An Introduction to WebGL

Arsène Pérard-Gayot

January 23, 2020

Introduction

What is WebGL?

- Cross-platform 3D web standard from Khronos
- Low-level API
- Based on OpenGL ES 2.0
- Supported by major browsers: Firefox, Chrome, Safari, Opera

Introduction

Why do 3D on the web?

- More interactive content
  - `http://nouvellevague.ultranoir.com/`
- Visualization in a web page
  - `https://www.biodigital.com/`
  - `https://www.chromeexperiments.com/globe`
- Video games
  - `https://www.unrealengine.com/html5/`

An Empty Web Page

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
<!doctype html>
<html>
  <head>
    <title>Page Title</title>
  </head>

  <body>
    Hello World!
  </body>
</html>
```

Setting up WebGL (1)

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
<!doctype html>
<html>
  <head>
    <title>Setting up WebGL</title>

    <script>
      /* ... */
    </script>
  </head>

  <body onload="setupWebGL();">
    <canvas id="glcanvas" width="500" height="500">
    </canvas>
  </body>
</html>
```

Setting up WebGL (2)


```
function setupWebGL() {
    var canvas = document.getElementById("glcanvas");
    try {
        gl = canvas.getContext("experimental-webgl");
        gl.viewportWidth = canvas.width;
        gl.viewportHeight = canvas.height;
    } catch (e) {
    }

    if (!gl) alert("Cannot initialize WebGL");

    gl.clearColor(0.0, 0.0, 0.0, 1.0);
    gl.enable(gl.DEPTH_TEST);
    gl.clear();
}
```

Setting up Shaders (1)

SAARLAND UNIVERSITY
COMPUTER SCIENCE

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

    uniform mat4 modelViewProj;
    attribute vec3 pos;

    void main(void) {
        gl_Position = modelViewProj * vec4(pos, 1);
    }
</script>
```

Setting up Shaders (2)


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

    void main(void) {
        gl_FragColor = vec4(1, 1, 1, 1);
    }
</script>
```

Setting up Shaders (3)


```
function getShader(id) {
    var shaderScript = document.getElementById(id);

    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);
    }

    gl.shaderSource(shader, shaderScript.innerHTML);
    gl.compileShader(shader);

    if (!gl.getShaderParameter(shader, gl.COMPILE_STATUS))
        alert(gl.getShaderInfoLog(shader));

    return shader;
}
```

Setting up Shaders (4)

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
function setupShaders(id) {
    var fragShader = getShader("shader-fs");
    var vertShader = getShader("shader-vs");
    shaderProgram = gl.createProgram();
    gl.attachShader(shaderProgram, vertShader);
    gl.attachShader(shaderProgram, fragShader);
    gl.linkProgram(shaderProgram);

    if (!gl.getProgramParameter(shaderProgram, gl.LINK_STATUS))
        alert("Cannot link program");

    gl.useProgram(shaderProgram);

    posAttribute = gl.getAttribLocation(shaderProgram, "pos");
    gl.enableVertexAttribArray(posAttribute);

    modelViewProjUniform = gl.getUniformLocation(shaderProgram,
        "modelViewProj");
}
```

Setting up Buffers


```
function setupBuffers() {
    vertexBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuffer);
    var vertices = [
         0,  0.5, 0,
      -0.5, -0.5, 0,
       0.5, -0.5, 0
    ];
    gl.bufferData(gl.ARRAY_BUFFER,
        new Float32Array(vertices),
        gl.STATIC_DRAW);
}
```

Drawing a Triangle

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
function draw() {
    var modelViewProj = new Float32Array([
        1, 0, 0, 0,
        0, 1, 0, 0,
        0, 0, 1, 0,
        0, 0, 0, 1
    ]);

    gl.uniformMatrix4fv(modelViewProjUniform, false,
    modelViewProj);

    gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuffer);
    gl.vertexAttribPointer(posAttribute, 3, gl.FLOAT, false,
     0, 0);
    gl.drawArrays(gl.TRIANGLES, 0, 3);
}
```

Matrix and Vector Math Library

Matrix operations?

- Required to set up the projection/view matrices
- Reuse existing libraries: `https://glmatrix.net`

```
<script type="text/javascript" src="gl-matrix-min.js">
</script>
```

Adding Perspective


```
function draw() {
    mat4.perspective(proj,
        90.0,                                // fov y
        viewportWidth / viewportHeight,      // ratio
        0.1,                                 // near
        100.0);                              // far
    mat4.identity(view);
    mat4.lookAt(view,
        [0, 0, 1],    // eye
        [0, 0, 0],    // center
        [0, 1, 0]);   // up
    mat4.identity(model);
    mat4.rotateY(model, model, angle);
    mat4.multiply(modelView, view, model);
    mat4.multiply(modelViewProj, proj, modelView);

    angle += 0.01;

    /* ... */
}
```

Updating the Canvas

Redraw when...

