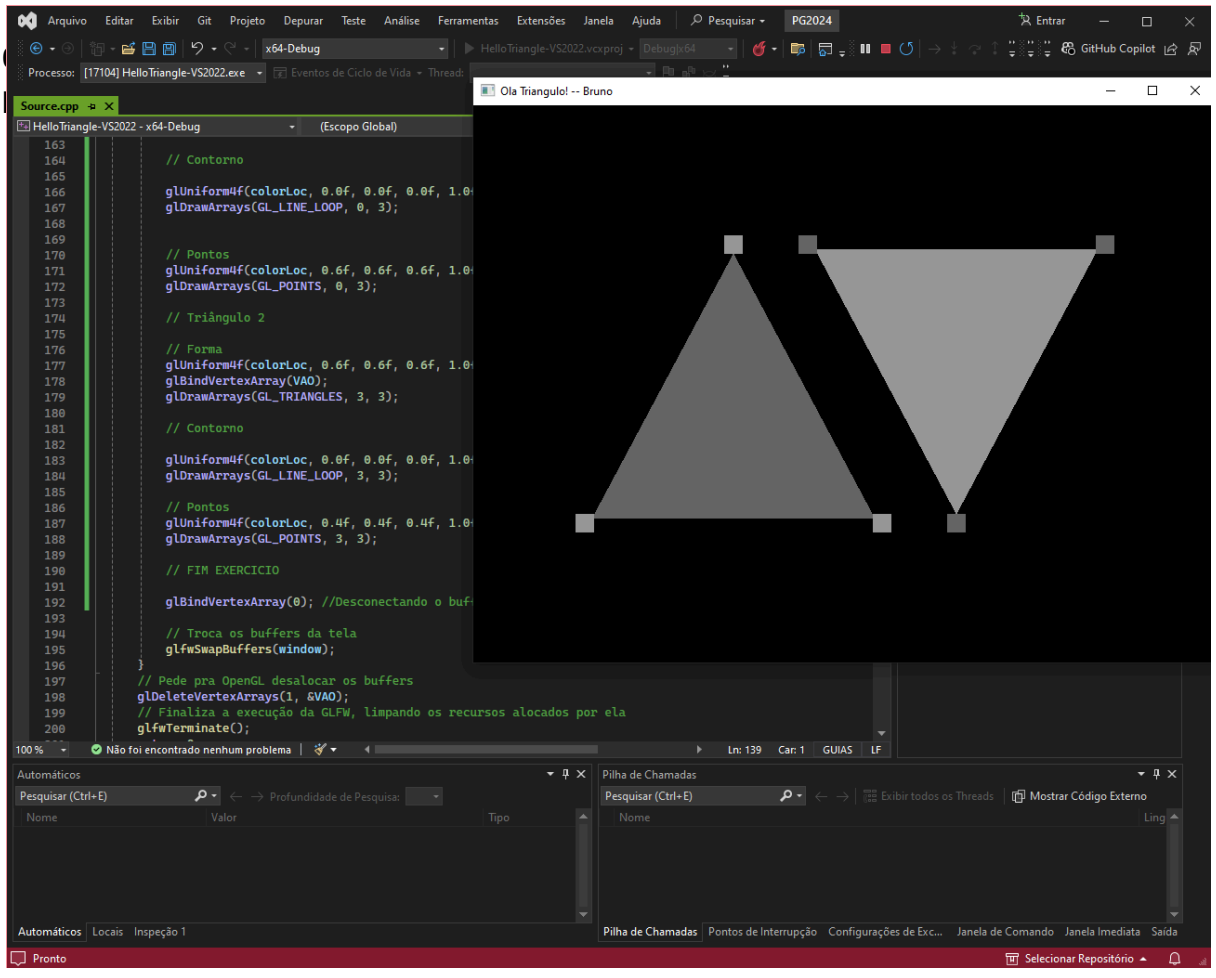
b. Apenas com contorno



