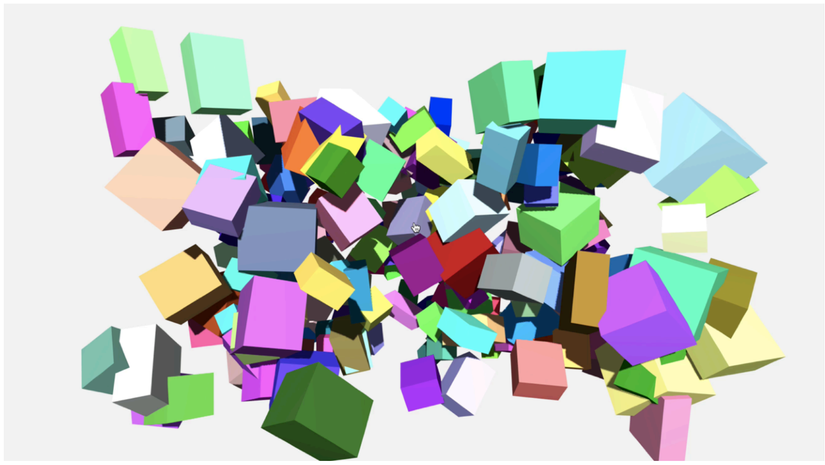
Visual Computing I:

Interactive Computer Graphics and Vision

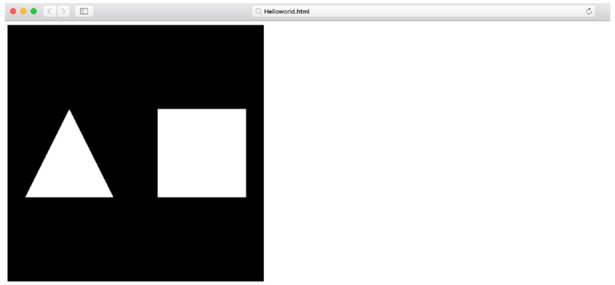


WebGL

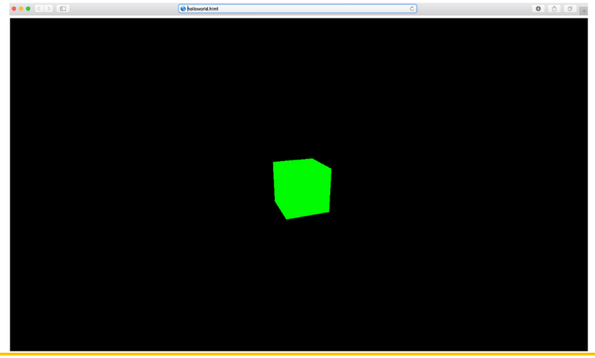
Stefanie Zollmann and Tobias Langlotz

TODAY

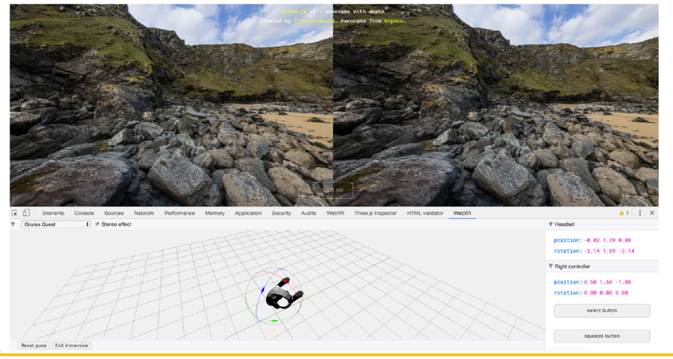
WEBGL



THREE.JS



WEBXR - SIMULATOR



WebGL

FRAMEWORKS

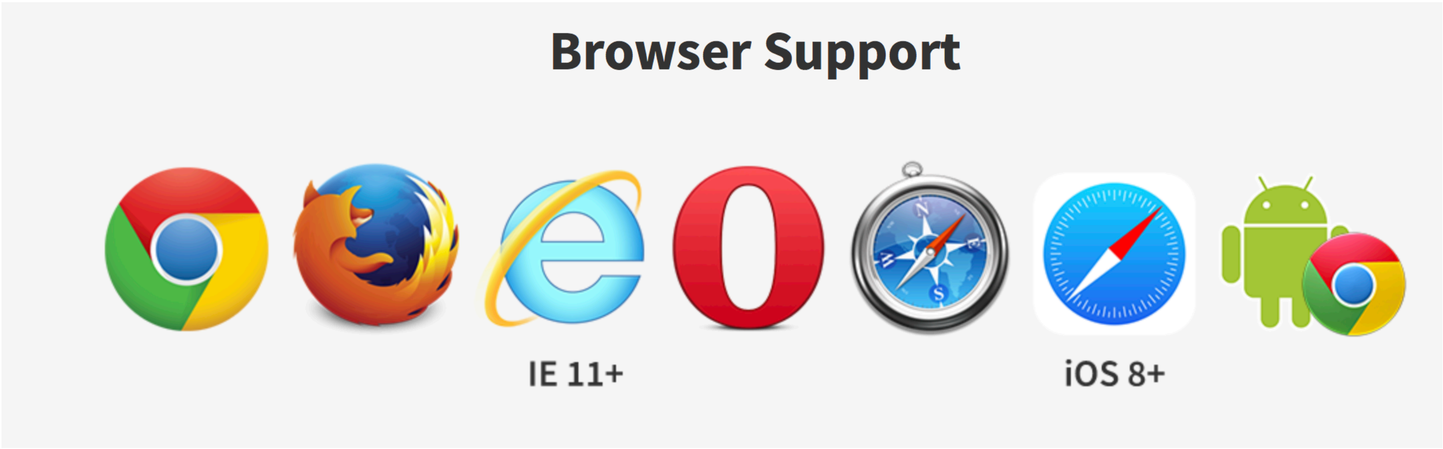
WEBXR

WebGL

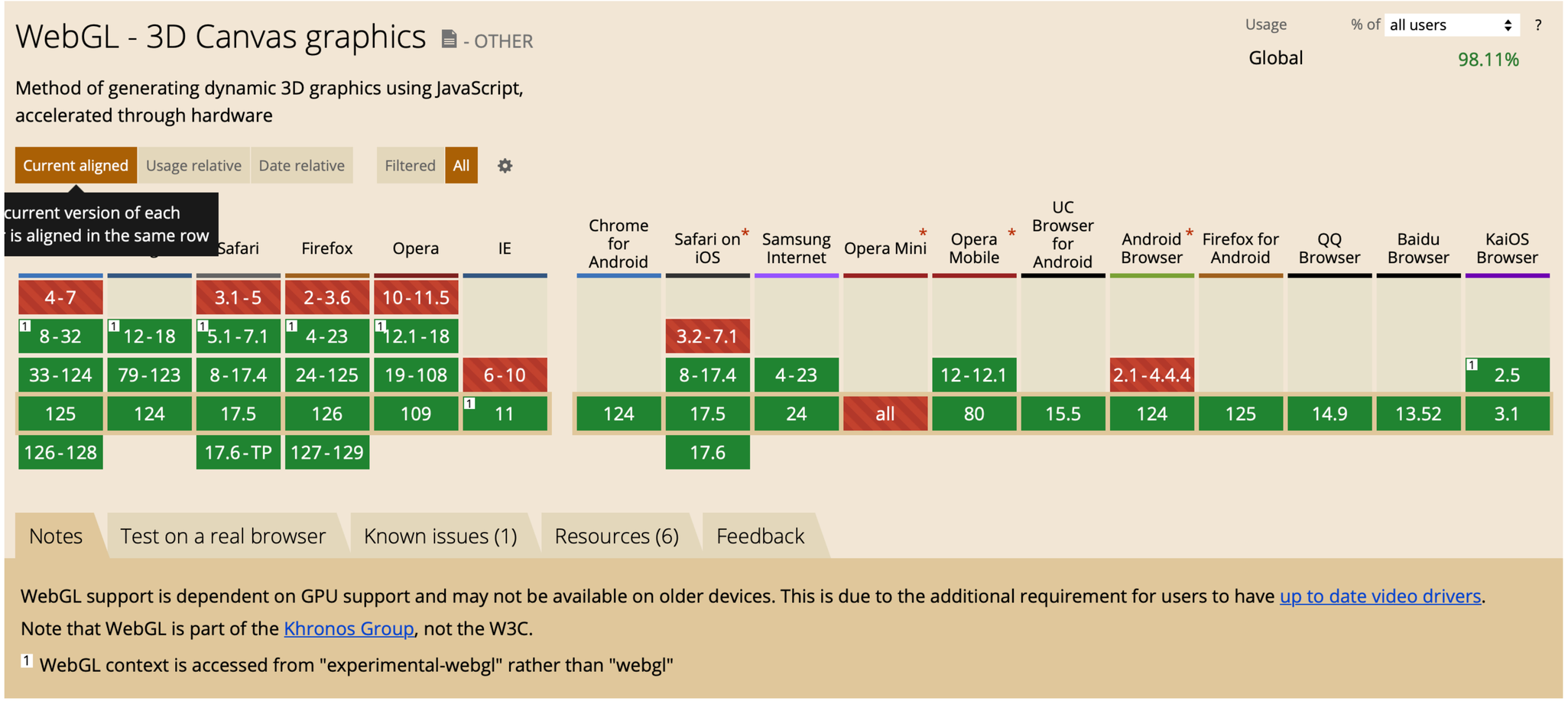
- Derivative of OpenGL
- More specifically, OpenGL ES version 2.0
- WebGL 2.0 is based on OpenGL ES 3.0
- Allows HTML pages to use WebGL to render using GPU resources
- Provides JavaScript bindings for OpenGL functions
- WebGL is under development by the Khronos Group
- Similar to modern OpenGL applications, all rendering is controlled by vertex and fragment shaders
- Supports GLSL

