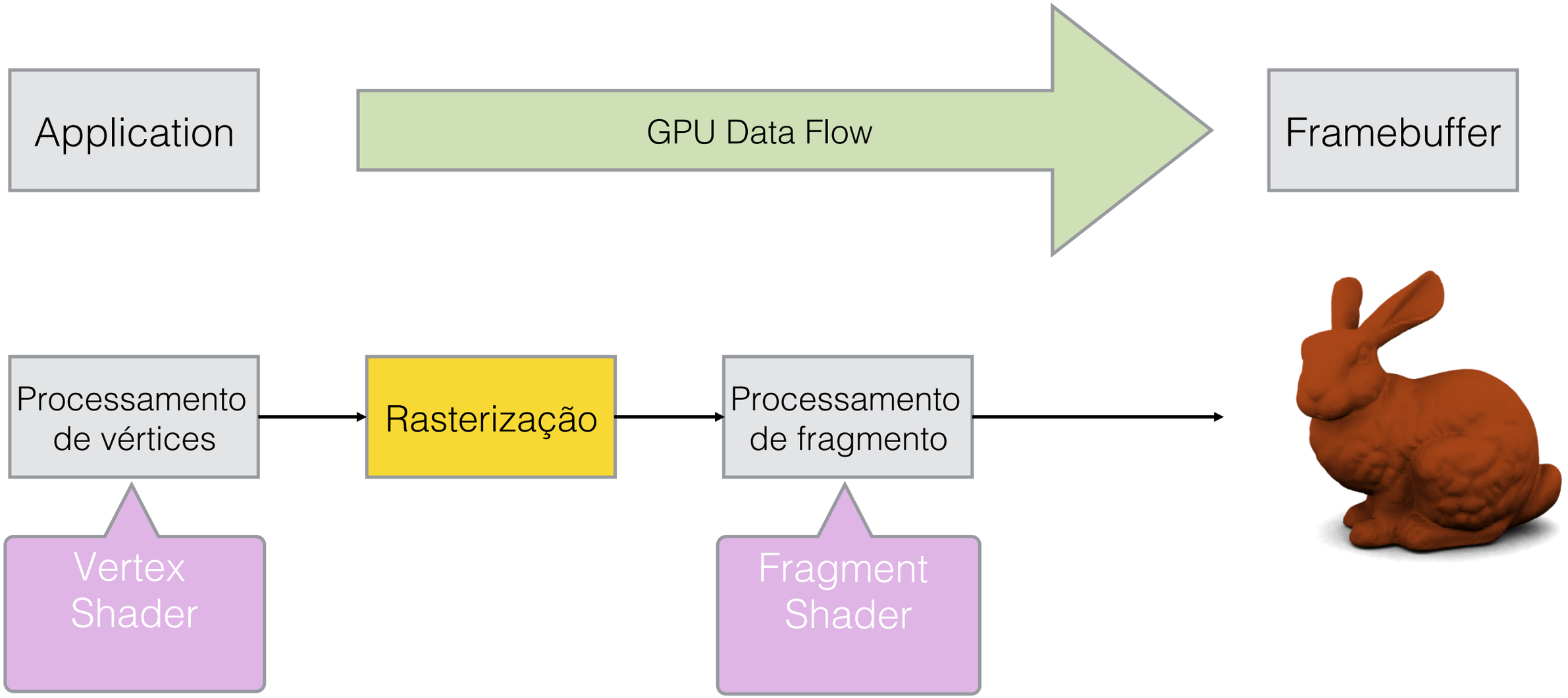
Computação Gráfica

TCC-00291

Assunto: Primeiros passos

Visão geral

Pipeline simplificado

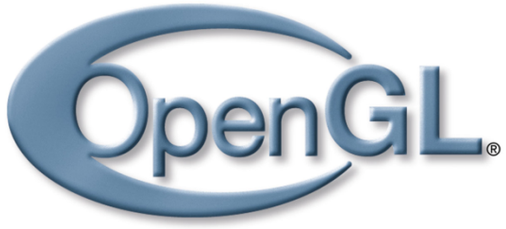


Visão geral

Programa básico

Escrever programas OpenGL modernos consiste em, essencialmente:

- Escrever shaders;
- Criar buffers e carregar dados;
- “Conectar” dados e variáveis dos shaders; e
- Renderizar.



