

No vertex shader:

```
#version 460

layout (location = 0) in vec3 position;

void main()
{
    gl_Position = vec4(position, 1.0);
}
```

No código cpp:

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

A descrição do layout dos vértices vai conter
1 atributo (na localização zero) **POSIÇÃO** com 3
valores (x, y, z)

No código cpp:

```
GLuint vVBO;
glGenBuffers(1, &vVBO);
glBindBuffer(GL_ARRAY_BUFFER,
vVBO);
glBufferData(GL_ARRAY_BUFFER, 9 *
sizeof(GLfloat), vertices,
GL_STATIC_DRAW);
```

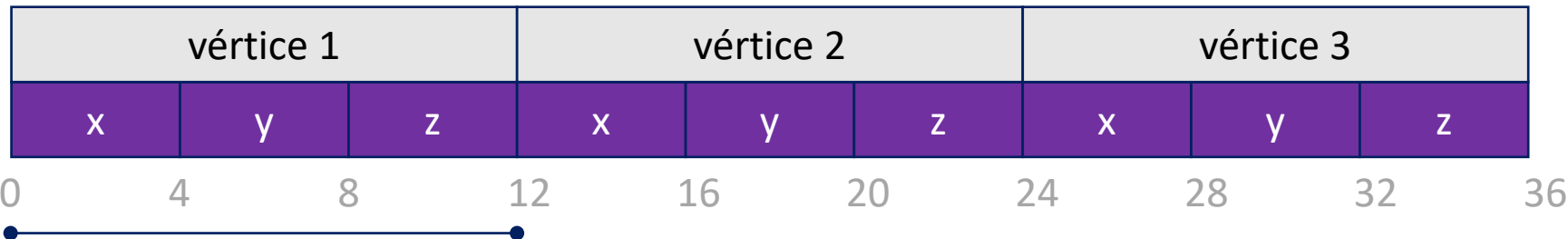
VBO

No código cpp:

```
GLuint VAO;
glGenVertexArrays(1, &VAO);
glBindVertexArray(VAO);

glBindBuffer(GL_ARRAY_BUFFER, vVBO);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(GLfloat), (GLvoid*)0);
glEnableVertexAttribArray(0);
```

VAO



- O atributo possui **3 valores** e ocupa **12 bytes**
- Toda a informação colocada de forma contígua na memória sobre 1 vértice ocupa 12 bytes (**3 * sizeof(GLfloat)**) – 1 atributo só, em 1 array
- Para encontrar este atributo precisamos nos deslocar **0 bytes** (é o primeiro)

No código cpp:

```
GLfloat vertices[] = {
    0.0,0.5,0.0,1.0,0.0,0.0,
    0.5,-0.5,0.0,0.0,1.0,0.0,
    -0.5,-0.5,0.0,0.0,0.0,1.0
};
```

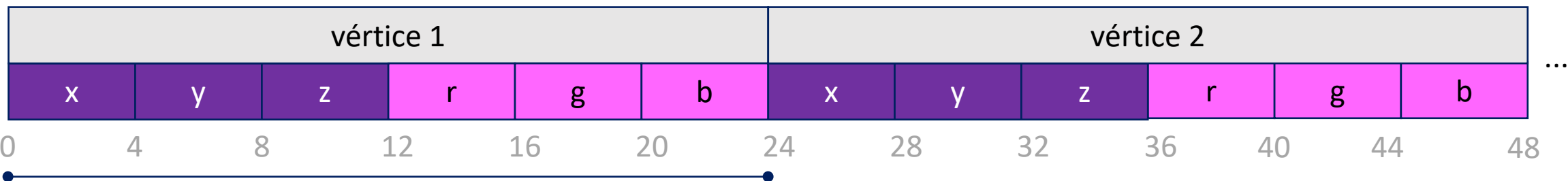
A descrição do layout dos vértices vai conter 2 atributos:

na localização 0, **POSICÃO** com 3 valores (x, y, z)
na localização 1, **COR** com 3 valores (r, g, b)

#version 460

```
layout (location = 0) in vec3 position;
layout (location = 1) in vec3 color;
```

...



- Cada vértice possui 2 atributos com 3 valores, ocupando 24 bytes ($6 * \text{sizeof}(\text{GLfloat})$) de memória de forma contígua
- O atributo **POSICÃO** possui 3 valores e ocupa 12 bytes. Para encontrar este atributo precisamos nos deslocar 0 bytes (é o primeiro)
- O atributo **COR** possui 3 valores e ocupa 12 bytes. Para encontrar este atributo precisamos nos deslocar 12 bytes ($3 * \text{sizeof}(\text{GLfloat})$, na sequência)

```
GLuint 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);
```

VAO

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

VBO

```
GLuint vVBO;
glGenBuffers(1, &vVBO);
glBindBuffer(GL_ARRAY_BUFFER,
    vVBO);
glBufferData(GL_ARRAY_BUFFER, 18 *
    sizeof(GLfloat), vertices,
    GL_STATIC_DRAW);
```


No *vertex* shader:

```
#version 460

layout (location = 0) in vec3 position;
layout (location = 1) in vec3 color;

out vec4 vertexColor;

void main()
{
    gl_Position = vec4(position, 1.0);
    vertexColor = vec4(color, 1.0);
}
```



No *fragment* shader:

```
#version 460

in vec4 vertexColor;
out vec4 color;

void main()
{
    color = vertexColor;
}
```