WebGL

- Inside an HTML <canvas> element.
- Development is likely to involve HTML, CSS, JavaScript, GLSL, in addition to WebGL's OpenGL-style naming.

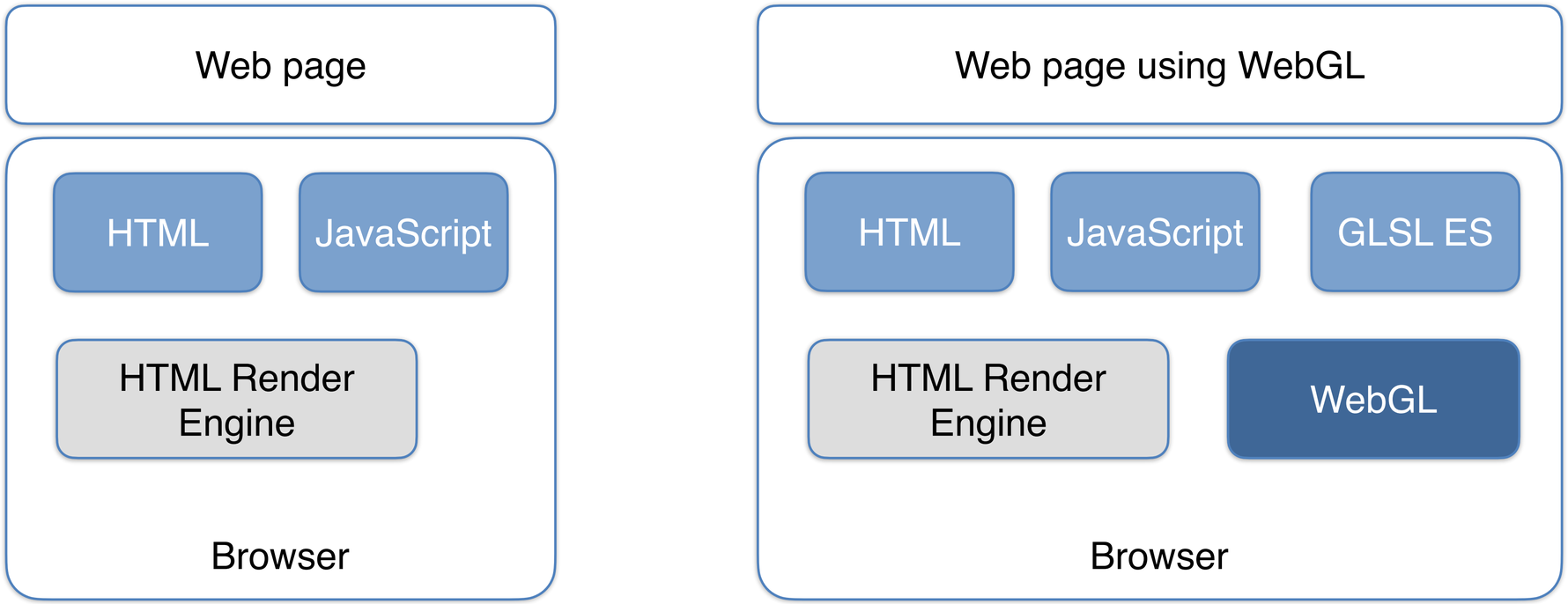


Support of WebGL

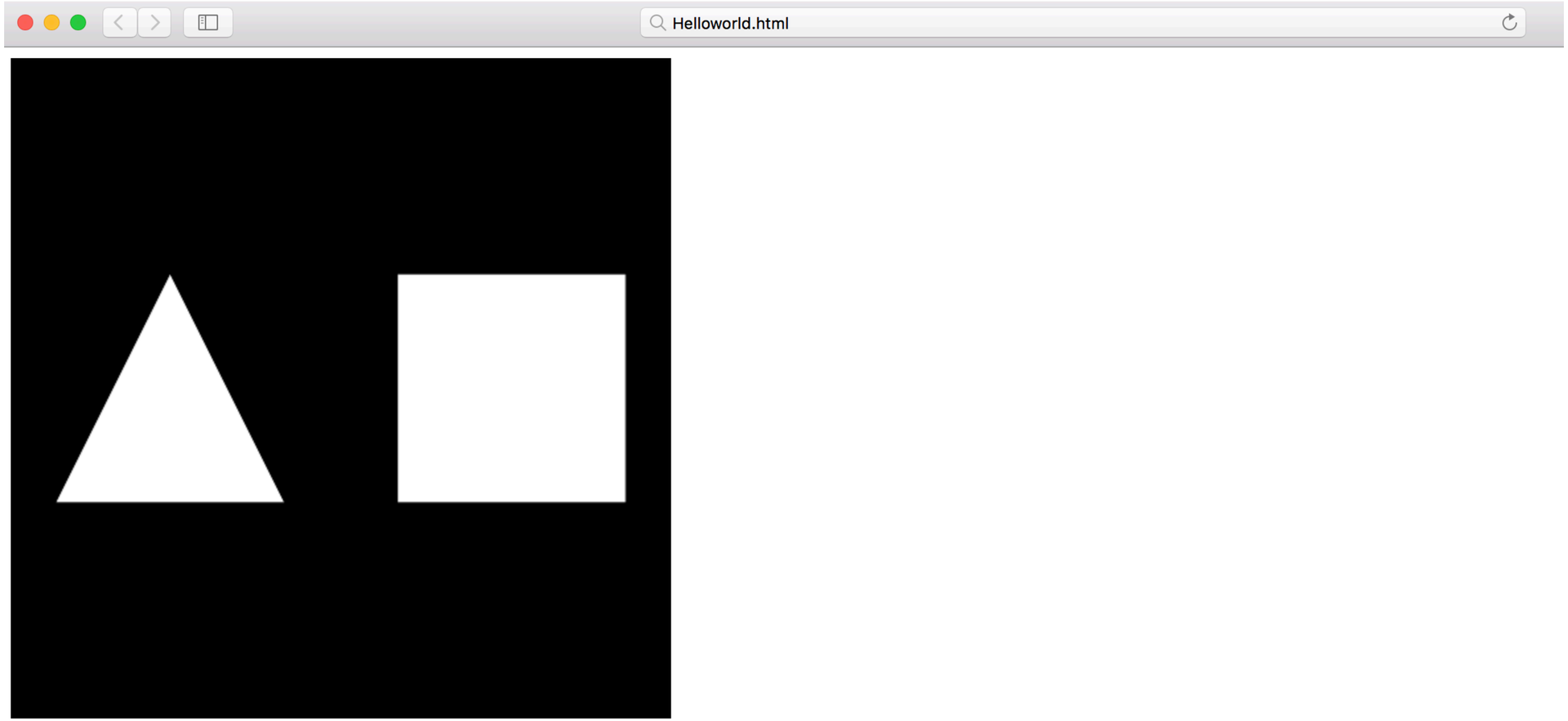


http://caniuse.com/#feat=webgl

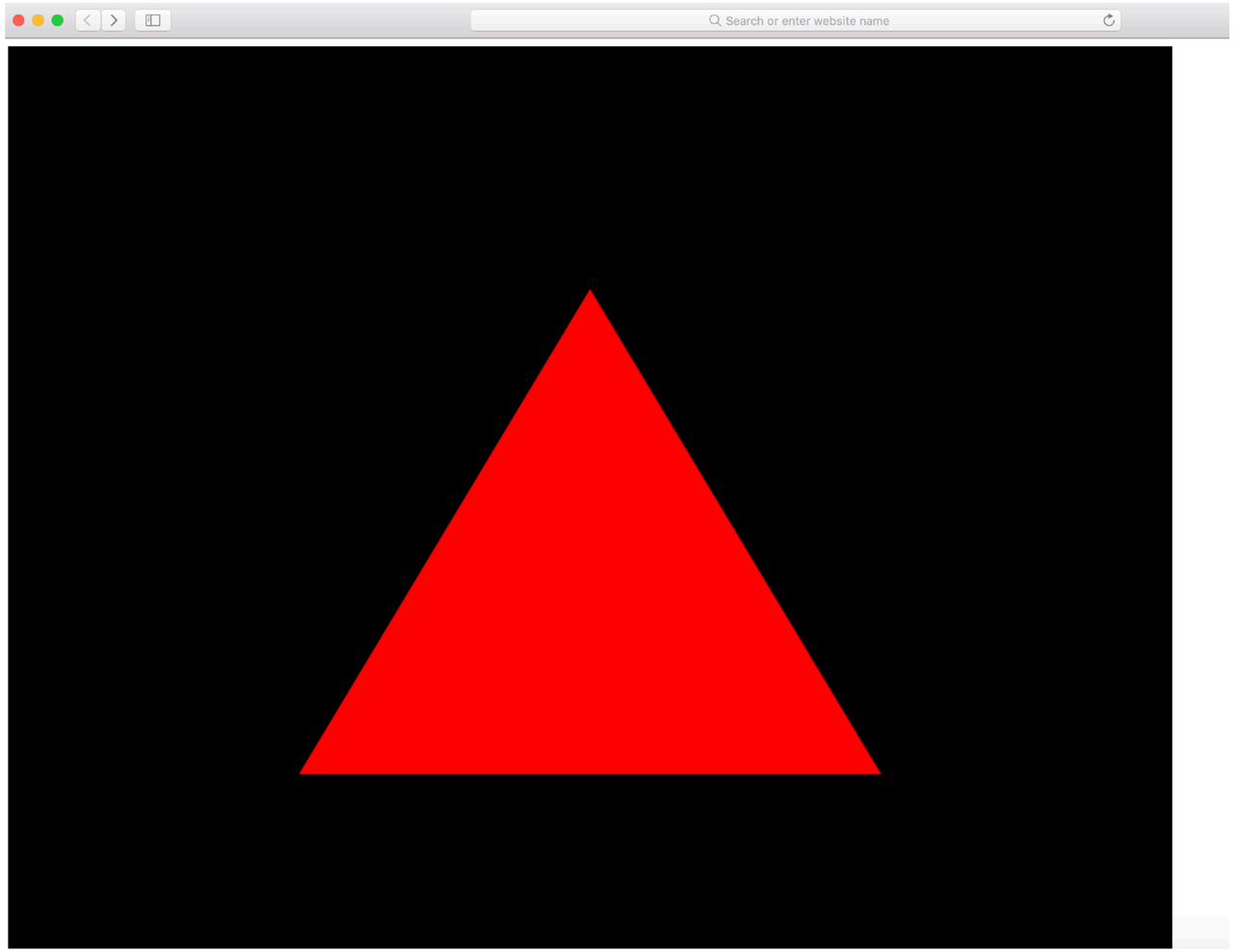
WebGL Application Structure



WebGL



WebGL



Example from WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL by Matsuda and Lea

WebGL “hello TRIANGLE”

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Hello Triangle</title>
  </head>

  <body onload="main()">
    <canvas id="webgl" width="1200" height="1000">
    Please use a browser that supports "canvas"
    </canvas>

    <script src="lib/webgl-utils.js"></script>
    <script src="lib/webgl-debug.js"></script>
    <script src="lib/cuon-utils.js"></script>
    <script src="HelloTriangle.js"></script>
  </body>
</html>
```

HTML: HELLOTRIANGLE.HTML

Example from WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL by Matsuda and Lea

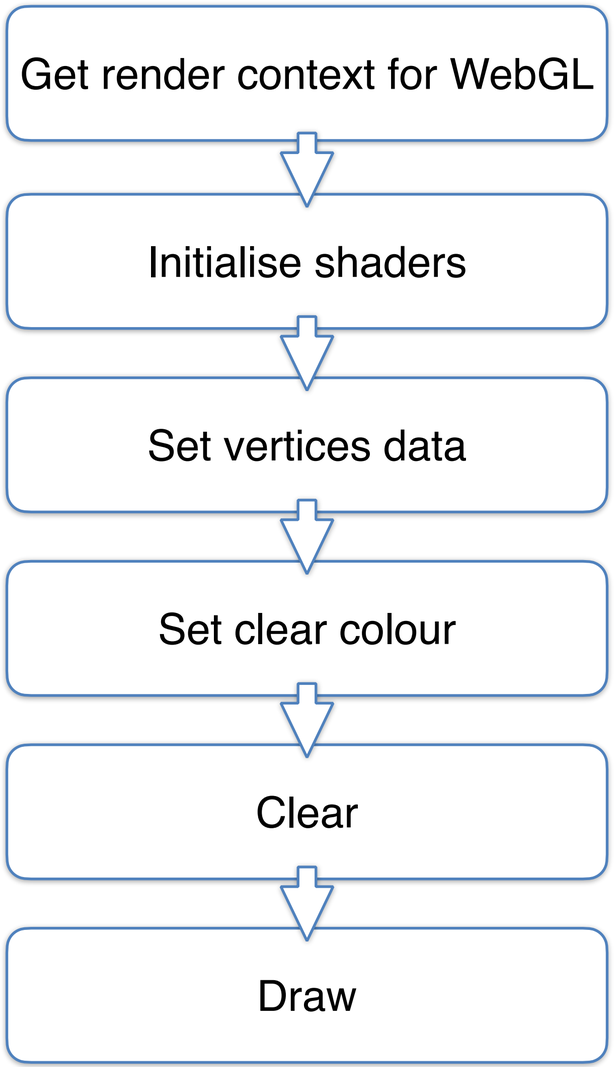
WebGL “hello TRIANGLE”

```
function main() {
  // Retrieve <canvas> element
  var canvas = document.getElementById('webgl');
  // Get the rendering context for WebGL
  var gl = getWebGLContext(canvas);
  if (!gl) {
    console.log('Failed to get the rendering context for WebGL’); return;
  }
  // Initialize shaders
  if (!initShaders(gl, VSHADER_SOURCE, FSHADER_SOURCE)) {
    console.log('Failed to intialize shaders.’); return;
