SGI - WebGL intro

Nuria Pelechano Gómez
npelechano@lsi.upc.edu

WebGL intro

- 3D graphics API based on OpenGL ES 2.0
- Shader-based API using GLSL
- cross-platform
- JavaScript & HTML5 Canvas element
- WebGL brings plugin-free 3D to the web
- Apple (Safari), Google (Chrome), Mozilla (Firefox), and Opera (Opera) are members of the WebGL Working Group.

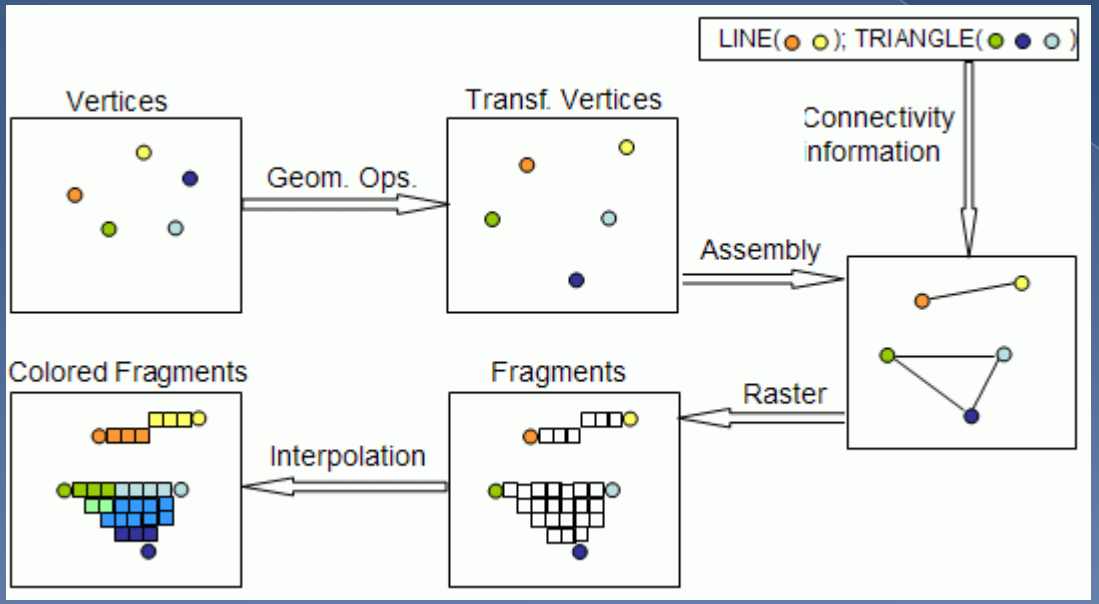
WebGL support on my system?

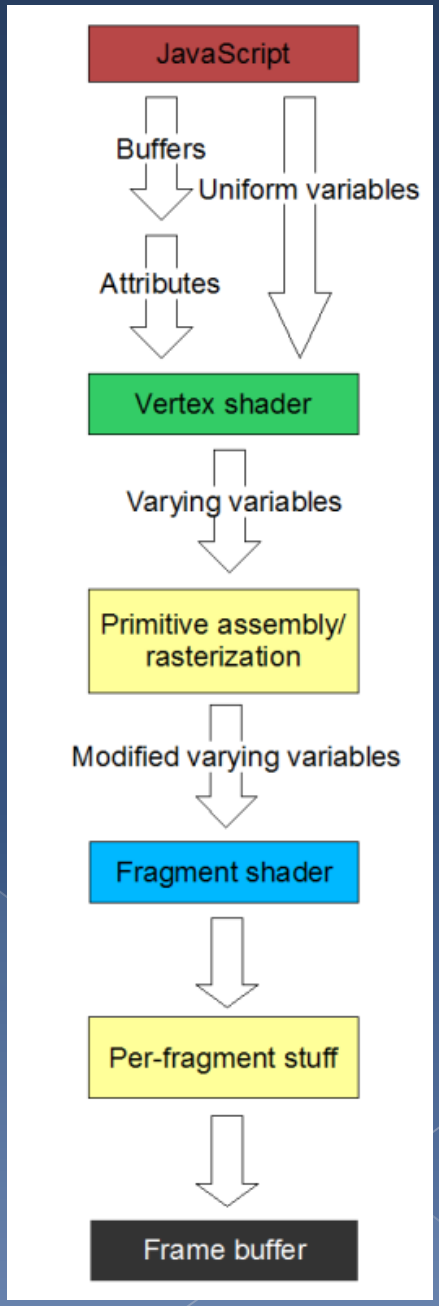
http://analyticalgraphicsinc.github.com/webglreport/

