Exercício 2

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

- *Primitivas gráficas* são formas geometricas básicas usadas na renderização. São os elementos mais simples que podem ser desenhados em uma cena servindo como base para a construção de um objeto mais complexo.
- *Armazenar vértices na OpenGL* envolve a criação e manipulacao de buffers de vértice, do qual envia dados dos vértices para GPU, como a posição, vetores normais, cores, etc. Estes dados são alocados diretamente na memória da GPU permitindo que os objetos sejam renderizados pela placa grafica com maior velocidade. Para fazer o armazenamento de vértices na OpenGL é preciso:
- *Definir a Estrutura de Dados dos Vértices* : Define os atributos como posicao, cor, textura... (normalmente é armazenada em um array) ex.:

```
GLfloat vertices[] = { -0.5f, -0.5f, 0.0f,
                                          0.5f, -0.5f, 0.0f,
                                          0.0f, 0.5f, 0.0f };
```

- *Criar um Buffer de Vértice*: É preciso criar um "buffer" para armazenar os dados do vértice. Cada buffer tem um identificados único, para isso, é preciso gerar um ID usando a seguinte função:

```
GLuint vbo;
glGenBuffers(1, &vbo);
```

- *Vincular Buffer Ativo*: Para isso, é preciso fazer um **bind** da seguinte maneira:

```
glBindBuffer (GL_ARRAY_BUFFER, vbo);
```

- *Carregar Dados para o Buffer*: Nesta parte, precisamos preencher o buffer com os dados do vértice, o desenvolvedor ja especificou o tipo de buffer (no passo anterior) e agora precisa dizer os dados que deseja carregar, para isso, é preciso usar a função `glBufferData()` que vai copiar os dados do array para a meoria do buffer:

```
glBufferData(GL_ARRAY_BUFFER, 9 * sizeof(GLfloat), vertices,
GL_STATIC_DRAW);
```

> O ultimo parâmetro especifica como o desenvolvedor quer que a GPU gerencie os dados fornecidos, dito isso, há 3 formas de fazer isso:
>
> - `GL_STATIC_DRAW` - para dados que não vão mudar (ou muito raramente);
> - `GL_DYNAMIC_DRAW` - para dados que mudam com frequência;
> - `GL_STREAM_DRAW` - para dados que vão mudar cada vez que forem desenhados.

- *Especifica o Layout dos Atributos*: É preciso informar a OpenGL como os dados estão organizados no buffer, especificando os atributos dos vertices e a definição do formato desses atributos. Isso pode ser feito da seguinte maneira:

```
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(GLfloat), (GLvoid*)0);
```

> 1. Parâmetro '0': Atributo do qual é linkando, o numero será o mesmo no vertex shader, identificado com a palavra-chave *location*;

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

> 2. Parâmetro '3': Tamanho do atributo (3 valores - xyz);
> 3. Parâmetro 'GL_FLOAT': Tamanho de cada dado;
> 4. Parâmetro 'GL_FALSE': Se os dados precisam ser normalizados, no exemplo usado não precisa;
> 5. Parâmetro '(GLvoid*)0': Deslocamento inicial do buffer, no caso é nenhum. O '0' pode ser substituido por 'NULL'.

> Fontes:
> - IntroduçãoPráticaOpenGL-2-Primitivas.pdf (usp.br)
> - Computação Gráfica (pucrs.br)