  }
  // Write the positions of vertices to a vertex shader
  var n = initVertexBuffers(gl);
  if (n < 0) {
    console.log('Failed to set the positions of the vertices’); return;
  }
  // Specify the color for clearing <canvas>
  gl.clearColor(0, 0, 0, 1);
  // Clear <canvas>
  gl.clear(gl.COLOR_BUFFER_BIT);
  // Draw the rectangle
  gl.drawArrays(gl.TRIANGLES, 0, n);
}
```

JAVASCRIPT: HELLOTRIANGLE.JS

Example from WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL by Matsuda and Lea

WebGL Simple Workflow



WebGL “hello TRIANGLE”

```
function main() {
  // Retrieve <canvas> element
  var canvas = document.getElementById('webgl');
  // Get the rendering context for WebGL
  var gl = getWebGLContext(canvas);
  if (!gl) {
    console.log('Failed to get the rendering context for WebGL’); return;
  }
  // Initialize shaders
  if (!initShaders(gl, VSHADER_SOURCE, FSHADER_SOURCE)) {
    console.log('Failed to intialize shaders.’); return;
  }
  // Write the positions of vertices to a vertex shader
  var n = initVertexBuffers(gl);
  if (n < 0) {
    console.log('Failed to set the positions of the vertices’); return;
  }
  // Specify the color for clearing <canvas>
  gl.clearColor(0, 0, 0, 1);
  // Clear <canvas>
  gl.clear(gl.COLOR_BUFFER_BIT);
  // Draw the rectangle
  gl.drawArrays(gl.TRIANGLES, 0, n);
}
```

JAVASCRIPT: HELLOTRIANGLE.JS

Example from WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL by Matsuda and Lea

WebGL Create Vertexbuffer

