Atividades 1 - Processamento Gráfico

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?

   Os shaders obrigatórios são o *vertex* e *fragment shader*, os quais operam a nível de vértices e fragmentos, respectivamente. O *vertex shader* permite definir a cor dos vértices e projeta-los do 3D para 2D. O *fragment shader* foca em cada fragmento ou pixel que será renderizado na tela, permitindo tratar cor, transparência e mais propriedades. Cabe destacar também as etapas de *geometry* e *tesselation shading* que são opcionais no *pipeline* programável. Os *shaders* são escritos utilizando a *OpenGL Shading Language* (GLSL), uma linguagem de *script* fortemente tipada e similar à C.

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

   As primitivas gráficas são as bases para a computação gráfica, como os tijolos são para a construção civil ou átomos que formam moléculas, elas são as formas/unidades básicas utilizadas para representação e construção de objetos. Elas consistem principalmente de linhas, pontos e triângulos, sendo estes utilizados para composição de formas mais complexas. Na OpenGL, os vértices são armazenados em buffers e variam conforme a primitiva utilizada, contendo as informações necessárias.

3. **Explique o que é VBO, VAO e EBO, e como se relacionam.**

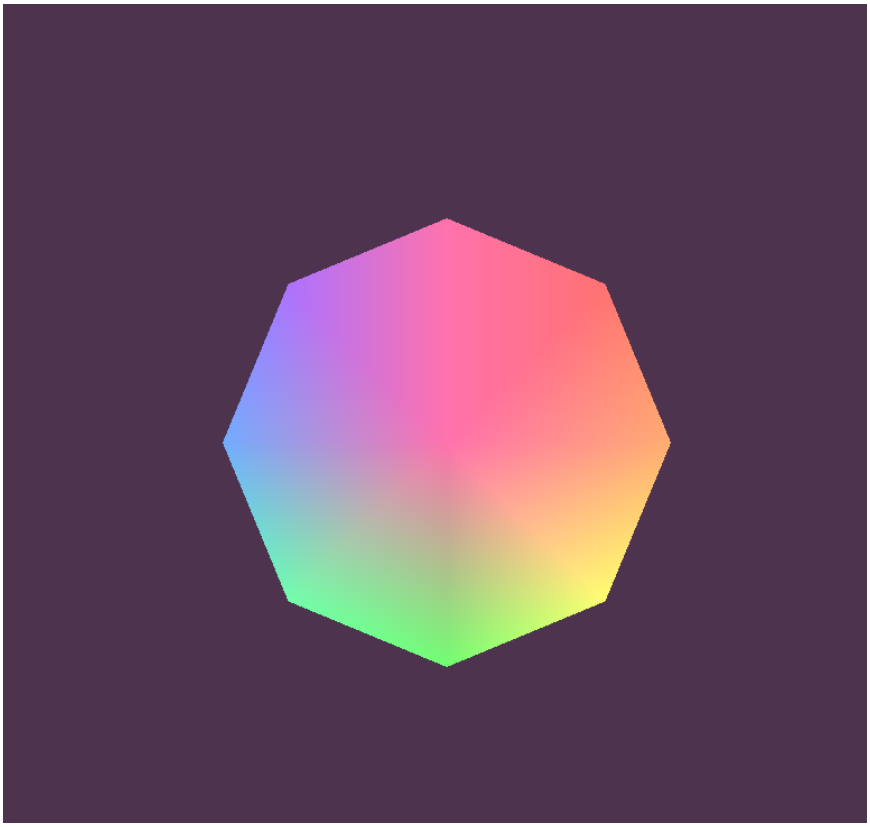
   VBO significa *Vertex Buffer Object* e é um objeto que representa um buffer de memória, cuja função é armazenar informações sobre os vértices. O VAO (*Vertex Array Object*) é um objeto que agrupa um ou mais VBOs contendo as informações para o desenho de um objeto completo. Um VAO pode por exemplo contar com um VBO para as posições dos vértices e outro associado para as suas respectivas cores.

O EBO (*Element Buffer Object)* é utilizado em conjunto com o VAO e VBO e sua função é a definição de elementos, especificando os vértices que os compõem. Ele simplifica o reaproveitamento de memória quando diversos elementos compartilham vértices, logo não sendo necessário duplica-los em memória.