Visão geral

Programa básico

Uma aplicação OpenGL precisa de um lugar para *renderizar* a informação processada.

Visão geral

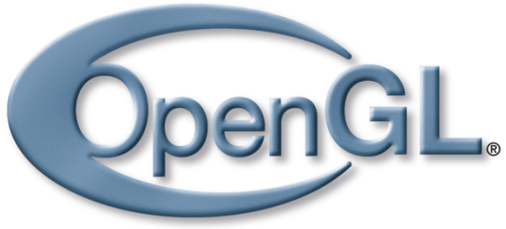
Programa básico

Uma aplicação OpenGL precisa de um lugar para *renderizar* a informação processada.

Normalmente, utiliza-se uma janela do sistema.

- Necessidade de comunicação com a API do sistema de janelas.
- Cada sistema de janela tem uma interface diferente...
- Opções: GLFW (C++), LWJGL(Java), ...

Obs: Veremos exemplos em Java adiante.



Visão geral

Representação de geometria

Objetos geométricos são representados através de ***vértices***.

Visão geral

Representação de geometria

Objetos geométricos são representados através de ***vértices***.

Def.: Um vértice é uma coleção de atributos.

- Coordenadas no espaço;
- Cores;
- Coordenadas de textura;
- Vetor normal; e
- ...

Visão geral

Representação de geometria

Objetos geométricos são representados através de ***vértices***.

Def.: Um vértice é uma coleção de atributos.

- Coordenadas no espaço;
- Cores;
- Coordenadas de textura;
- Vetor normal; e
- ...

Obs.: As coordenadas do ponto são armazenadas usando 4 coordenadas. ***(Voltaremos a este assunto!!)***

Visão geral

Representação de geometria

Os dados relacionados aos vértices devem ser armazenados em arrays, chamados de *“vertex buffer objects”, ou VBOs*.

Visão geral

Representação de geometria

Os dados relacionados aos vértices devem ser armazenados em arrays, chamados de *“vertex buffer objects”, ou VBOs*.

VBOs devem ser organizados em arrays, chamados de *“vertex array objects”, ou VAOs*.

Visão geral

Representação de geometria

Os dados relacionados aos vértices devem ser armazenados em arrays, chamados de *"vertex buffer objects", ou VBOs*.

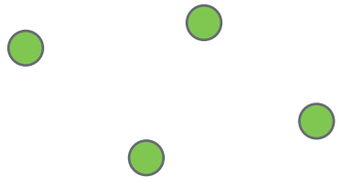
VBOs devem ser organizados em arrays, chamados de *"vertex array objects", ou VAOs*.

As informações contidas nos VBOs devem ser organizadas de acordo com as *primitivas geométricas* que representam.

