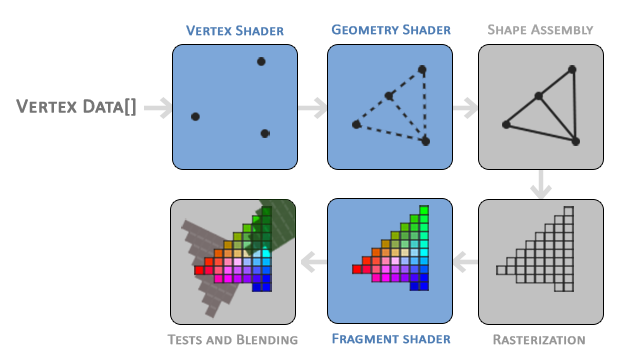
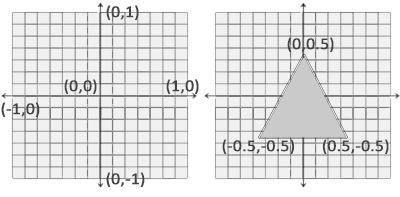
Olá Triangulo



|  |
| --- |
| float[] vertices = {  -0.5f, -0.5f, 0.0f, //Bottom-left vertex  0.0f, 0.5f, 0.0f, //Top vertex  0.5f, -0.5f, 0.0f //Bottom-right vertex  }; |



shader.vert

|  |
| --- |
| #version 330 core  layout (location = 0) in vec3 aPosition;  void main() {  gl\_Position = vec4(aPosition, 1.0);  } |

OnLoad()

|  |
| --- |
| string vertexPath = "../../../Shaders/shader.vert"; |

|  |
| --- |
| string vertexShaderSource = File.ReadAllText(vertexPath); |

|  |
| --- |
| int vertexShader;  vertexShader = GL.CreateShader(ShaderType.VertexShader); |

|  |
| --- |
| GL.ShaderSource(vertexShader, vertexShaderSource);  GL.CompileShader(vertexShader); |

shader.frag

|  |
| --- |
| #version 330 core  out vec4 FragColor;  void main() {  FragColor = vec4(1.0f, 0.5f, 0.2f, 1.0f);  } |

OnLoad()

|  |
| --- |
| string fragmentPath = "../../../Shaders/shader.frag";  string fragmentShaderSource = File.ReadAllText(fragmentPath);  int fragmentShader;  fragmentShader = GL.CreateShader(ShaderType.FragmentShader);  GL.ShaderSource (fragmentShader, fragmentShaderSource);  GL.CompileShader(fragmentShader); |

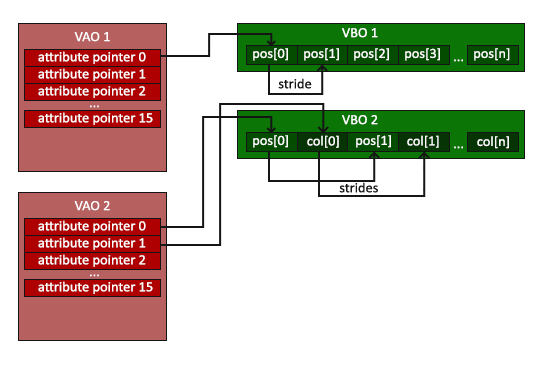
|  |
| --- |
| int shaderProgram; |

OnLoad()

|  |
| --- |
| shaderProgram = GL.CreateProgram(); |

|  |
| --- |
| GL.AttachShader(shaderProgram, vertexShader);  GL.AttachShader(shaderProgram, fragmentShader);  GL.LinkProgram(shaderProgram); |

|  |
| --- |
| GL.DeleteShader(vertexShader);  GL.DeleteShader(fragmentShader); |



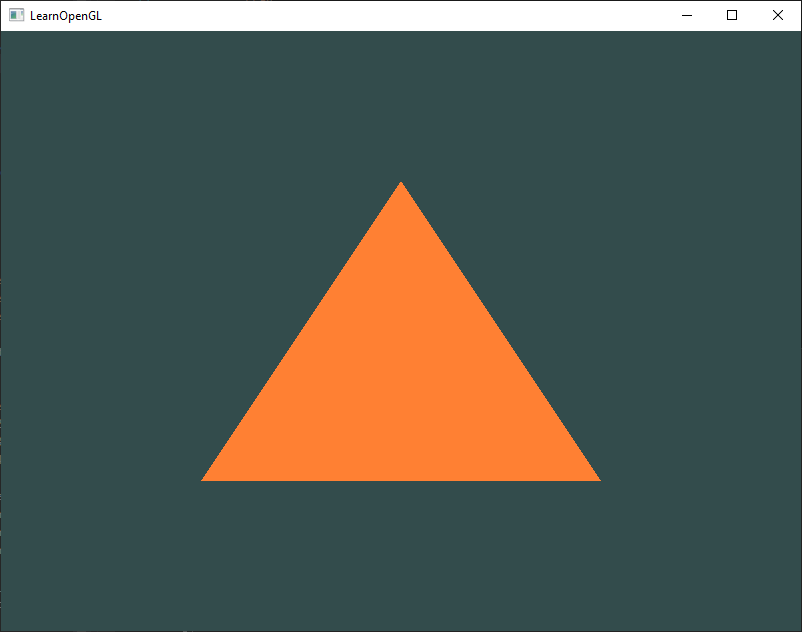
|  |
| --- |
| int VAO; // VertexArrayObject  int VBO; // VertexBufferObject |