WebGL

- Introduction
- HTML5 - canvas
- Initializing WebGL
- Scripts for shaders
- Buffers
- Interaction
- Animation
- Textures
- Shaders – What's new in WebGL?
- Useful Javascript libraries

Pipeline


LINE( ); TRIANGLE( )
Vertices
Transf. Vertices
Connectivity
nformation
Geom. Ops.
Assembly
Colored Fragments
Fragments
Raster
Interpolation

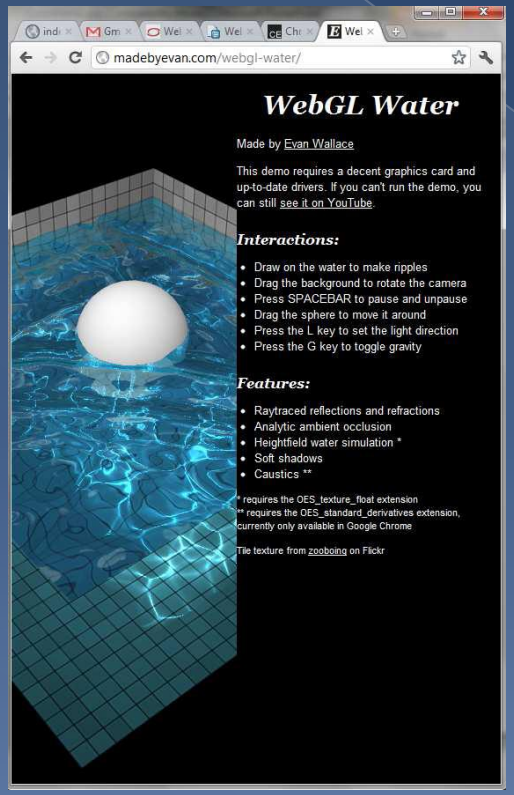

JavaScript
Buffers
Uniform variables
Attributes
Vertex shader
Varying variables
Primitive assembly/
rasterization
Modified varying variables
Fragment shader
Per-fragment stuff
Frame buffer

WebGL

- Includes
  - Vertex shaders
  - Fragment shaders
  - Vertex buffers
  - Textures
  - Framebuffers
  - Render states
  - …
- Does not include
  - Geometry shaders
  - Tessellation shaders
  - Vertex Array
  - Multiple render targets
  - Floating-point textures
  - Compressed textures
  - FS depth writes
  - …