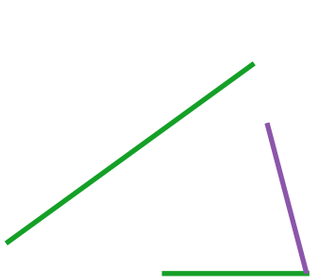
Visão geral

Representação de geometria

GL_POINTS

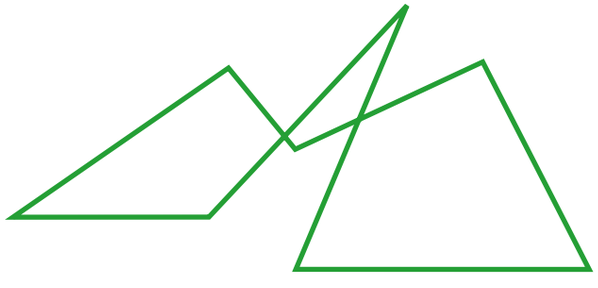


GL_LINES

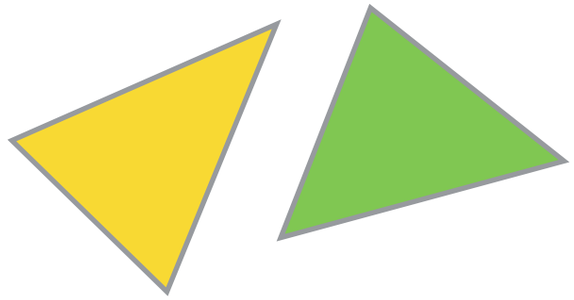


GL_LINE_STRIP

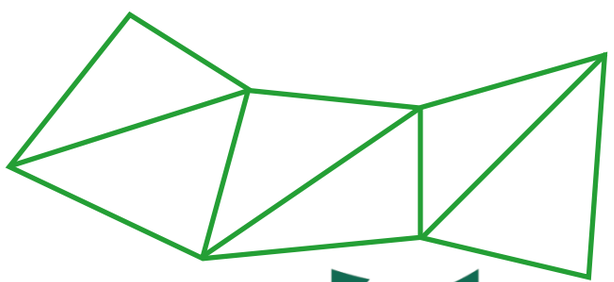
GL_LINE_LOOP



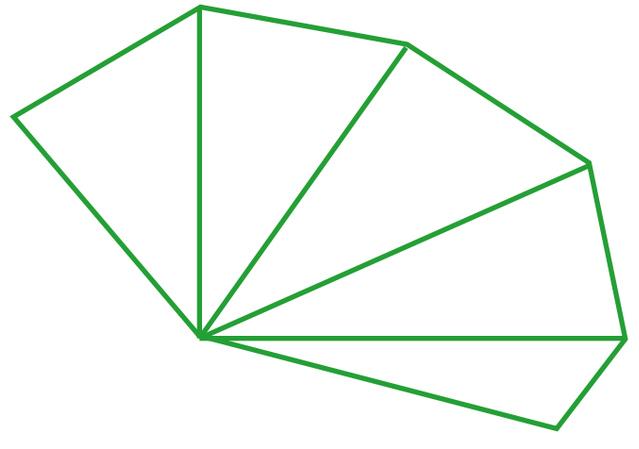
GL_TRIANGLES



GL_TRIANGLE_STRIP



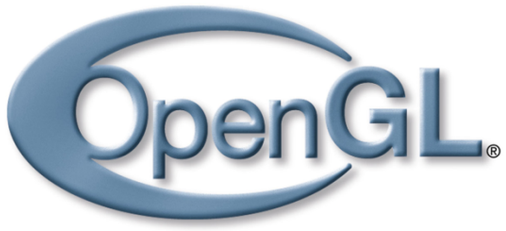
GL_TRIANGLE_FAN



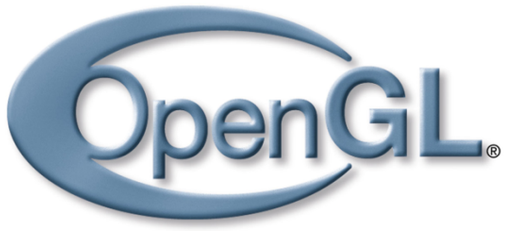
Visão geral

Representação de geometria

Primitiva	Descrição	#verts
GL_POINTS	Renderiza um ponto por vértice.	n
GL_LINES	Renderiza uma linha para cada par de vértice.	2n
GL_LINE_STRIP	Renderiza uma linha para cada novo vértice, conectando com o anterior.	n+1
GL_LINE_LOOP	Renderiza todos os vértices em um loop de segmentos.	n
GL_TRIANGLES	Renderiza um triângulo para cada trio de vértices.	3n
GL_TRIANGLE_STRIP	Renderiza um triângulo para cada novo vértice, conectado com os dois anteriores.	n+2
GL_TRIANGLE_FAN	Renderiza um triângulo para cada novo par de vértices, usando o vértice inicial.	n+2



Hello World!



Hello World

Desenhando um triângulo

Desenharemos um triângulo com cores distintas em seus vértices.