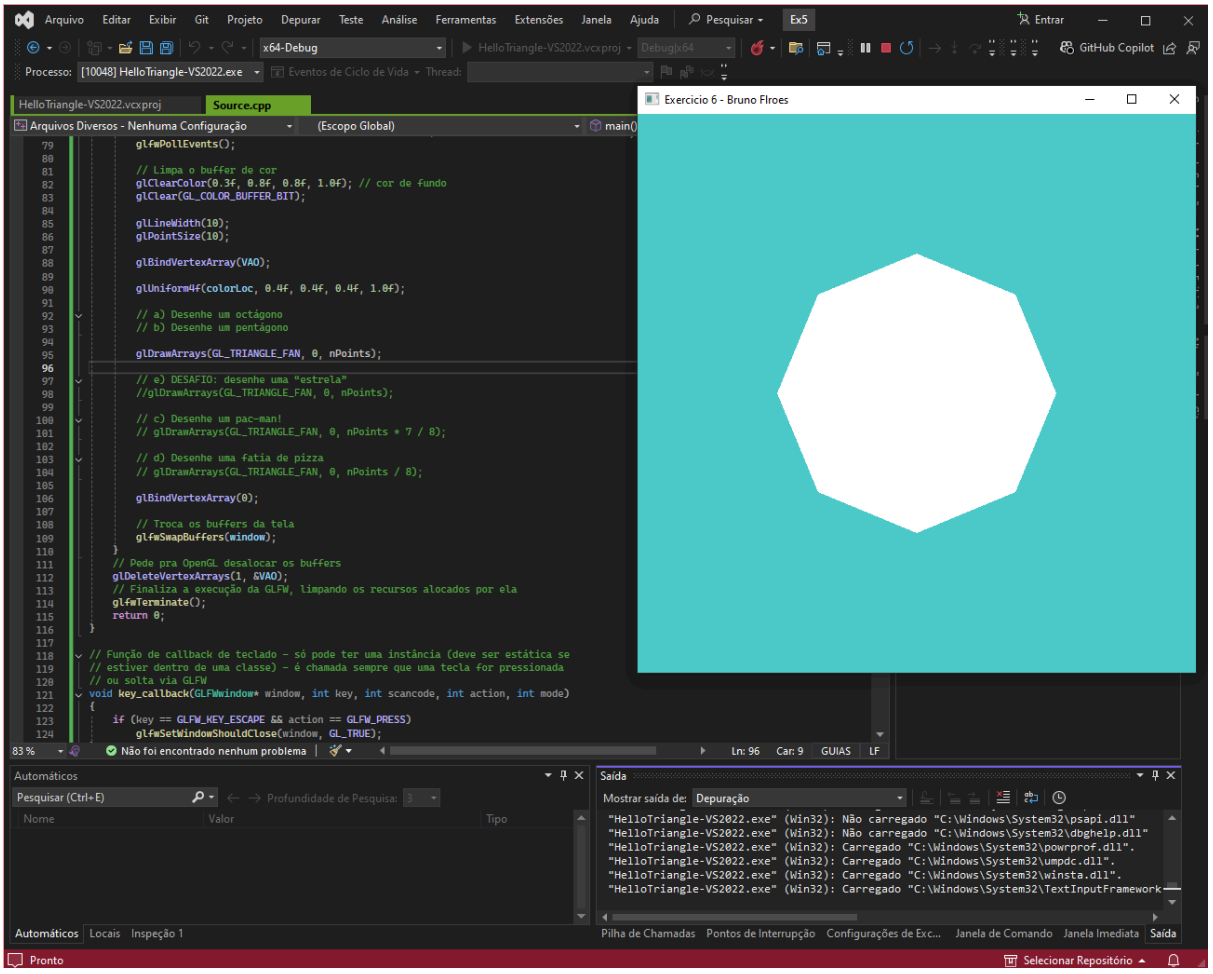
c. Apenas como pontos



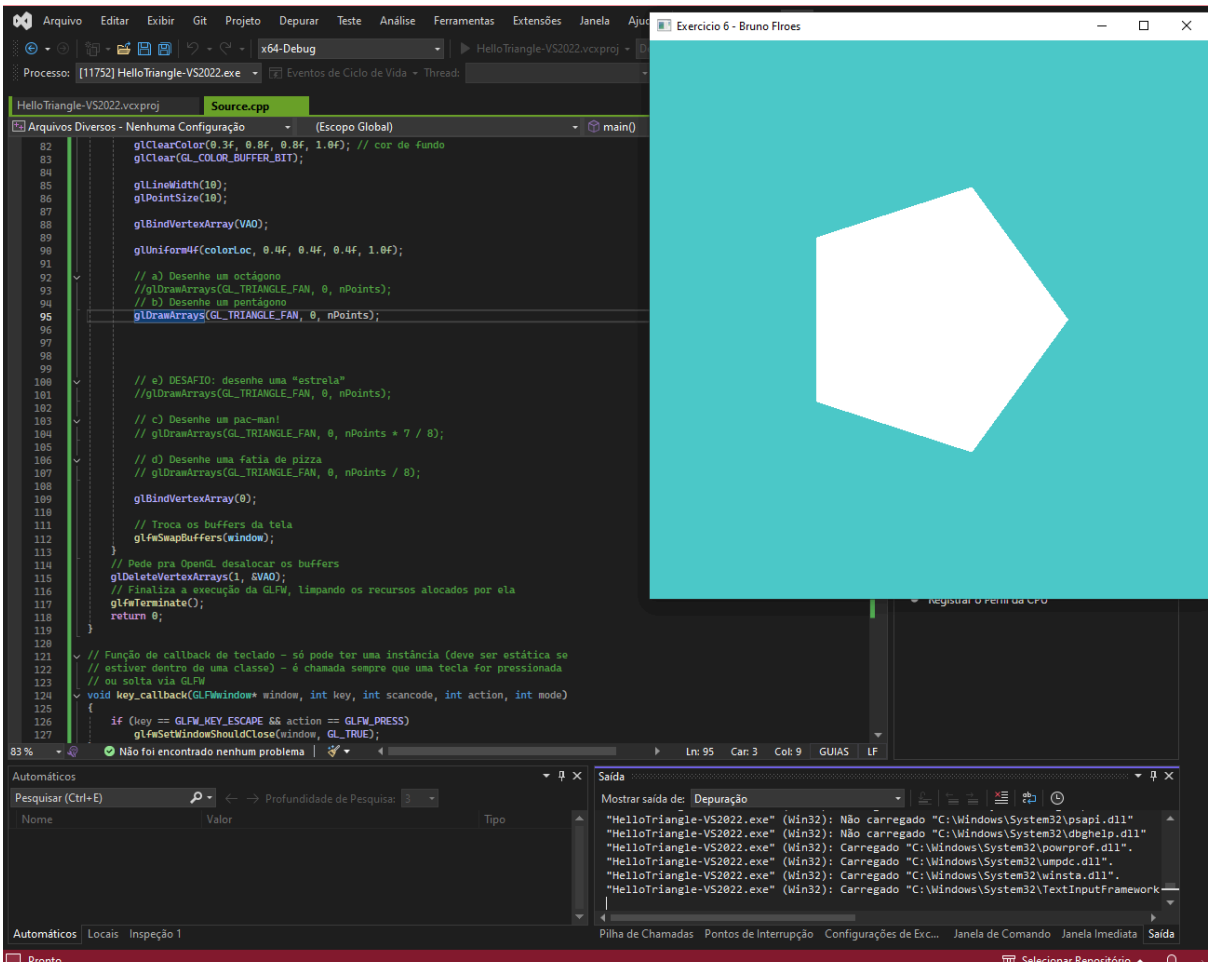
