

# Computação Gráfica e Interfaces

2016-2017  
Fernando Birra

# Introdução ao WebGL

2016-2017

Fernando Birra

# Objetivos

- Componentes duma aplicação WebGL
- Convenções a usar (bibliotecas auxiliares e organização em ficheiros)
- Familiarização com *shaders*
- Familiarização com primitivas gráficas e sua invocação

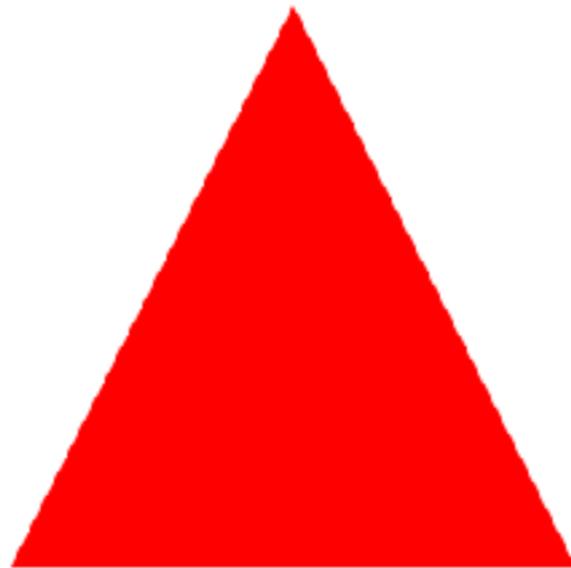
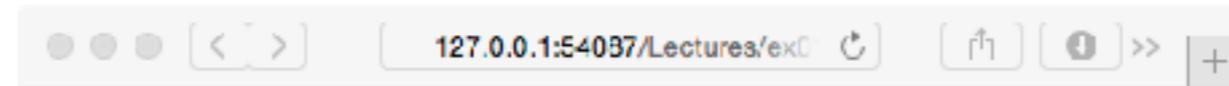
# Exemplo

- Uma aplicação WebGL tem duas componentes obrigatórias:
  - HTML + Javascript
- **HTML:**
  - descreve a página
  - poderá incluir código de bibliotecas
  - no nosso caso, inclui ainda os shaders
- **Javascript:**
  - contém o código da aplicação + código de bibliotecas

# Vantagens em usar WebGL

- A aplicação corre num browser
  - dispensa instalação de ferramentas (compilador, bibliotecas)
  - é independente do sistema operativo
  - funciona em todos os browsers recentes
- O código é escrito em Javascript
- Obriga à utilização de shaders
- A API é bastante mais simples do que a do OpenGL

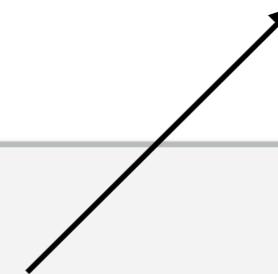
# Exemplo: triangle.html



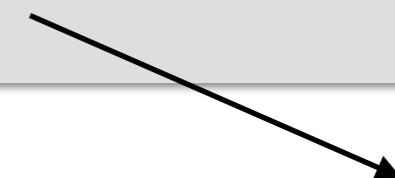
# HTML

```
<!DOCTYPE html>
<html>
<head>
<script id="vertex-shader" type="x-shader/x-vertex">
attribute vec4 vPosition;
void main(){
    gl_Position = vPosition;
}
</script>
<script id="fragment-shader" type="x-shader/x-fragment">
precision mediump float;
void main() {
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
}
</script>
<script type="text/javascript" src="../Common/webgl-utils.js"></script>
<script type="text/javascript" src="../Common/initShaders.js"></script>
<script type="text/javascript" src="../Common/MV.js"></script>
<script type="text/javascript" src="triangle.js"></script>
</head>
<body>
    <title>Triangle</title>
    <canvas id="gl-canvas" width="512" height="512">
        Oops... your browser doesn't support the HTML5 canvas element"
    </canvas>
</body>
</html>
```

shaders



código JS da aplicação



# Javascript

variável  
global

```
var gl;
```

função executada automaticamente quando o browser  
terminar de carregar a página

```
window.onload = function init() {  
  var canvas = document.getElementById("gl-canvas");  
  gl = WebGLUtils.setupWebGL(canvas);  
  if(!gl) { alert("WebGL isn't available"); }  
}
```

id de <canvas>  
no HTML

```
// Three vertices  
var vertices = [  
  vec2(-0.5, -0.5),  
  vec2(0, 0.5),  
  vec2(0.5, -0.5)  
];
```

As coordenadas dos vértices do triângulo

...

# Javascript

```
...  
// Configure WebGL  
gl.viewport(0,0,canvas.width, canvas.height);  
gl.clearColor(1.0, 1.0, 1.0, 1.0);  
  
// Load shaders and initialize attribute buffers  
var program = initShaders(gl, "vertex-shader", "fragment-shader");  
gl.useProgram(program);  
  
// Load the data into the GPU  
var bufferId = gl.createBuffer();  
gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);  
gl.bufferData(gl.ARRAY_BUFFER, flatten(vertices), gl.STATIC_DRAW);  
...
```

id's dos shaders no HTML

# Javascript

```
...  
// Associate our shader variables with our data buffer  
var vPosition = gl.getAttributeLocation(program, "vPosition");  
gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0);  
gl.enableVertexAttribArray(vPosition);  
render();  
}
```

```
function render() {  
  gl.clear(gl.COLOR_BUFFER_BIT);  
  gl.drawArrays(gl.TRIANGLES, 0, 3);  
}
```

Descreve quais e como os dados se encontram armazenados no buffer

limpa o fundo do canvas

manda desenhar o triângulo