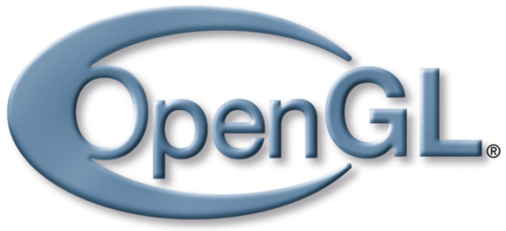
Hello World

Desenhando um triângulo

Desenharemos um triângulo com cores distintas em seus vértices.

Objetivos:

- Manipular dados de vértices;
- Organizar dados para serem renderizados; e
- Construir primitivas geométricas



Hello World

Desenhando um triângulo

Primeiro passo:

Determinar a quantidade de memória necessária.

Hello World

Desenhando um triângulo

Primeiro passo:

Determinar a quantidade de memória necessária.

Hello World

Desenhando um triângulo

Primeiro passo:

Determinar a quantidade de memória necessária.

Cores:
1 triângulo x
3 vértices por triângulo x
4 canais por vértice

RGBA

Hello World

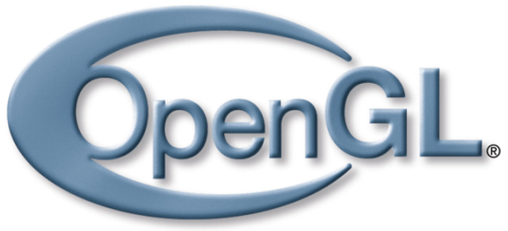
Desenhando um triângulo

Segundo passo:

Antes de inicializar os VBOs, precisamos criar os dados

```
// creation of coordinates
float[] positionData = new float[]{
    0.0f, 0.0f, 0.0f, 1.0f,
   -0.5f, 0.0f, 0.0f, 1.0f,
    0.0f, 0.5f, 0.0f, 1.0f
};

// creation of colors
float[] colorData = new float[]{
    0.0f, 0.0f, 1.0f, 1.0f,
    1.0f, 0.0f, 0.0f, 1.0f,
    0.0f, 1.0f, 0.0f, 1.0f
};
```



Hello World

Desenhando um triângulo

Terceiro passo:

Criação e inicialização dos VBOs

```
// copy the positions array to the buffer
FloatBuffer positionBuffer = BufferUtils.createFloatBuffer(positionData.length);
positionBuffer.put(positionData);
positionBuffer.flip();

// copy the colors array to buffer
FloatBuffer colorBuffer = BufferUtils.createFloatBuffer(colorData.length);
colorBuffer.put(colorData);
colorBuffer.flip();
```

Hello World

Desenhando um triângulo

Terceiro passo:

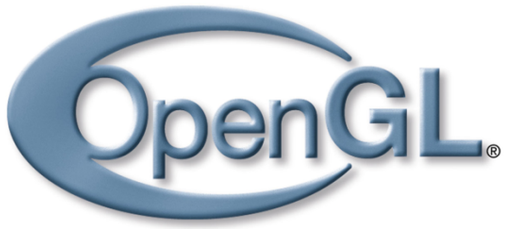
Criação e inicialização dos VBOs

```
// copy the positions array to the buffer
FloatBuffer positionBuffer = BufferUtils.createFloatBuffer(positionData.length);
positionBuffer.put(positionData);
positionBuffer.flip();

// copy the colors array to buffer
FloatBuffer colorBuffer = BufferUtils.createFloatBuffer(colorData.length);
colorBuffer.put(colorData);
colorBuffer.flip();
```

Qual a função do método flip()?

http://download.java.net/jdk7/archive/b123/docs/api/java/nio/Buffer.html



Hello World

Desenhando um triângulo

Terceiro passo:

Criação e inicialização dos VBOs

```
// creation of the vertex byffer object (VBO) for vertices
int positionBufferHandle = GL15.glGenBuffers();
GL15.glBindBuffer(GL15.GL_ARRAY_BUFFER, positionBufferHandle);
GL15.glBufferData(GL15.GL_ARRAY_BUFFER, positionBuffer, GL15.GL_STATIC_DRAW);

// creation of the VBO for color values
int colorBufferHandle = GL15.glGenBuffers();
GL15.glBindBuffer(GL15.GL_ARRAY_BUFFER, colorBufferHandle);
GL15.glBufferData(GL15.GL_ARRAY_BUFFER, colorBuffer, GL15.GL_STATIC_DRAW);
```