OnLoad()

|  |
| --- |
| // ..:: Código de inicialização (feito uma vez (a menos que seu objeto mude frequentemente)) :: ..  // 1. vincular Objeto Vertex Array  VAO = GL.GenVertexArray();  GL.BindVertexArray(VAO);  // 2. copie nosso array de vértices em um buffer para OpenGL usar  VBO = GL.GenBuffer();  GL.BindBuffer(BufferTarget.ArrayBuffer, VBO);  GL.BufferData(BufferTarget.ArrayBuffer, vertices.Length \* sizeof(float), vertices, BufferUsageHint.StaticDraw);  // 3. então defina nossos ponteiros de atributos de vértice  GL.VertexAttribPointer(0, 3, VertexAttribPointerType.Float, false, 3 \* sizeof(float), 0);  GL.EnableVertexAttribArray(0); |

OnRenderFrame

|  |
| --- |
| // ..:: Código de desenho (em loop de renderização) :: ..  // 4. desenhe o objeto  GL.UseProgram(Handle);  GL.BindVertexArray(VAO);  GL.DrawArrays(PrimitiveType.Triangles, 0, 3); |



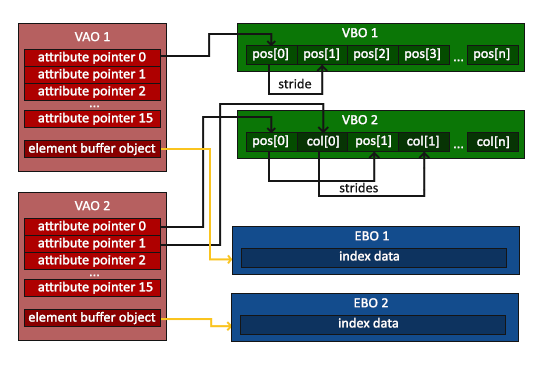
Element Buffer Objects

|  |
| --- |
| Float[] vertices = {  // first triangle  -0.5f, -0.5f, 0.0f, // bottom left  -0.5f, 0.5f, 0.0f, // top left  0.5f, 0.5f, 0.0f, // top right  // second triangle  -0.5f, -0.5f, 0.0f, // bottom left  0.5f, 0.5f, 0.0f, // top right  0.5f, -0.5f, 0.0f, // bottom right  }; |

|  |
| --- |
| GL.DrawArrays(PrimitiveType.Triangles, 0, 6); |

Mas não vamos fazer isso.

|  |
| --- |
| float[] vertices = {  -0.5f, -0.5f, 0.0f, // bottom left  -0.5f, 0.5f, 0.0f, // top left  0.5f, 0.5f, 0.0f, // top right  0.5f, -0.5f, 0.0f // bottom right  };  int[] indices = { // note that we start from 0!  0, 1, 2, // first triangle  0, 2, 3 // second triangle  }; |



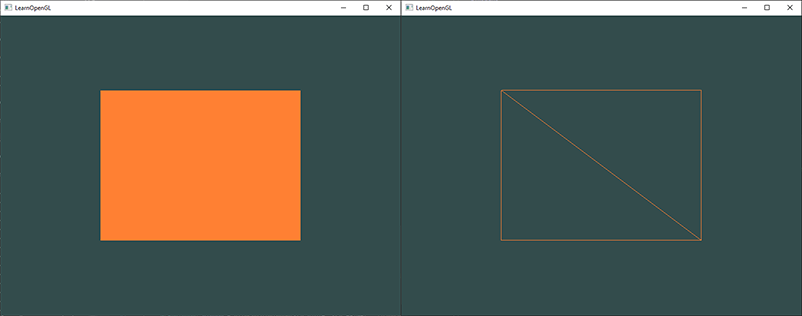
|  |
| --- |
| int EBO; // Element Buffer Object |

OnLoad()

|  |
| --- |
| EBO = GL.GenBuffer(); |

|  |
| --- |
| GL.BindBuffer(BufferTarget.ElementArrayBuffer, EBO);  GL.BufferData(BufferTarget.ElementArrayBuffer, indices.Length \* sizeof(float), indices, BufferUsageHint.StaticDraw); |

|  |
| --- |
| ~~GL.DrawArrays(PrimitiveType.Triangles, 0, 3);~~  GL.DrawElements(PrimitiveType.Triangles, indices.Length, DrawElementsType.UnsignedInt, 0); |



Wireframe mode

OnUpdateFrame()

|  |
| --- |
| if(KeyboardState.IsKeyPressed(Keys.Up)) {  GL.PolygonMode(MaterialFace.FrontAndBack, PolygonMode.Line);  }  if(KeyboardState.IsKeyPressed(Keys.Down)) {  GL.PolygonMode(MaterialFace.FrontAndBack, PolygonMode.Fill);  } |