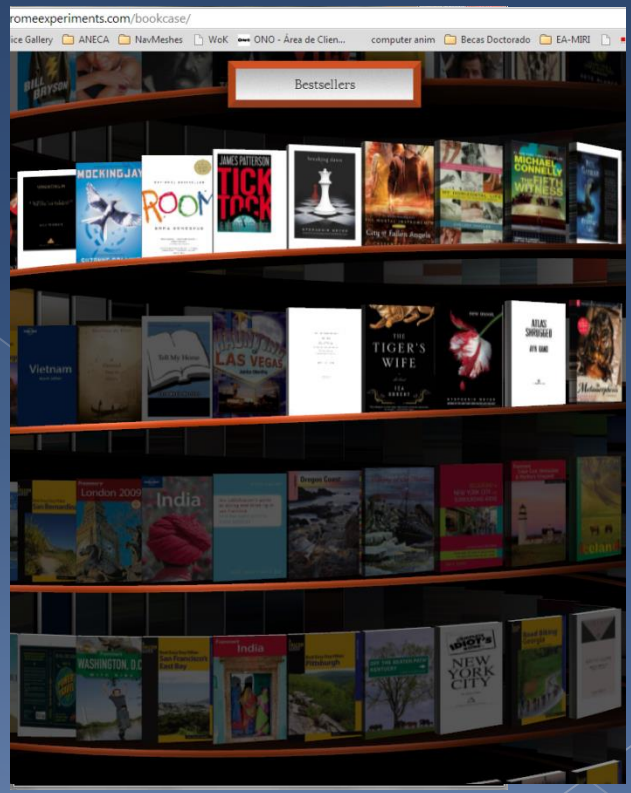
Demos

WebGL Water



http://madebyevan.com/webgl-water/

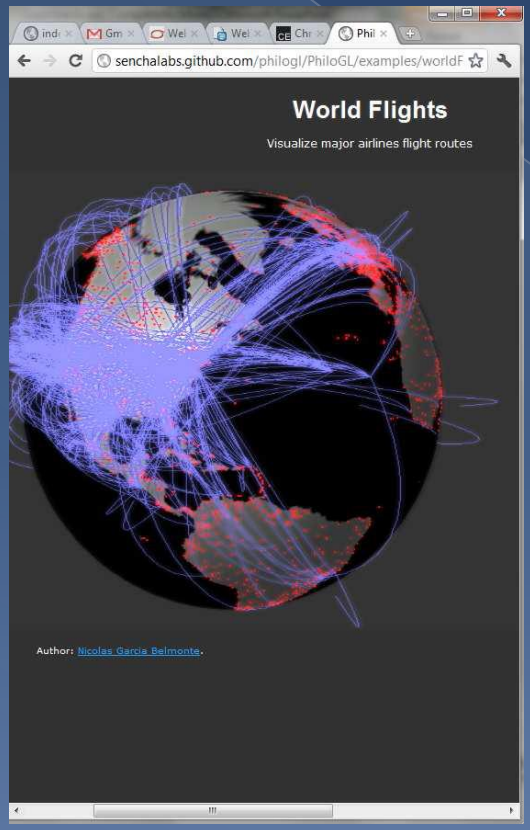
Bookcase



http://workshop.chromeexperiments.com/bookcase/

Demos

World Flights



http://senchalabs.github.com/philogl/PhiloGL/examples/worldFlights/

WebGL Jellyfish

http://chrysaora.com/

The HTML document

- What do we need?
  - A function to manage the graphics interface. Defined inside the <body>.
  - A canvas to draw all the WebGL objects.
  - external libraries using the <script> tags.

```
<html>
<head>
<script src="webGlFunctions.js“ type="text/javascript"></script>
</head>

<body onload="webGLStart();">
<canvas id="lesson01-canvas" width=“600" height=“600">
</canvas>
</body>
</html>
```

Init function

```
<script type="text/javascript">
function webGLStart() {
   var canvas = document.getElementById("lesson01-canvas");
   initGL(canvas);
   initShaders();
   initBuffers();
   gl.clearColor(0.0, 0.0, 1.0, 1.0);
   gl.enable(gl.DEPTH_TEST);
   drawScene();
}
</script>
```

Init GL

- Getting the webGL context

```
var gl;
  function initGL(canvas) {
    try {
      gl = canvas.getContext("experimental-webgl");
      gl.viewportWidth = canvas.width;
      gl.viewportHeight = canvas.height;
    } catch(e) {
    }
    if (!gl) {
      alert("Could not initialise WebGL, sorry :-(");
    }
  }
```

Init Shader

```
var shaderProgram;
function initShaders() {
    var fragmentShader = getShader(gl, "shader-fs");
    var vertexShader = getShader(gl, "shader-vs");
    shaderProgram = gl.createProgram();
    gl.attachShader(shaderProgram, vertexShader);
    gl.attachShader(shaderProgram, fragmentShader);
    gl.linkProgram(shaderProgram);

    gl.useProgram(shaderProgram);
    shaderProgram.vertexPositionAttribute = gl.getAttribLocation(shaderProgram,
"aVertexPosition");
    gl.enableVertexAttribArray(shaderProgram.vertexPositionAttribute);
}
```

Get Shader

```
function getShader(gl, id) {
    var shaderScript = document.getElementById(id);
    if (!shaderScript) { return null; }
    var str = "";
    var k = shaderScript.firstChild;
    while (k) {
        if (k.nodeType == 3) {
            str += k.textContent;
        }
        k = k.nextSibling;
    }
    var shader;
    if (shaderScript.type == "x-shader/x-fragment") {
        shader = gl.createShader(gl.FRAGMENT_SHADER);
    } else if (shaderScript.type == "x-shader/x-vertex") {
        shader = gl.createShader(gl.VERTEX_SHADER);
    } else { return null; }
    gl.shaderSource(shader, str);
    gl.compileShader(shader);
    if (!gl.getShaderParameter(shader, gl.COMPILE_STATUS)) {
        alert(gl.getShaderInfoLog(shader));
        return null;
    }
    return shader;
}
```

Fragment Shader

- Written in GLSL
- tell the graphics card how precise we want it to be with floating-point numbers (medium precision is supported by all WebGL devices)
- then simply specifies that everything that is drawn will be drawn in red.

```
<script id="shader-fs" type="x-shader/x-fragment">
  precision mediump float;

  void main(void) {
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
  }
</script>
```