d. Com as 3 formas de desenho juntas



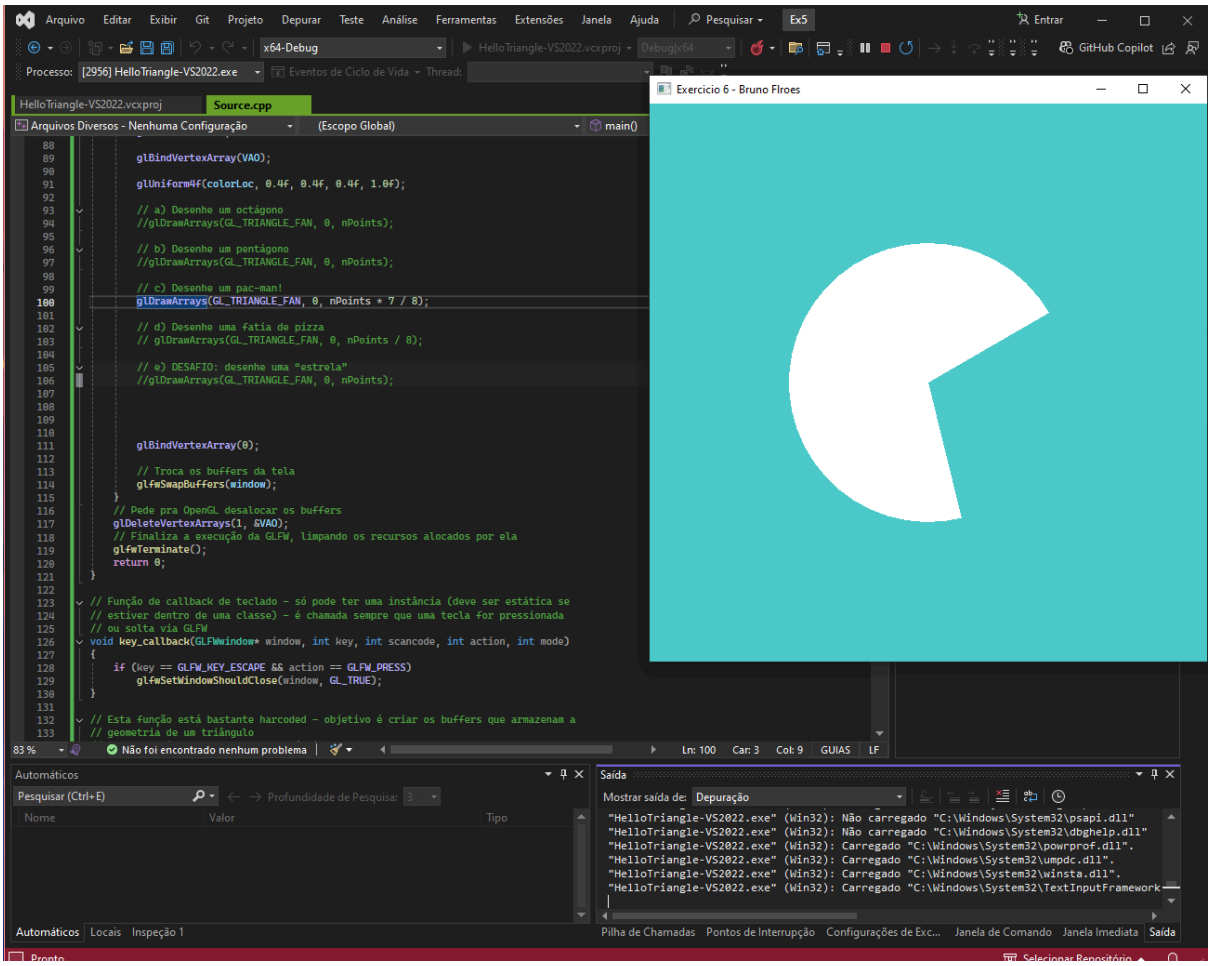