Hello World

Desenhando um triângulo

Terceiro passo:

Criação e inicialização dos VBOs

```
// creation of the vertex byffer object (VBO) for vertices
int positionBufferHandle = GL15.glGenBuffers();
GL15.glBindBuffer(GL15.GL_ARRAY_BUFFER, positionBufferHandle);
GL15.glBufferData(GL15.GL_ARRAY_BUFFER, positionBuffer, GL15.GL_STATIC_DRAW);

// creation of the VBO for color values
int colorBufferHandle = GL15.glGenBuffers();
GL15.glBindBuffer(GL15.GL_ARRAY_BUFFER, colorBufferHandle);
GL15.glBufferData(GL15.GL_ARRAY_BUFFER, colorBuffer, GL15.GL_STATIC_DRAW);
```

glBindBuffer ?

http://www.opengl.org/sdk/docs/man/xhtml/glBindBuffer.xml

Hello World

Desenhando um triângulo

Terceiro passo:

Criação e inicialização dos VBOs

```
// creation of the vertex byffer object (VBO) for vertices
int positionBufferHandle = GL15.glGenBuffers();
GL15.glBindBuffer(GL15.GL_ARRAY_BUFFER, positionBufferHandle);
GL15.glBufferData(GL15.GL_ARRAY_BUFFER, positionBuffer, GL15.GL_STATIC_DRAW);

// creation of the VBO for color values
int colorBufferHandle = GL15.glGenBuffers();
GL15.glBindBuffer(GL15.GL_ARRAY_BUFFER, colorBufferHandle);
GL15.glBufferData(GL15.GL_ARRAY_BUFFER, colorBuffer, GL15.GL_STATIC_DRAW);
```

glBufferData ?

https://www.opengl.org/sdk/docs/man4/xhtml/glBufferData.xml

Hello World

Desenhando um triângulo

Quarto passo:

Criação e inicialização dos VAOs

```
// creation of the vertex array object (VAO)
int vaoHandle = GL30.glGenVertexArrays();
GL30.glBindVertexArray(vaoHandle);
GL20.glEnableVertexAttribArray(0);
GL20.glEnableVertexAttribArray(1);

// assignment of the position VBO to slot 0 of VAO
GL15.glBindBuffer(GL15.GL_ARRAY_BUFFER, positionBufferHandle);
GL20.glVertexAttribPointer(0, 4, GL11.GL_FLOAT, false, 0, 0);

// assignment of the color VBO to slot 1 of VAO
GL15.glBindBuffer(GL15.GL_ARRAY_BUFFER, colorBufferHandle);
GL20.glVertexAttribPointer(1, 4, GL11.GL_FLOAT, false, 0, 0);
```

Exercício: consultar a documentação dos métodos.

Hello World

Desenhando um triângulo

Quinto passo:

Renderização dos VAOs.

```
// draw VAO
GL11.glDrawArrays(GL11.GL_TRIANGLES, 0, 3);
```

Obs:

- Deve ser chamado depois da definição dos *shaders.*
- É normalmente invocado no *callback de display.*

Hello World

Desenhando um triângulo

Quinto passo:

Renderização dos VAOs.

```
// draw VAO
GL11.glDrawArrays(GL11.GL_TRIANGLES, 0, 3);
```

Obs:

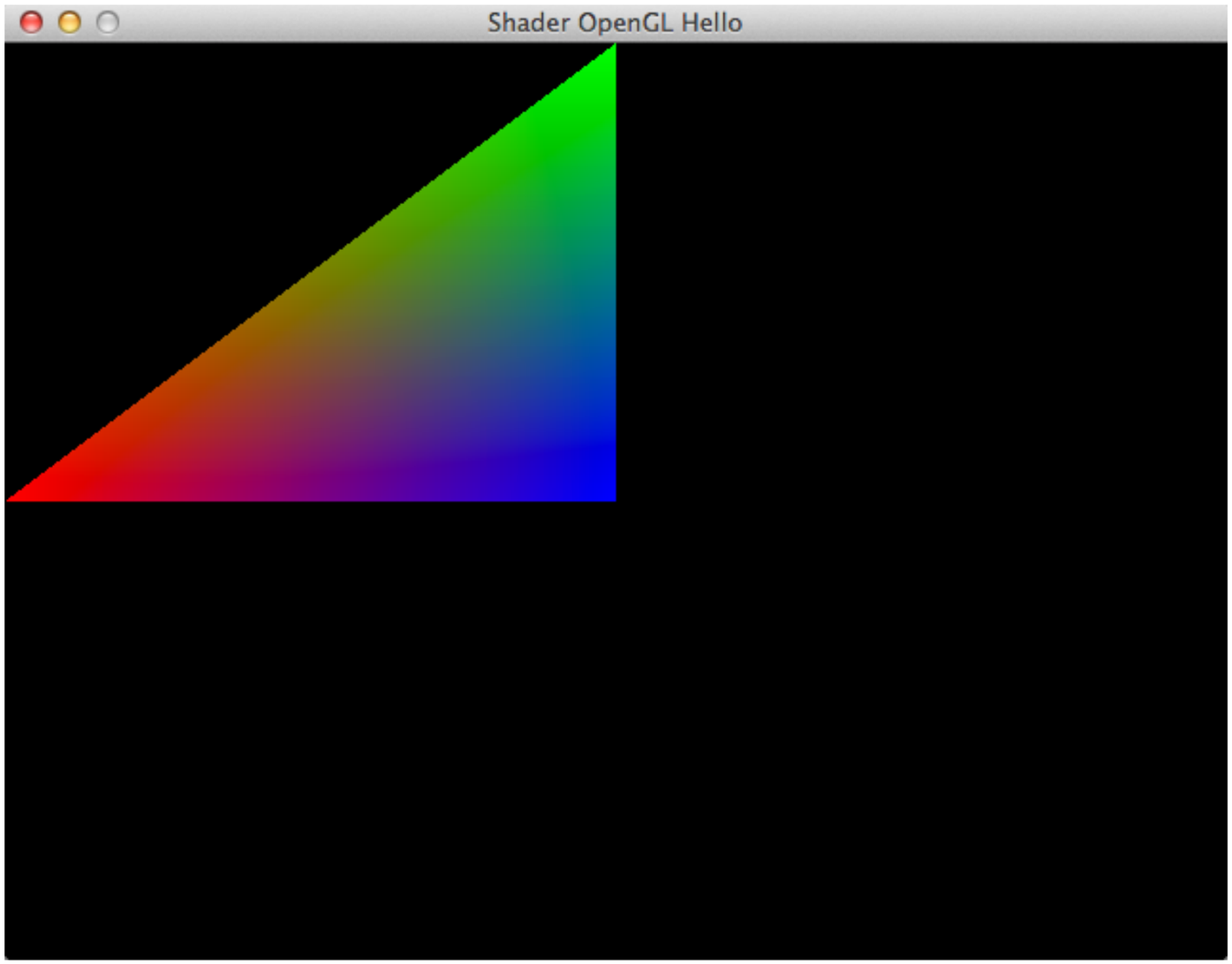
- Deve ser chamado depois da definição dos ***shaders.***
- É normalmente invocado no ***callback de display.***

glDrawArrays?