```
function initVertexBuffers(gl) {
  var vertices = new Float32Array([ 0, 0.5,   -0.5, -0.5,   0.5, -0.5 ]);
  var n = 3; // The number of vertices
  // Create a buffer object
  var vertexBuffer = gl.createBuffer();
  if (!vertexBuffer) {
    console.log('Failed to create the buffer object');
    return -1;
  }
  // Bind the buffer object to target
  gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuffer);
  // Write date into the buffer object
  gl.bufferData(gl.ARRAY_BUFFER, vertices, gl.STATIC_DRAW);
  var a_Position = gl.getAttribLocation(gl.program, 'a_Position');
  if (a_Position < 0) {
    console.log('Failed to get the storage location of a_Position');
    return -1;
  }
  // Assign the buffer object to a_Position variable
   gl.vertexAttribPointer(a_Position, 2, gl.FLOAT, false, 0, 0);
  // Enable the assignment to a_Position variable
  gl.enableVertexAttribArray(a_Position);
  return n;
}
```

JAVASCRIPT: HELLOTRIANGLE.JS

Example from WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL by Matsuda and Lea

WebGL “hello TRIANGLE”

```
function main() {
  // Retrieve <canvas> element
  var canvas = document.getElementById('webgl');
  // Get the rendering context for WebGL
  var gl = getWebGLContext(canvas);
  if (!gl) {
    console.log('Failed to get the rendering context for WebGL’); return;
  }
  // Initialize shaders
  if (!initShaders(gl, VSHADER_SOURCE, FSHADER_SOURCE)) {
    console.log('Failed to intialize shaders.’); return;
  }
  // Write the positions of vertices to a vertex shader
  var n = initVertexBuffers(gl);
  if (n < 0) {
    console.log('Failed to set the positions of the vertices’); return;
  }
  // Specify the color for clearing <canvas>
  gl.clearColor(0, 0, 0, 1);
  // Clear <canvas>
  gl.clear(gl.COLOR_BUFFER_BIT);
  // Draw the rectangle
  gl.drawArrays(gl.TRIANGLES, 0, n);
}
```

JAVASCRIPT: HELLOTRIANGLE.JS

Example from WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL by Matsuda and Lea

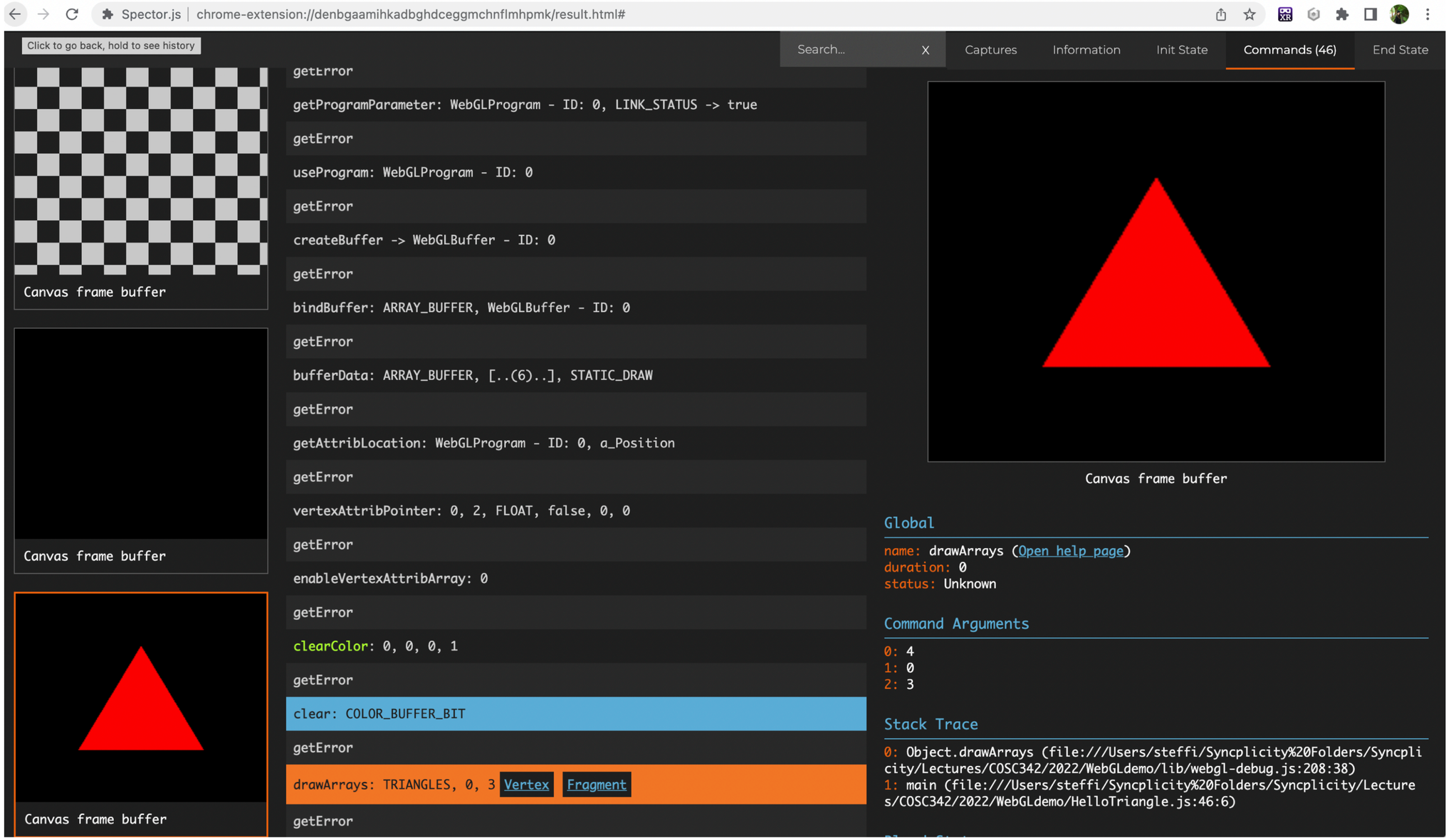
WebGL Shaders

```
var VSHADER_SOURCE =
  'attribute vec4 a_Position;\n' +
  'void main() {\n' +
  '  gl_Position = a_Position;\n' +
  '}\n';