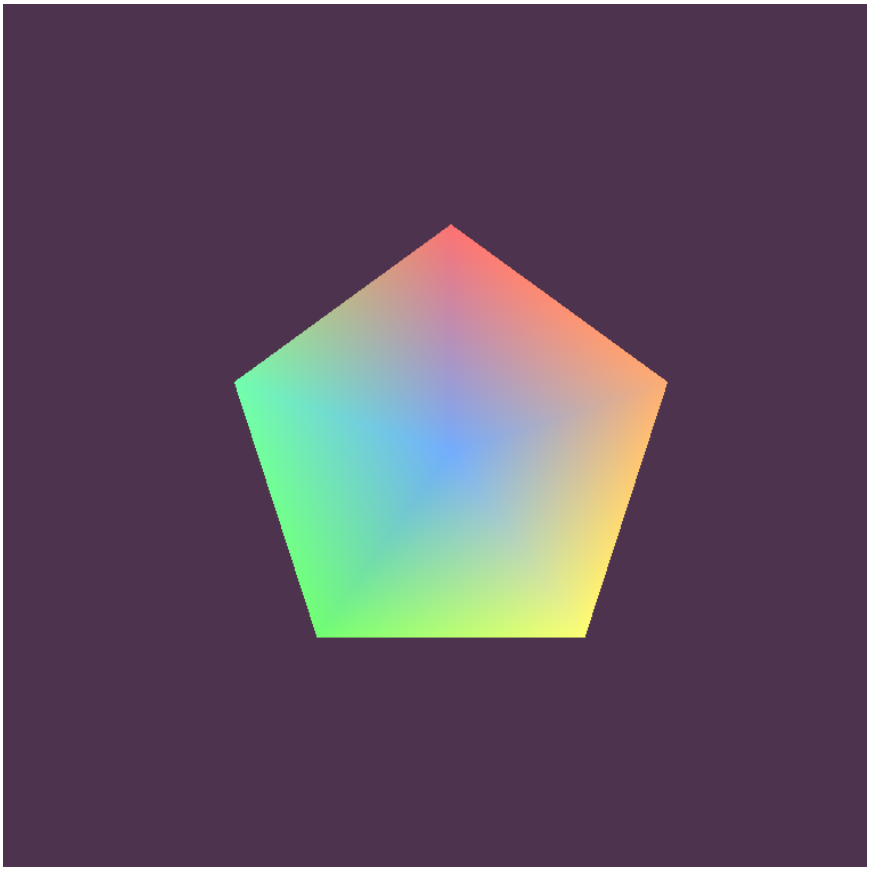
5)

6.

a)

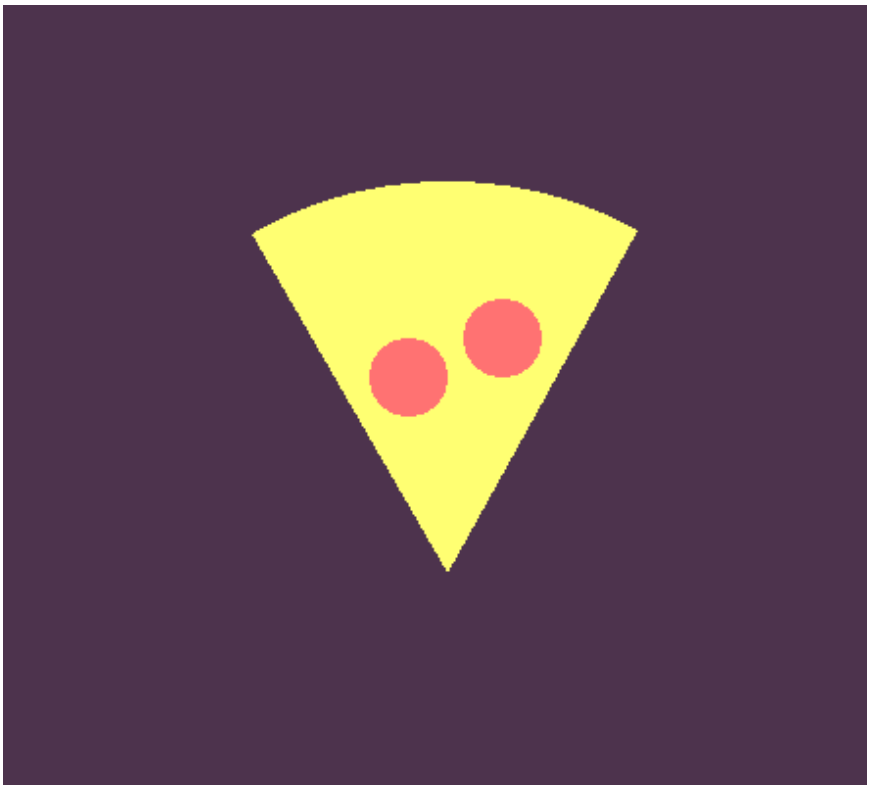


b)



c)

d)



7)

9)