a) Desenhe um octágono



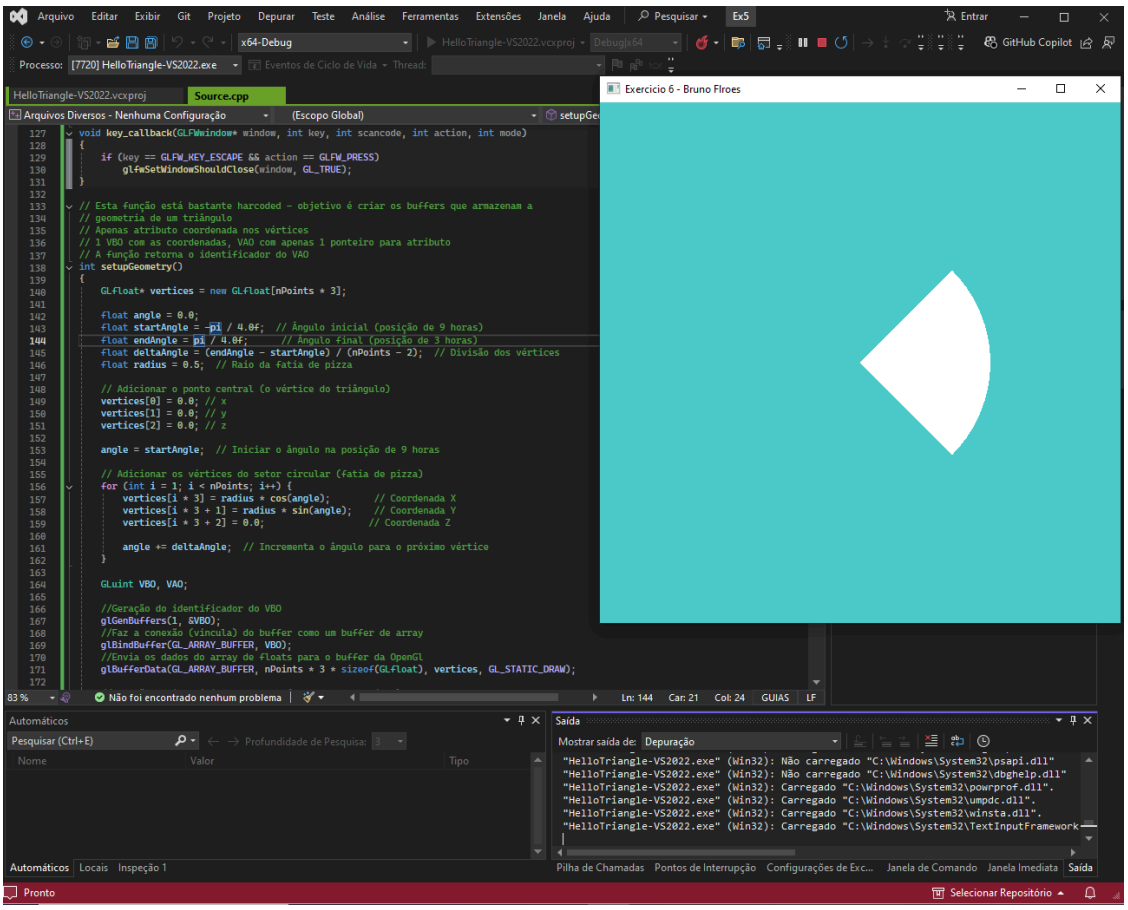