// Fragment shader program
var FSHADER_SOURCE =
  'void main() {\n' +
  '  gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);\n' +
  '}\n';
```

JAVASCRIPT: HELLOTRIANGLE.JS

Example from WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL by Matsuda and Lea

WebGL Debugging



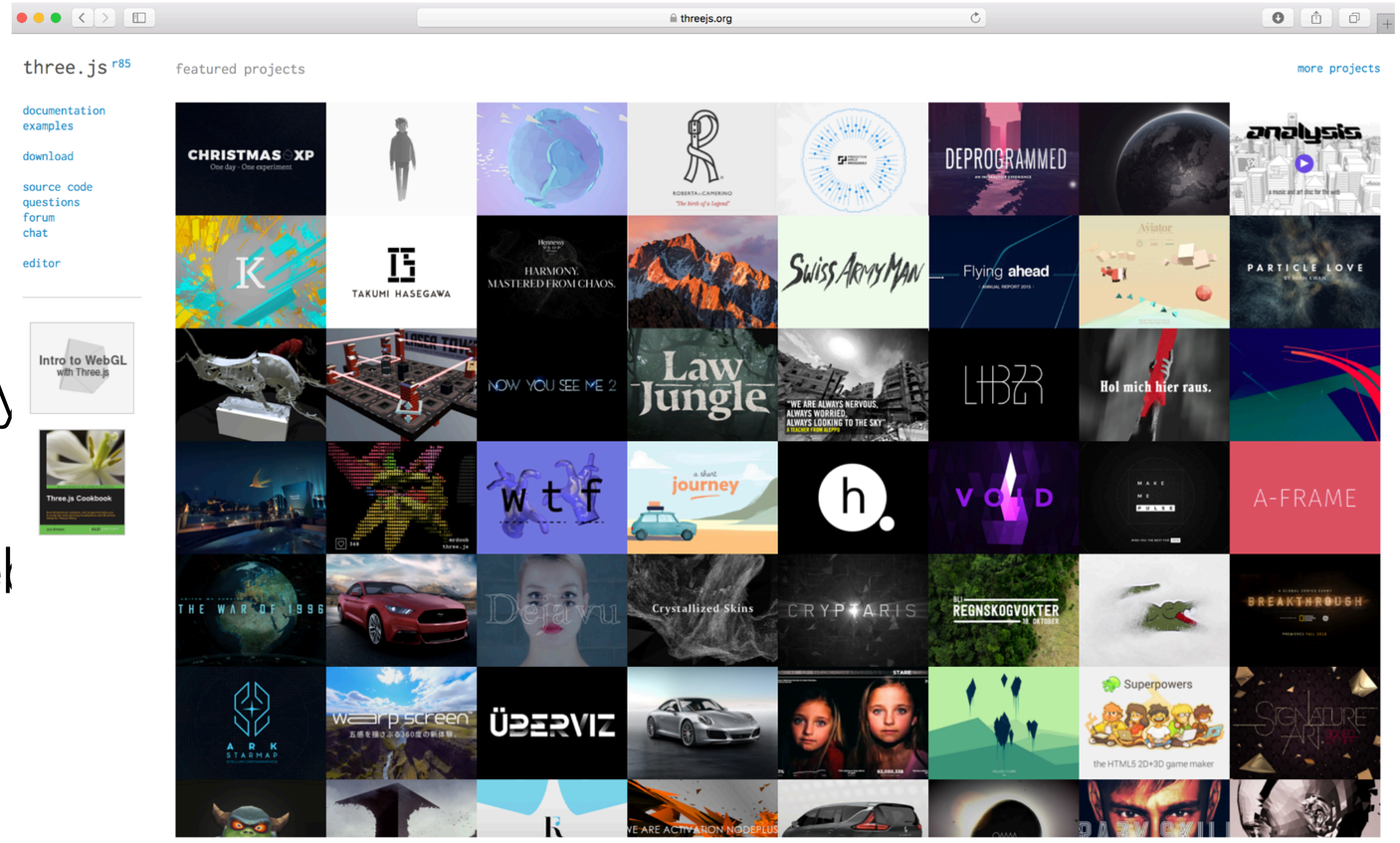
Spector.js

THREE.JS

https://threejs.org/examples/#webgl_panorama_cube

THREE.JS

- First released April 2010
- 3D Javascript Library
- Under MIT license
- Uses WebGL for rendering (previously also CanvasRenderer, SVGRenderer)
- Runs in all browsers that support Wel
- Website: threejs.org
- Library is in single javascript file



THREE.JS

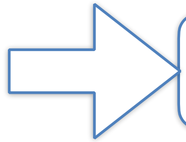
- Features:
  - Scenes: For adding and removing objects during runtime
  - Cameras: Different camera models (perspective, orthographic)
  - Lights: Ambient, direction, point and spot lights
  - Shadows: Control shadow casting and receiving
  - Materials: Phong, textures
  - Shaders: GLSL
  - Objects: Meshes
  - Geometries: Box, Planes etc.
  - Loaders: Obj, Json, Collada

THREE.JS: Hello Cube

```
<html>
	<head>
		<title>Hello Cube</title>
	</head>
	<body>
		<script src="js/three.js"></script>
		<script>
			var scene = new THREE.Scene();
			var camera = new THREE.PerspectiveCamera( 75, window.innerWidth/window.innerHeight, 0.1, 1000 );
			var renderer = new THREE.WebGLRenderer();
			renderer.setSize( window.innerWidth, window.innerHeight );
			document.body.appendChild( renderer.domElement );
			var geometry = new THREE.BoxGeometry( 1, 1, 1 );
			var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
			var cube = new THREE.Mesh( geometry, material );
			scene.add( cube );
			camera.position.z = 5;

			var render = function () {
				requestAnimationFrame( render );
				cube.rotation.x += 0.1;
				cube.rotation.y += 0.1;
				renderer.render(scene, camera);
			};
			render();
		</script>
	</body>
</html>
```

Hello Cube: HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset=utf-8>
        <title>Hello Cube</title>
    </head>
    <body>
        <script src="js/three.js"></script>
        <script>
            //Javascript
        </script>
    </body>
</html>
```



Library is in single javascript file

Hello Cube: Javascript

- First steps
  - Initialise scene
  - Initialise camera
  - Initialise renderer (Obtains rendering context from its domElement)

```
var scene = new THREE.Scene();
var camera = new THREE.PerspectiveCamera( 75, window.innerWidth / window.innerHeight, 0.1, 1000 );

var renderer = new THREE.WebGLRenderer();
renderer.setSize( window.innerWidth, window.innerHeight );
document.body.appendChild( renderer.domElement );
```

Hello Cube: Javascript

- Setup
  - Create geometry
  - Add geometry to scene
  - Move camera

```
var geometry = new THREE.BoxGeometry( 1, 1, 1 );
var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
var cube = new THREE.Mesh( geometry, material );
scene.add( cube );
camera.position.z = 5;
```

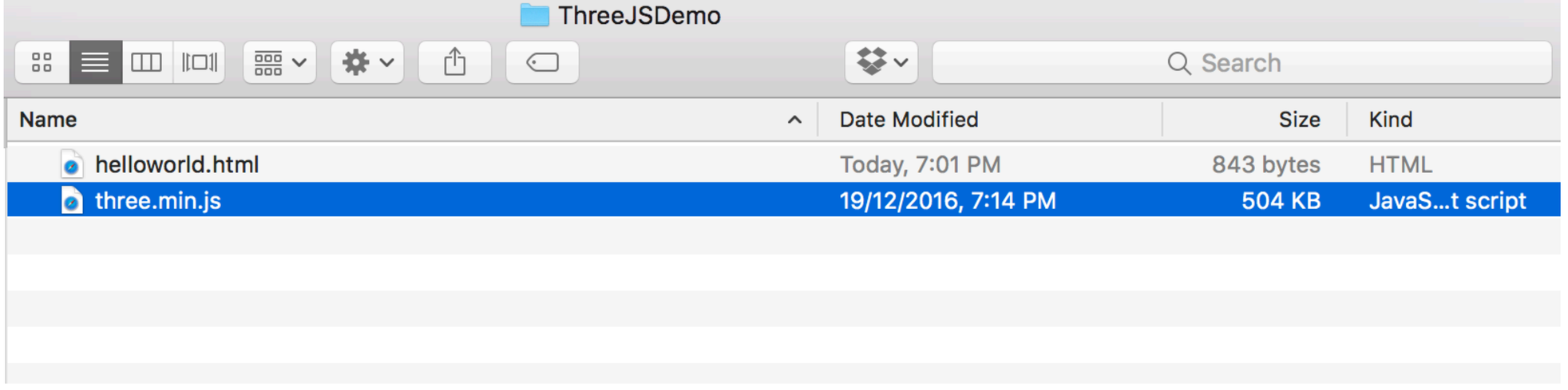
Hello Cube: Render Loop

- Define rendering
- requestAnimationFrame setups redrawing
- Add animation rotating the cube
- Call render method