Vertex Shader

- Written in GLSL

```
<script id="shader-vs" type="x-shader/x-vertex">
  attribute vec3 aVertexPosition;

  void main(void) {
    gl_Position = vec4(aVertexPosition, 1.0);
  }
</script>
```

Built-In Inputs, Outputs

- **Vertex Shader Special Variables**
  - **Outputs:**
    - highp vec4 gl_Position (transformed vertex, position clip coordinates)
    - mediump float gl_PointSize (transformed point size, pixels)
- **Fragment Shader Special Variables**
  - **Inputs:**
    - mediump vec4 gl_FragCoord (fragment position within frame buffer, device coordinates)
    - bool gl_FrontFacing (fragment belongs to a front-facing primitive, Boolean)
    - mediump vec2 gl_PointCoord (fragment position within a point (point rasterization only) 0.0 to 1.0 for each component)
  - **Outputs:**
    - mediump vec4 gl_FragColor (fragment color, RGBA color)
    - mediump vec4 gl_FragData[*n*] (fragment color for color attachment *n*, RGBA color)

Init Buffers

- buffers are actually a bit of memory on the graphics card
- by putting the vertex positions on the card once in our initialization code and then, when we come to draw the scene, essentially just telling WebGL to "draw those things I told you about earlier", we can make our code really efficient, especially once we start animating the scene and want to draw the object tens of times every second to make it move.

Init Buffers:


```
// define global variables:
var triangleVertexPositionBuffer;
var squareVe rtexPositionBuffer;

function initBuffers()
{
   triangleVertexPositionBuffer = gl.createBuffer();
   gl.bindBuffer(gl.ARRAY_BUFFER, triangleVertexPositionBuffer);
   var vertices = [
            0.0,  1.0,  0.0,
           -1.0, -1.0,  0.0,
            1.0, -1.0,  0.0
        ];
    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(vertices),
                          gl.STATIC_DRAW);
   triangleVertexPositionBuffer.itemSize = 3;
   triangleVertexPositionBuffer.numItems = 3;
```

Init Buffers:


```
    squareVertexPositionBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, squareVertexPositionBuffer);
    vertices = [
         1.0,  1.0,  0.0,
        -1.0,  1.0,  0.0,
         1.0, -1.0,  0.0,
        -1.0, -1.0,  0.0
    ];
    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(vertices),
                  gl.STATIC_DRAW);
    squareVertexPositionBuffer.itemSize = 3;
    squareVertexPositionBuffer.numItems = 4;
}
```

Draw Scene (1)

- Specify the viewport size with the canvas information
- Clear the canvas

```
function drawScene() {
   gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);
   gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
…
```

Draw Scene (2)

- call gl.bindBuffer to specify a current buffer, and then call the code that operates on it.
- Tell WebGL that the values in it should be used for vertex positions.
- draw the array of vertices as triangles, starting with item 0 in the array and going up to the numItems element

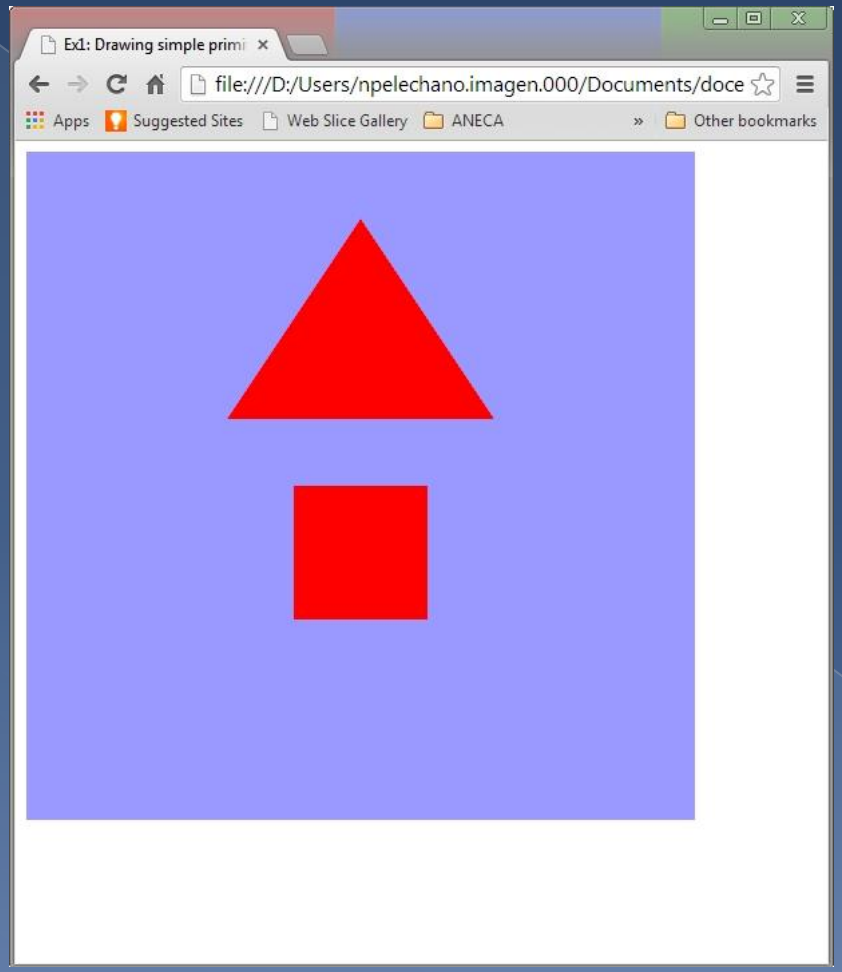
```
…
  gl.bindBuffer(gl.ARRAY_BUFFER, triangleVertexPositionBuffer);
  gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute,
            triangleVertexPositionBuffer.itemSize, gl.FLOAT, false, 0, 0);
  gl.drawArrays(gl.TRIANGLES, 0,
                    triangleVertexPositionBuffer.numItems);
…
```