b) Desenhe um pentágono



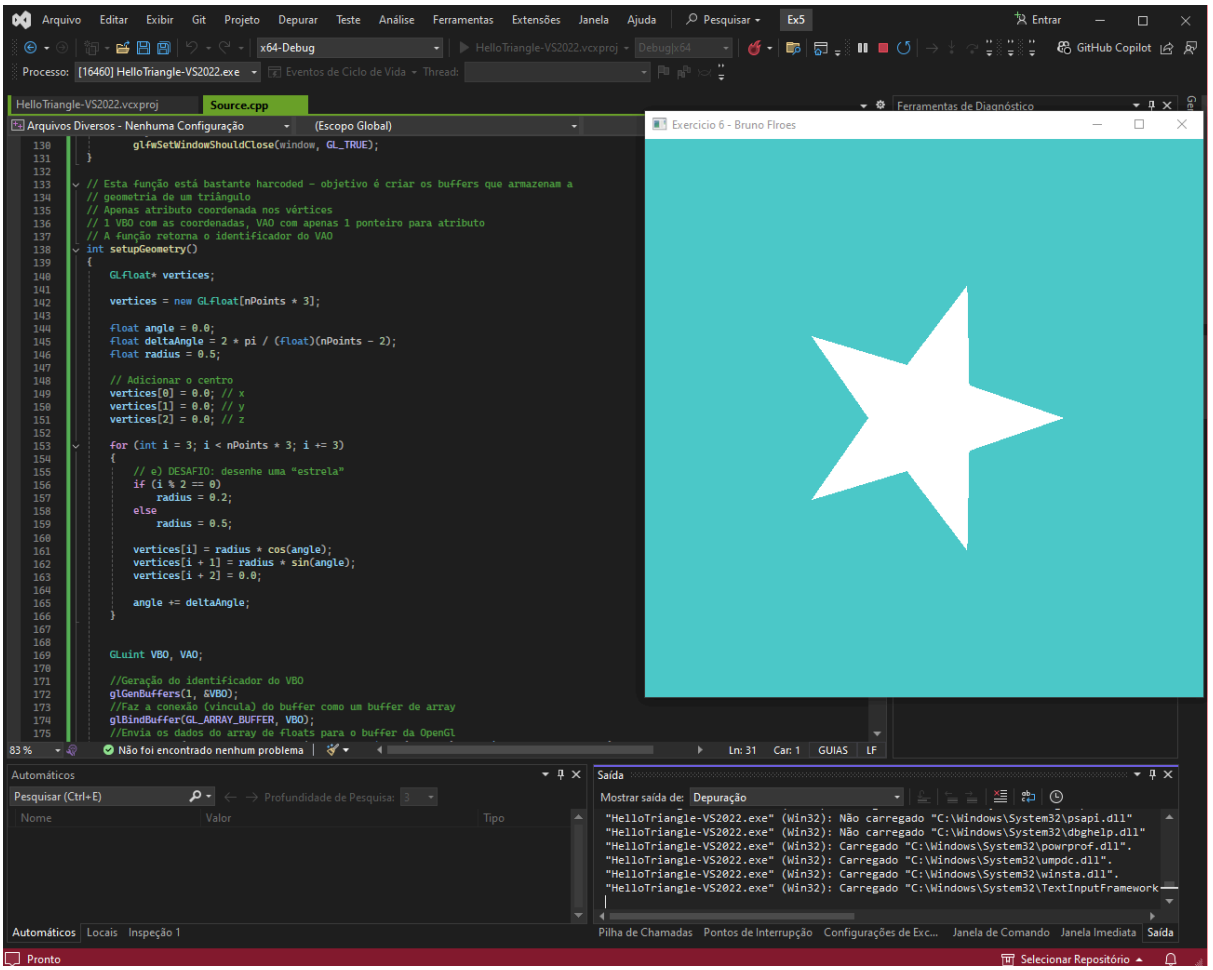
c) Desenhe um pac-man!