- Games: every 16ms (for 60 FPS)
  - Use `setInterval()`
- Others: on user input
  - Javascript events `onmousemove`, `onclick`, ...

Adding Textures (1)

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
function setupTextures() {
    texture = gl.createTexture();
    var image = new Image();
    image.onload = function() {
        gl.bindTexture(gl.TEXTURE_2D, texture);
        gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, true);
        gl.texImage2D(gl.TEXTURE_2D,
            0,                  // Mipmap level
            gl.RGBA,            // Internal format
            gl.RGBA,            // Format of the pixel data
            gl.UNSIGNED_BYTE,   // Type of the pixel data
            image);             // Pixel data
        gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER,
            gl.NEAREST);
        gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER,
            gl.NEAREST);
    }
    image.src = "texture.png";
}
```

Adding Textures (2)

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
precision mediump float;

uniform mat4 modelViewProj;

attribute vec3 pos;
attribute vec2 uv;

varying vec2 texCoord;

void main(void) {
    gl_Position = modelViewProj * vec4(pos, 1);
    texCoord = uv;
}
```

Adding Textures (2)

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
precision mediump float;

varying vec2 texCoord;

uniform sampler2D sampler;

void main(void) {
    gl_FragColor = texture2D(sampler, texCoord);
}
```

Adding Textures (3)


```
function setupBuffers() {
    vertexBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuffer);
    var vertices = [
    //      x,     y,     z,     u,     v
            0,   0.5,     0,   0.5,     1,
         -0.5,  -0.5,     0,     0,     0,
          0.5,  -0.5,     0,     1,     0
    ];
    gl.bufferData(gl.ARRAY_BUFFER,
                  new Float32Array(vertices),
                  gl.STATIC_DRAW);
}
```

Adding Textures (4)


```
function draw() {
    /* ... */
    gl.activeTexture(gl.TEXTURE0);
    gl.bindTexture(gl.TEXTURE_2D, texture);
    gl.uniform1i(samplerUniform, 0);

    gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuffer);
    gl.vertexAttribPointer(posAttribute,
        3, gl.FLOAT, false,
        5 * 4, 0);
    gl.vertexAttribPointer(uvAttribute,
        2, gl.FLOAT, false,
        5 * 4, 3 * 4);
    gl.drawArrays(gl.TRIANGLES, 0, 3);
}
```

Mipmapping

Trilinear filtering

- Hardware accelerated texture lookups

```
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER,
  gl.LINEAR_MIPMAP_LINEAR);
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER,
  gl.LINEAR);
```

- Mipmap level generation with `gl.generateMipmap()`
- Transparent for the shaders
  - Hidden behind the sampler object

Adding Lighting (1)

Simple Lighting

- Direct lighting, no shadow computation
- Point light position & intensity passed as uniforms
- Normals are added to the vertex buffer
- Use the inverse of the transposed model matrix for normals!

Adding Lighting (2)

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
/* ... */
attribute vec3 pos;
attribute vec3 n;
attribute vec3 uv;

varying vec2 texCoord;
varying vec3 worldPos;
varying vec3 normal;

void main(void) {
    gl_Position = modelViewProj * vec4(pos, 1);

    texCoord = uv;
    worldPos = vec3(modelMatrix * vec4(pos, 1));
    normal   = vec3(normalMatrix * vec4(n, 0));
}
```

Adding Lighting (3)

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
precision mediump float;

varying vec2 texCoord;
varying vec3 worldPos;
varying vec3 normal;

uniform sampler2D sampler;
uniform vec3 lightPos;
uniform vec3 lightColor;

void main(void) {
    vec3 n = normalize(normal);
    vec3 lightVec = lightPos - worldPos;
    float diffuse = abs(dot(n, normalize(lightVec)) /
                        dot(lightVec, lightVec));
    vec4 color = texture2D(sampler, texCoord);
    gl_FragColor =
    vec4(color.rgb * lightColor * diffuse, 1);
}
```

Adding Lighting (4)


```
function draw() {
    /* ... */

    gl.uniformMatrix4fv(modelMatrixUniform, false, model);
    gl.uniformMatrix4fv(normalMatrixUniform, false, normal);

    /* ... */

    gl.uniform3fv(lightPosUniform,
        new Float32Array([0, 0, 10]));
    gl.uniform3fv(lightColorUniform,
        new Float32Array([100, 100, 100]));

    /* ... */

    gl.vertexAttribPointer(nAttribute, 3, gl.FLOAT, false,
        8 * 4, 3 * 4);

    /*...*/
}
```

Conclusion

What we have seen

- How to set up a canvas, and initialize a WebGL context
- How to load and compile shaders
- How to render a triangle
- How to use transformations in the shaders
- How to load and use images as textures
- How to do mipmapping
- How to render simple lighting effects