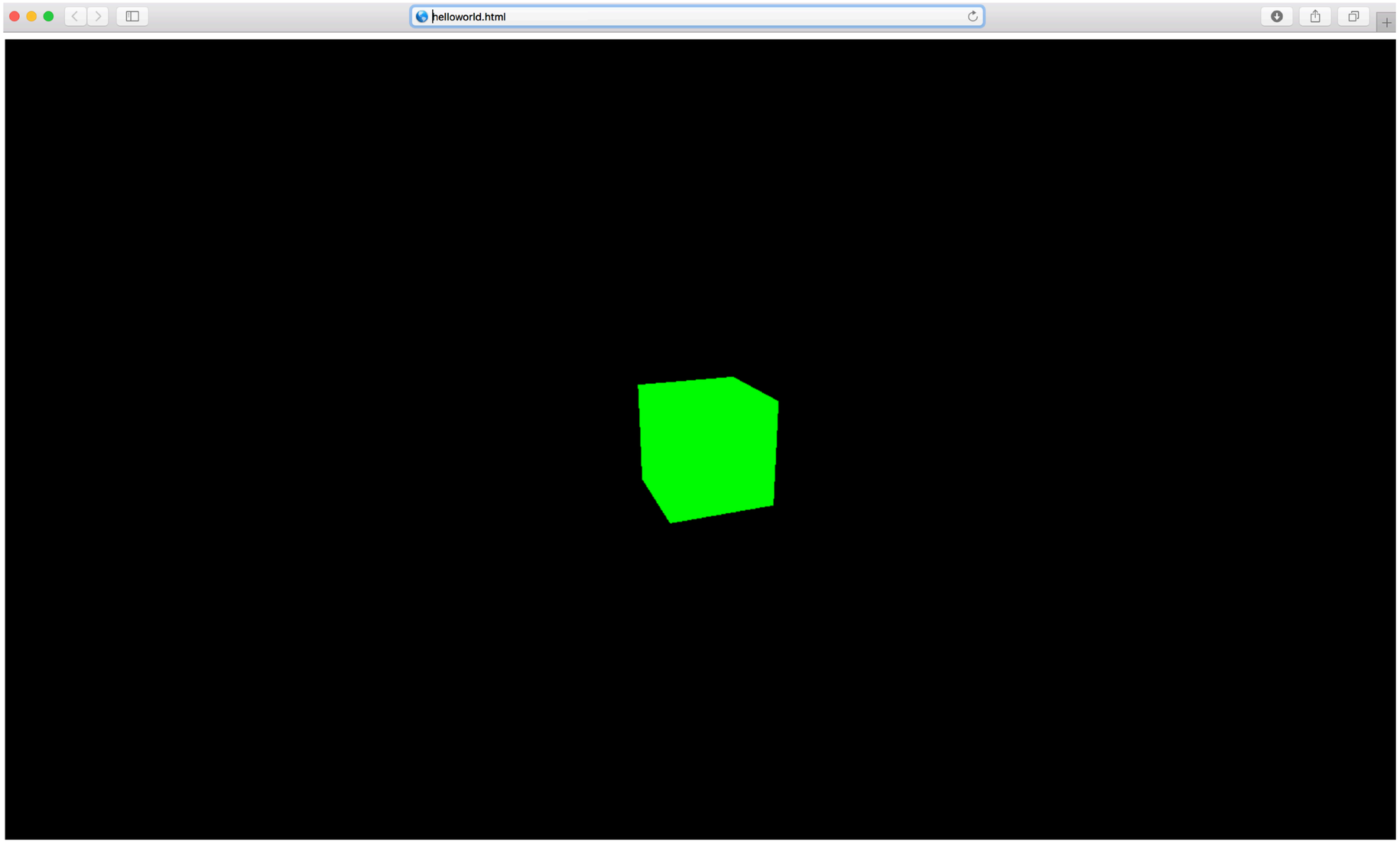
```
function render() {
    requestAnimationFrame( render );
    cube.rotation.x += 0.01;
    cube.rotation.y += 0.01;
    renderer.render(scene, camera);
}
render();
```

Setup Files

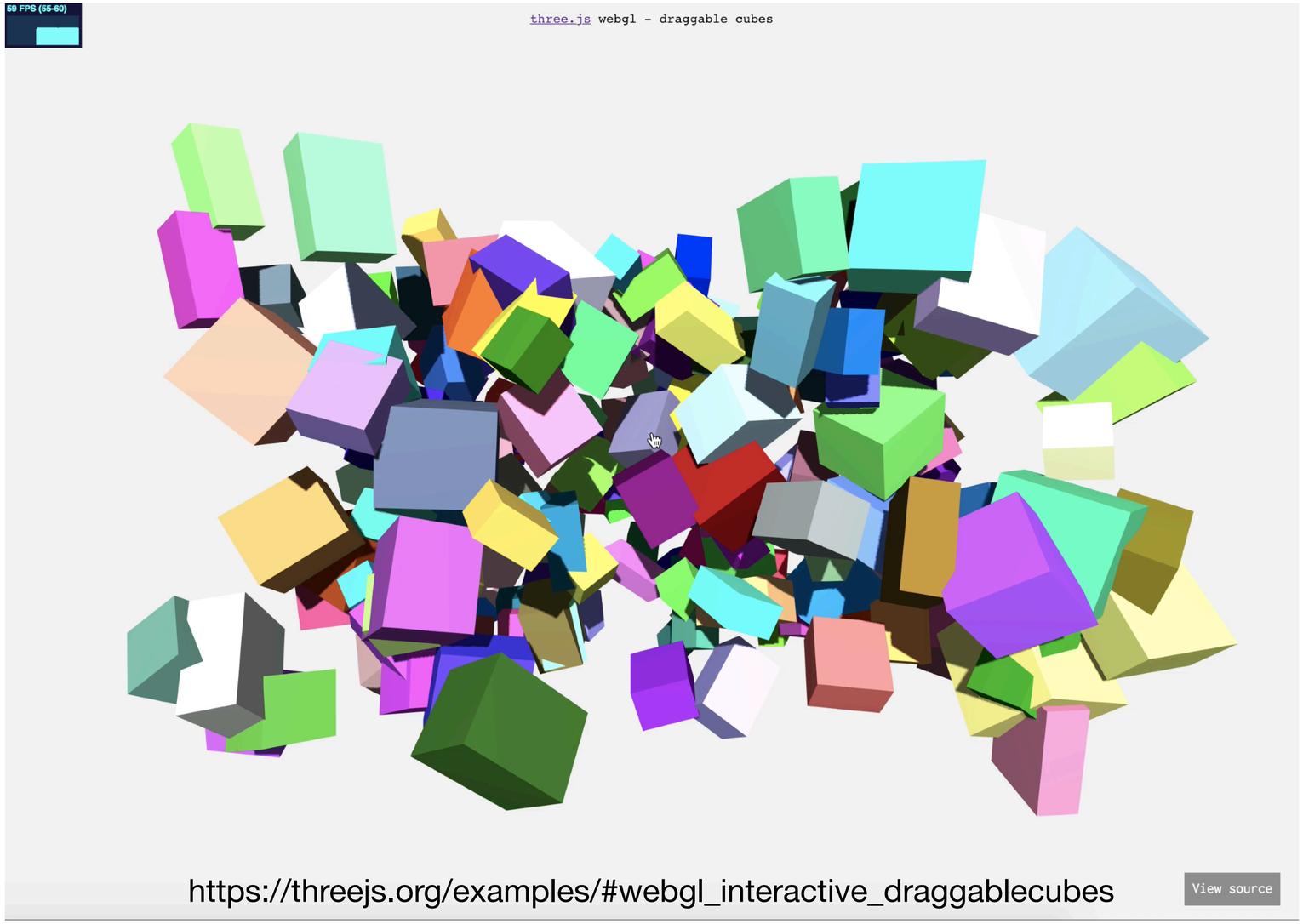
- Put the html and the THREEJS library in the same folder
- If no additional geometries or textures are loaded, html files can be opened directly
- Otherwise: Run files from a local web server:
  - Access e.g. http://localhost/ThreeJSDemo/helloworld.html



Hello Cube: Demo



INTERACTIVE DEMOS



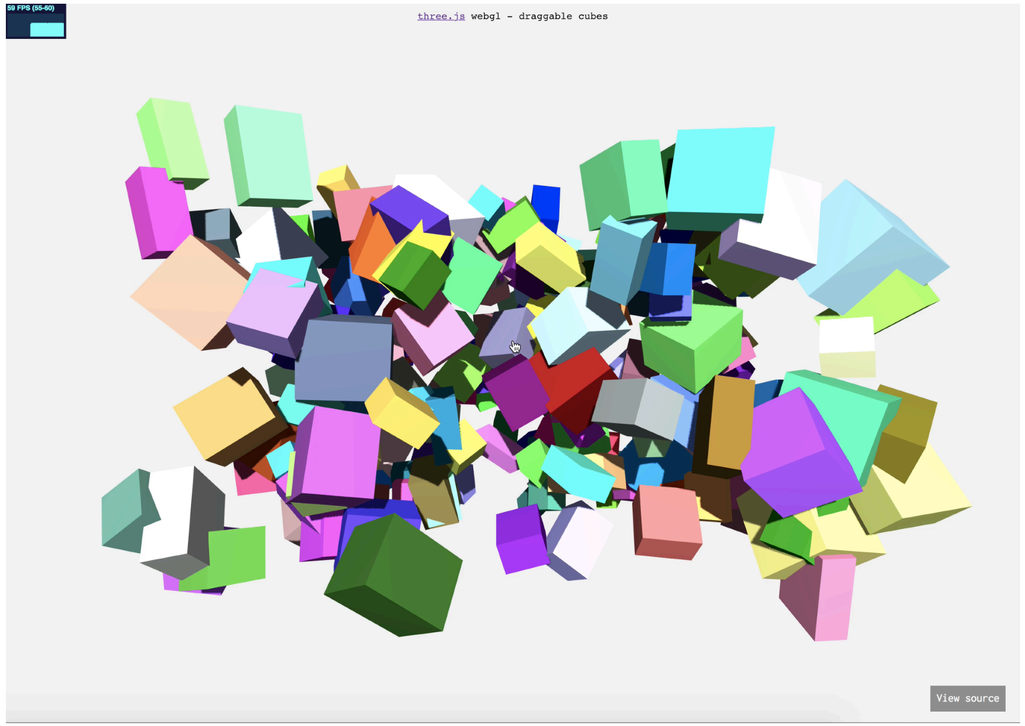
Interactive: Object Picking

- Raycaster computes the intersection of a ray and a set of scene objects
- Raycaster used for mouse picking
- Given a mouse coordinate computes which objects in the 3d space are hit
- Returns an array of intersected objects

```
var raycaster = new THREE.Raycaster();
raycaster.setFromCamera( mouse, camera );
var intersects = raycaster.intersectObjects( scene.children );
```