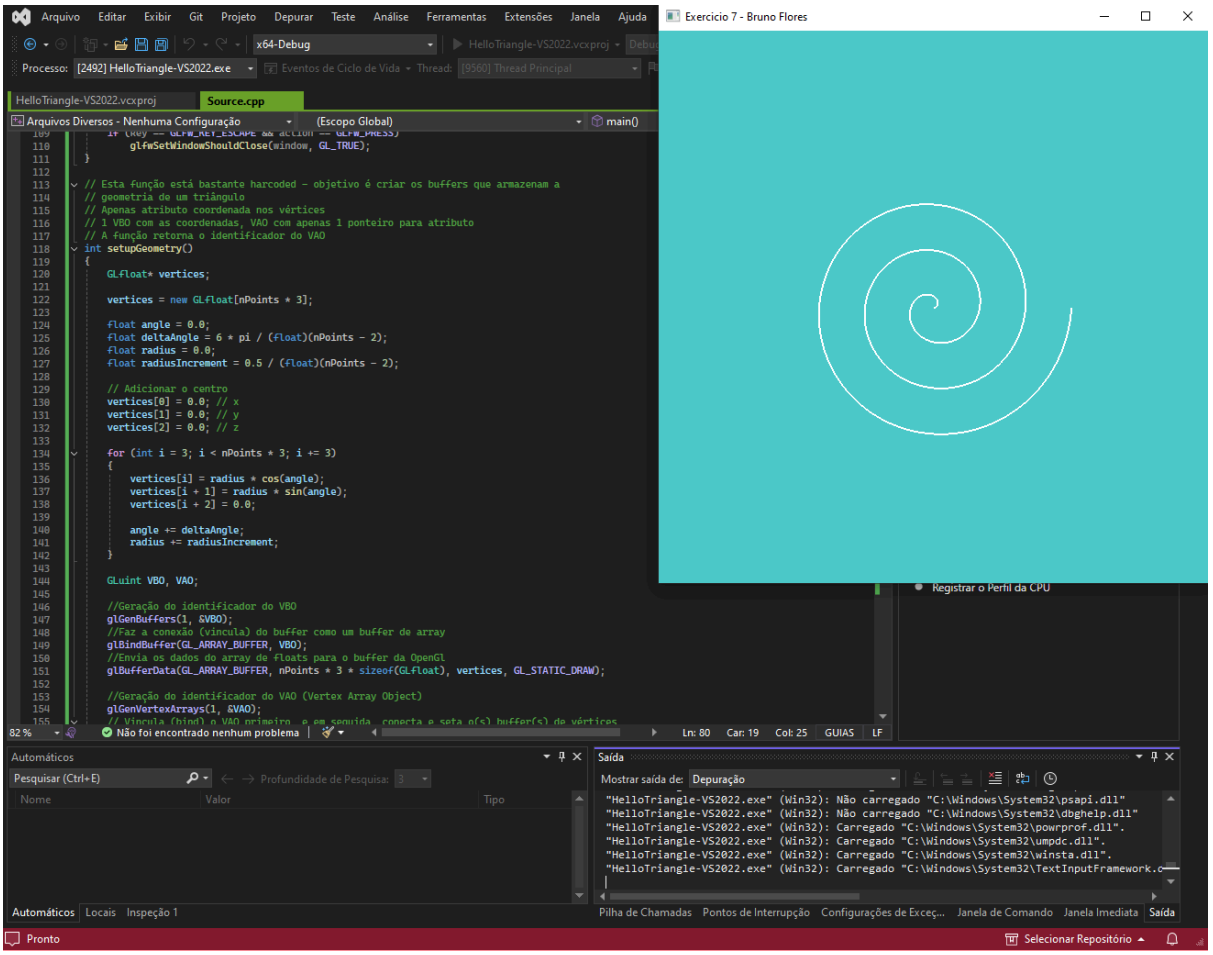
d) Desenhe uma fatia de pizza



e) DESAFIO: desenhe uma “estrela”



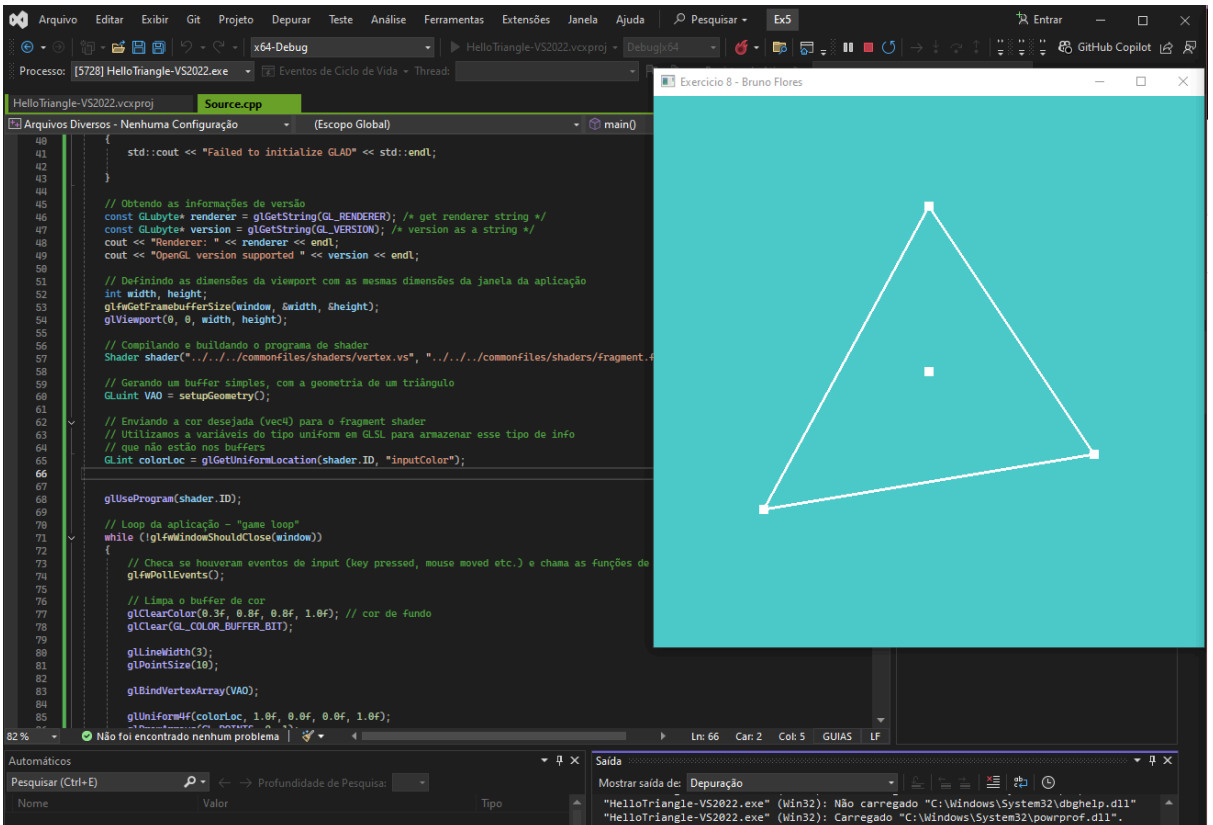
7. Desenhe uma espiral, assim:



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?



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.