http://www.opengl.org/sdk/docs/man4/xhtml/glDrawArrays.xml

Hello World

Desenhando um triângulo



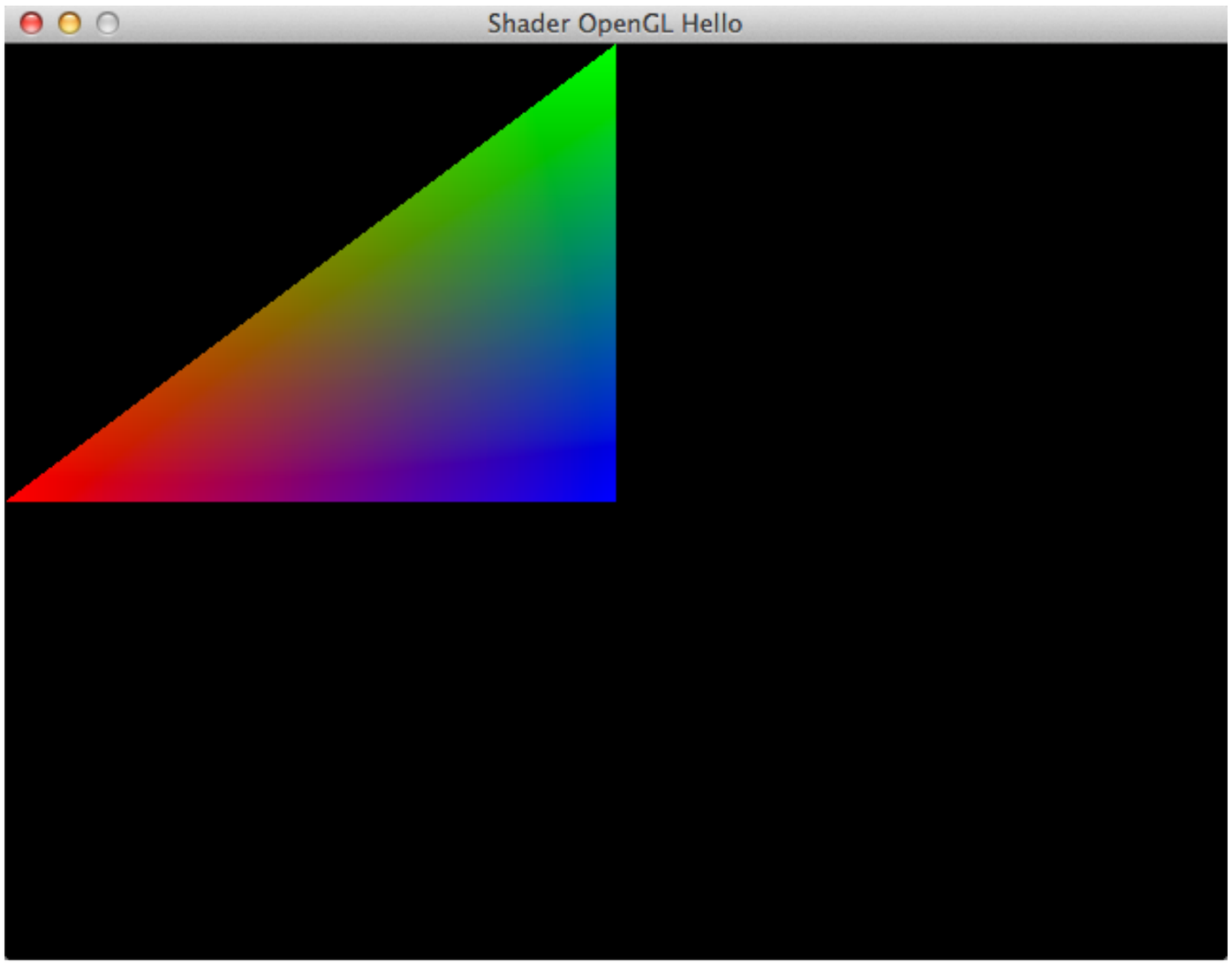
Escondemos parte do código em uma classe auxiliar:

`ShaderProgram.java`

Em breve, veremos seu conteúdo.