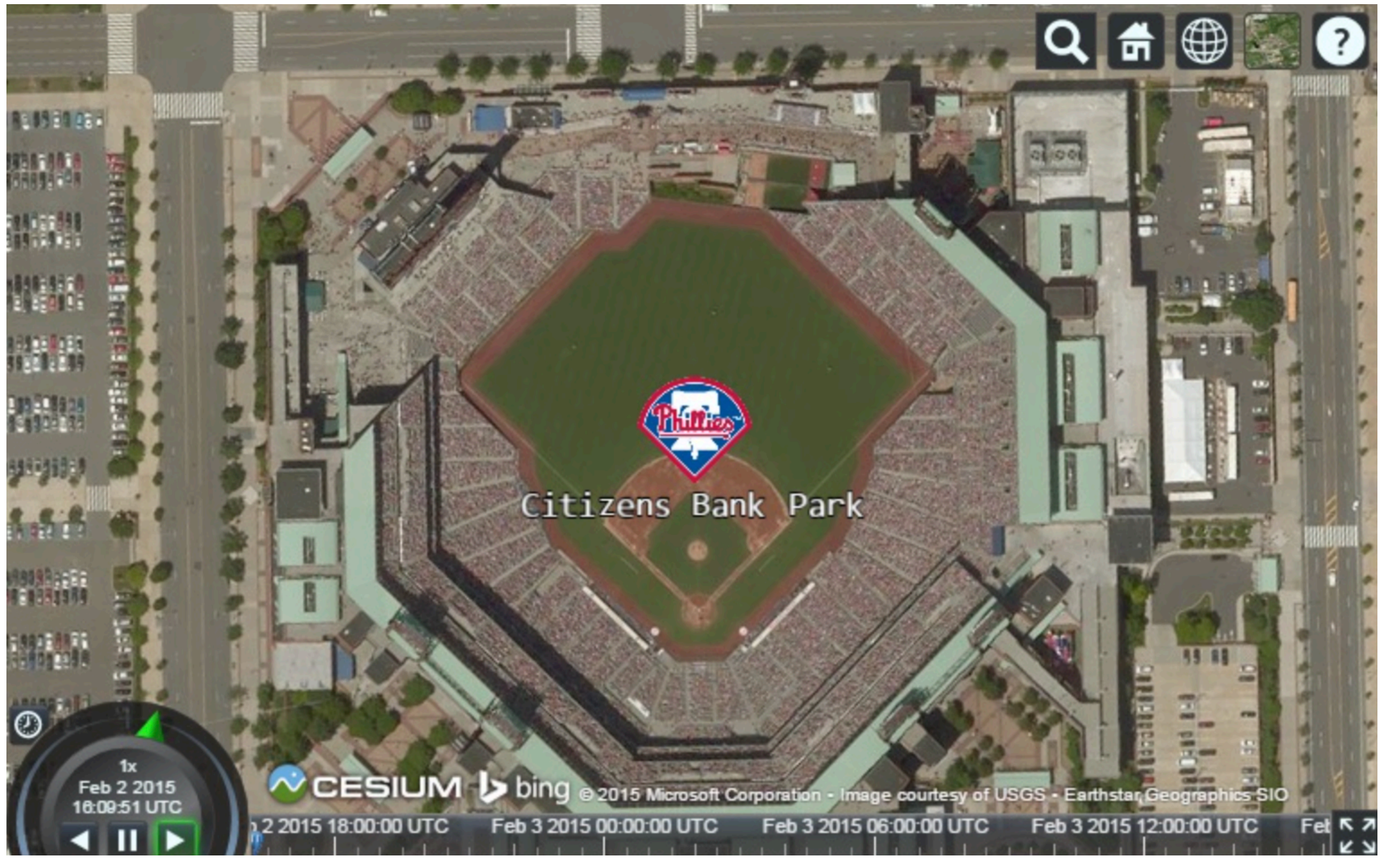
Interactive: Object Picking

```
var raycaster = new THREE.Raycaster();
var mouse = new THREE.Vector2();

function onMouseMove( event ) {
	// calculate mouse position in normalized device coordinates
	mouse.x = ( event.clientX / window.innerWidth ) * 2 - 1;
	mouse.y = - ( event.clientY / window.innerHeight ) * 2 + 1;
}
function render() {
	// update the picking ray with the camera and mouse position
	raycaster.setFromCamera( mouse, camera );
	// calculate objects intersecting the picking ray
	var intersects = raycaster.intersectObjects( scene.children );
	for ( var i = 0; i < intersects.length; i++ ) {
		intersects[ i ].object.material.color.set( 0xff0000 );
	}
	renderer.render( scene, camera );
}
window.addEventListener( 'mousemove', onMouseMove, false );
```

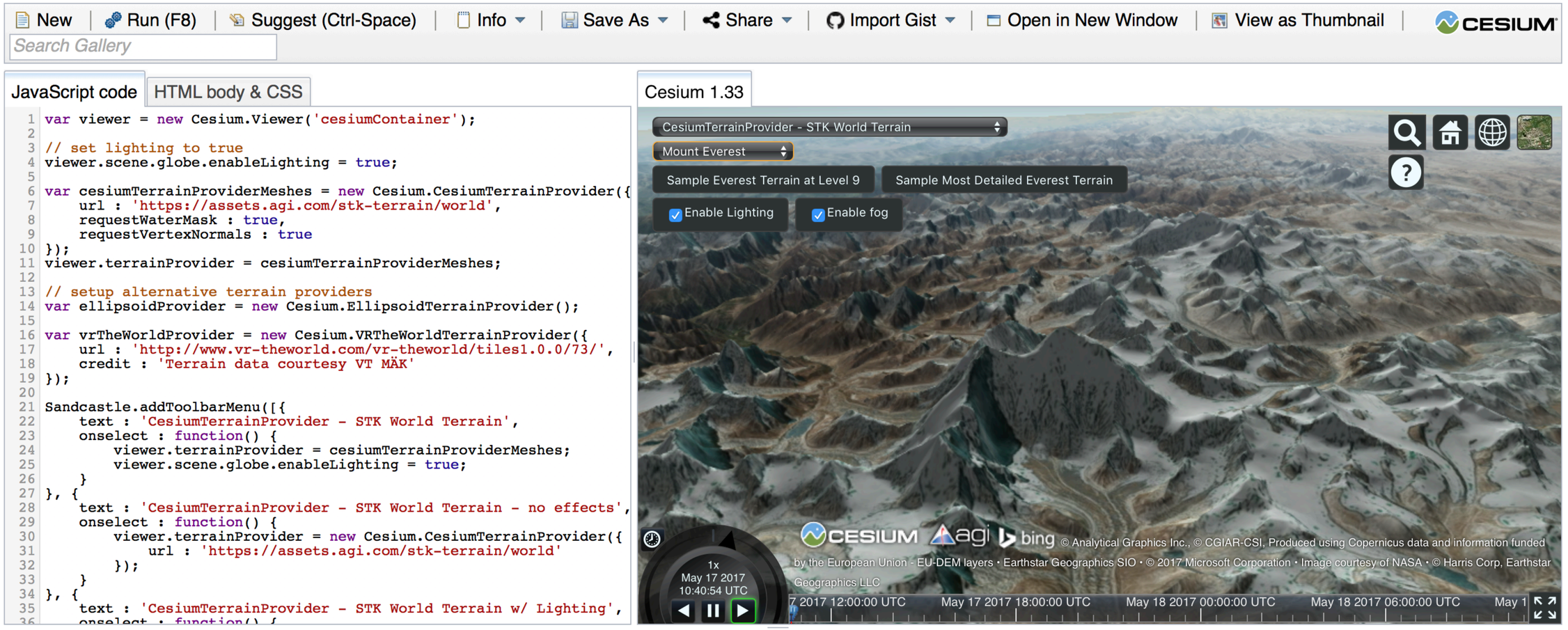


Other webGL frameworks

- Cesium
  - Open-source WebGL framework for rendering globes and maps
  - Focus on visualising dynamic data
  - Support of labels, billboards
  - Sandbox