Draw Scene (3)

- Same for the square:

```
…
gl.bindBuffer(gl.ARRAY_BUFFER, squareVertexPositionBuffer);
gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute,
                       squareVertexPositionBuffer.itemSize, gl.FLOAT, false, 0, 0);
gl.drawArrays(gl.TRIANGLE_STRIP, 0,
                       squareVertexPositionBuffer.numItems);
 }
```

Draw Scene (4)


Ex1: Drawing simple primi
file:///D:/Users/npelechano.imagen.000/Documents/doce
Apps
Suggested Sites
Web Slice Gallery
ANECA
Other bookmarks

Interaction

- demo
- Programmed with JS using eventListeners.

```
canvas.addEventListener('mousedown', function(e){
        click_x=e.clientX;
        click_y=e.clientY;
        moving=true;
        }, false);
canvas.addEventListener('mousemove', function(e){
        calculateDesp(e.clientX, e.clientY, canvas);
        }, false);
canvas.addEventListener('mouseup', function(e){
        moving=false;
        }, false);
```

Animation (I)

- demo
- …

```
function tick() {
    requestAnimFrame(tick);
    drawScene();
    animate();
}

function webGLStart() {
    …
    tick();
}
```

Animation (II)

- Just before drawScence() we add two new variables:

```
var rTri = 0;
var rSquare = 0;
```

- These are used to track the rotation of the triangle and the square respectively

Animation (III)

- In drawScene():

```
mat4.perspective(45, gl.viewportWidth/gl.viewportHeight, 0.1,100.0,pMatrix);
mat4.identity(mvMatrix);
mat4.translate(mvMatrix, [-1.5, 0.0, -7.0]);

mvPushMatrix();
mat4.rotate(mvMatrix, degToRad(rTri), [0, 1, 0]);

gl.bindBuffer(gl.ARRAY_BUFFER, triangleVertexPositionBuffer);
gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute, …
gl.bindBuffer(gl.ARRAY_BUFFER, triangleVertexColorBuffer);
gl.vertexAttribPointer(shaderProgram.vertexColorAttribute, …

setMatrixUniforms();
gl.drawArrays(gl.TRIANGLES, 0, triangleVertexPositionBuffer.numItems);
mvPopMatrix();
```

Animation (IV)

- In animate();

```
var lastTime = 0;

  function animate() {
      var timeNow = new Date().getTime();
      if (lastTime != 0) {
          var elapsed = timeNow - lastTime;

          rTri += (90 * elapsed) / 1000.0;
          rSquare += (75 * elapsed) / 1000.0;
      }
      lastTime = timeNow;
  }
```

Useful WebGL links:

- http://www.khronos.org/webgl/
- http://learningwebgl.com/blog/

Useful Javascript libraries

- Animations: webgl-utils.js
- Matrices, camera: gl-Matrix-min.js
  - http://glmatrix.net/docs/2.2.0/symbols/mat4.html
- Cameras, objects, lights, materials and more: Three.js: https://github.com/mrdoob/three.js/
- Scene Graph: sceneJS: http://scenejs.org/
- Many more:
- http://www.khronos.org/webgl/wiki/User_Contribution