Hello World

Desenhando um triângulo



Escondemos parte do código em uma classe auxiliar:

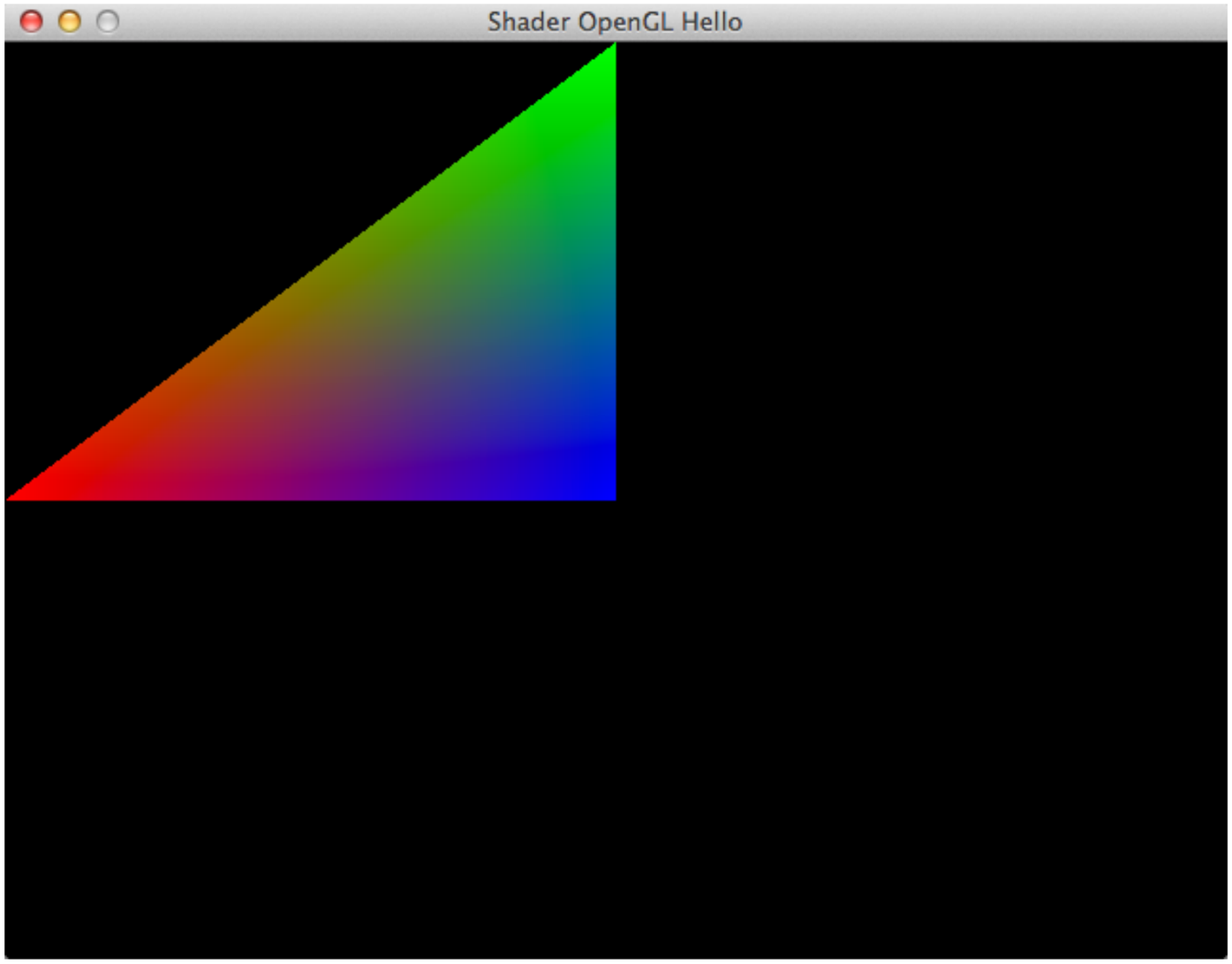
`ShaderProgram.java`

Em breve, veremos seu conteúdo.

Código completo:
Ver página do curso.

Hello World

Desenhando um triângulo



Códigos OpenGL escritos em outras linguagens são análogos.

C++:

http://openglbook.com/the-book/chapter-2-vertices-and-shapes/

Python:

http://pyopengl.sourceforge.net/documentation/index.html

JavaScript:

http://www.khronos.org/webgl/wiki/Main_Page

Computação Gráfica

TCC-00291

Assunto: Primeiros passos