Other WebGL frameworks



https://cesiumjs.org/Cesium/Apps/Sandcastle/index.html?src=Terrain.html&label=undefined

Other webGL frameworks

- Babylon.js:
  - JavaScript framework for building 3D games with HTML 5 and WebGL
  - Focus on games
  - Features like physics engine, shadows, assets management

https://www.babylonjs.com/Demos/Flat2009/

WEBXR

- API is for accessing virtual reality (VR) and augmented reality (AR) devices on the Web
- Includes sensors and head-mounted displays
- Successor of WebVR API
- Often used in combination with WebGL frameworks

	Headset Devices	Handheld Device e.g. Phone
VR	VR Devices, previously handled by WebVR	Magic Window Behaviour
AR	Mixed Reality Headsets	Phone AR

https://github.com/immersive-web/webxr

https://immersive-web.github.io/webxr/

WEBXR

Supported devices

- ARCore-compatible devices
- Google Daydream
- HTC Vive
- Magic Leap One
- Microsoft Hololens
- Oculus Rift
- Samsung Gear VR
- Windows Mixed Reality headsets

	Headset Devices	Handheld Device e.g. Phone
VR	VR Devices, previously handled by WebVR	Magic Window Behaviour
AR	Mixed Reality Headsets	Phone AR

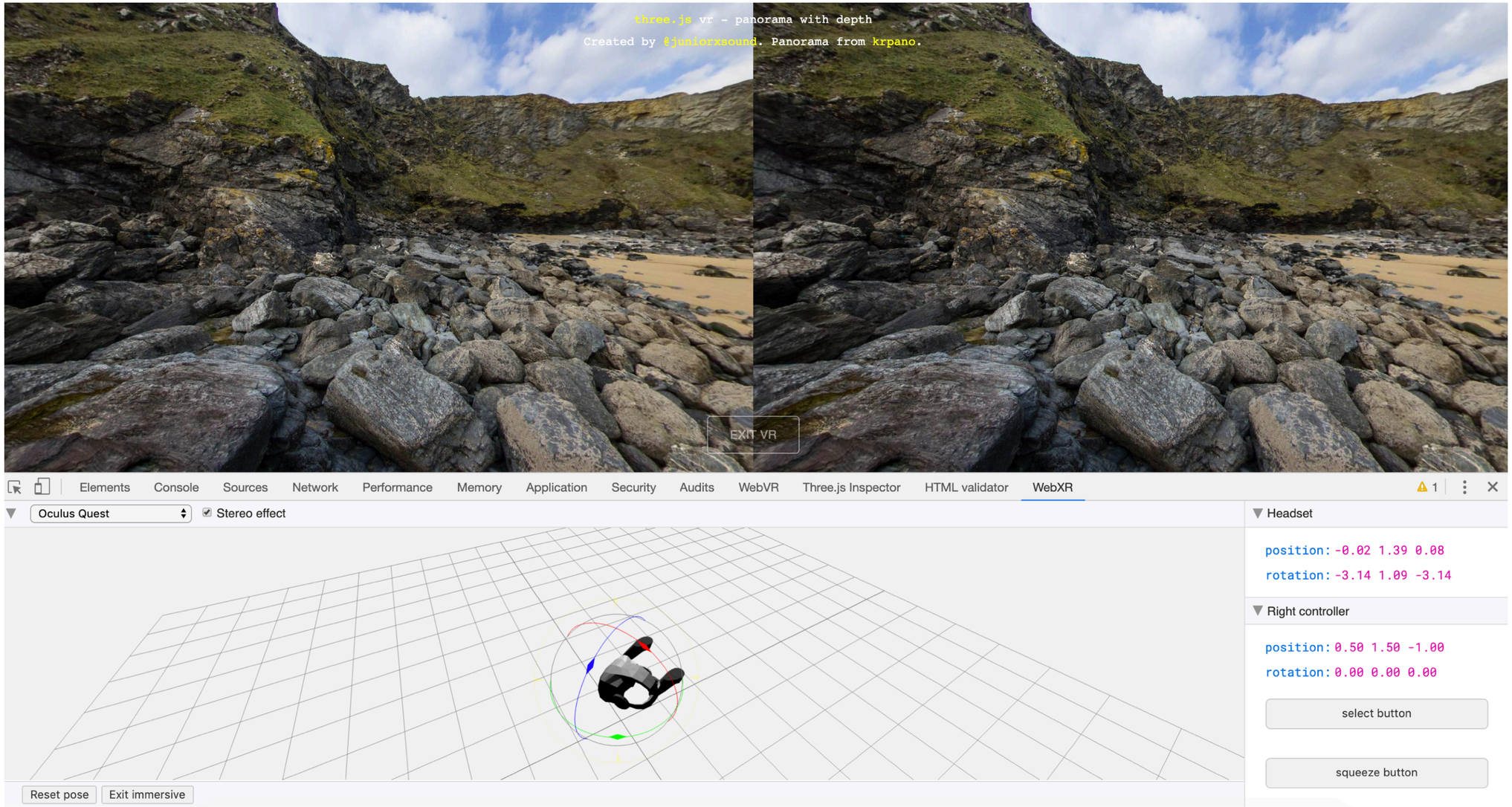
https://github.com/immersive-web/webxr

https://immersive-web.github.io/webxr/

WEBXR - Getting Started

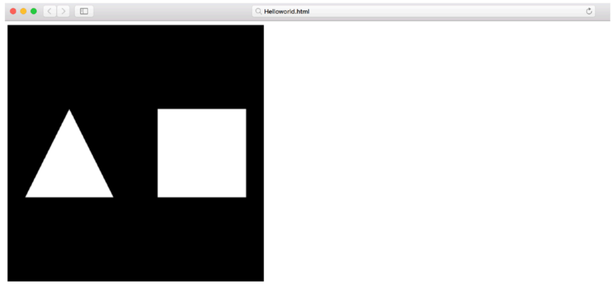
```
scene = new THREE.Scene();

var light = new THREE.AmbientLight( 0xffffff, 1 );
scene.add( light );

camera = new THREE.PerspectiveCamera( 70, window.innerWidth / window.innerHeight, 1, 2000 );
scene.add( camera );

// Create the panoramic sphere geometery
var panoSphereGeo = new THREE.SphereBufferGeometry( 6, 256, 256 );


renderer = new THREE.WebGLRenderer();
renderer.setPixelRatio( window.devicePixelRatio );
renderer.setSize( window.innerWidth, window.innerHeight );
renderer.xr.enabled = true;
container.appendChild( renderer.domElement );

document.body.appendChild( VRButton.createButton( renderer, { referenceSpaceType: 'local' } ) );
```

WEBXR - Simulator



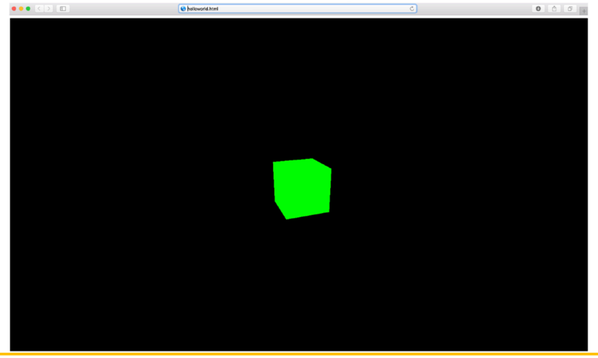
Summary

WEBGL



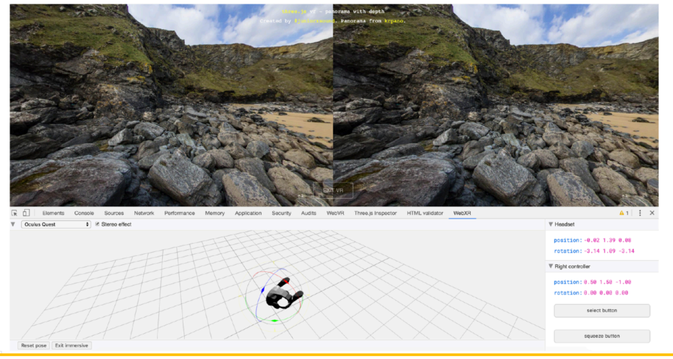
WebGL

THREE.JS



FRAMEWORKS

WEBXR - SIMULATOR



WEBXR

The end!