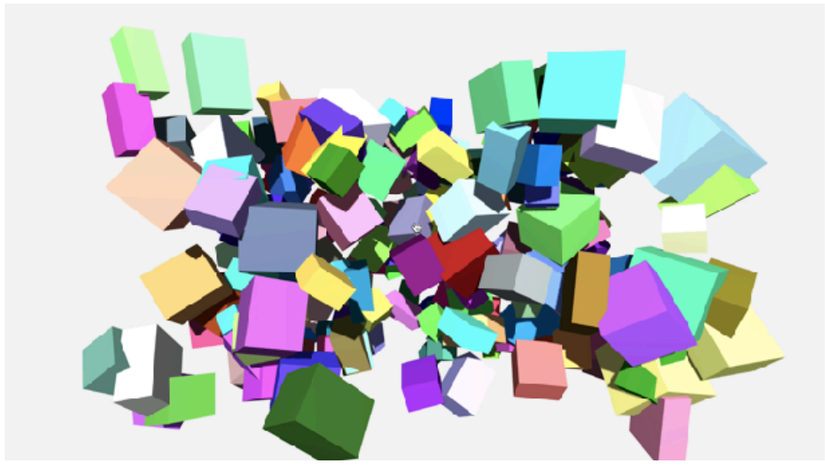
COSC342: Computer Graphics

2017

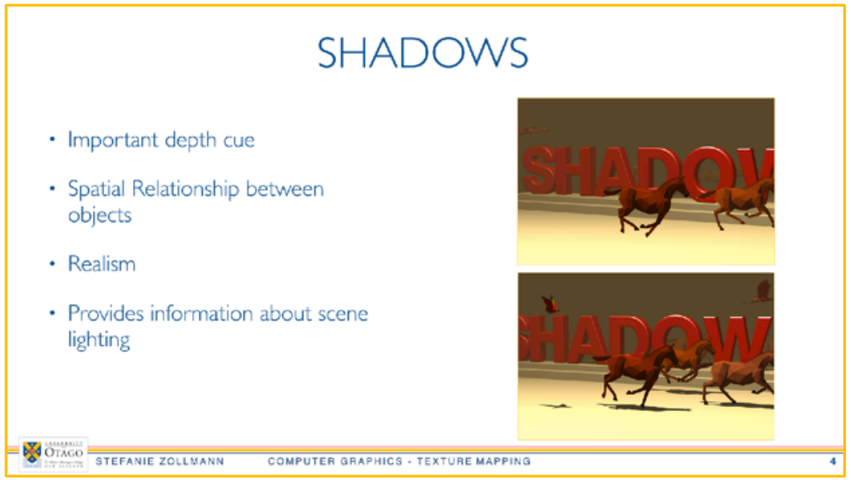


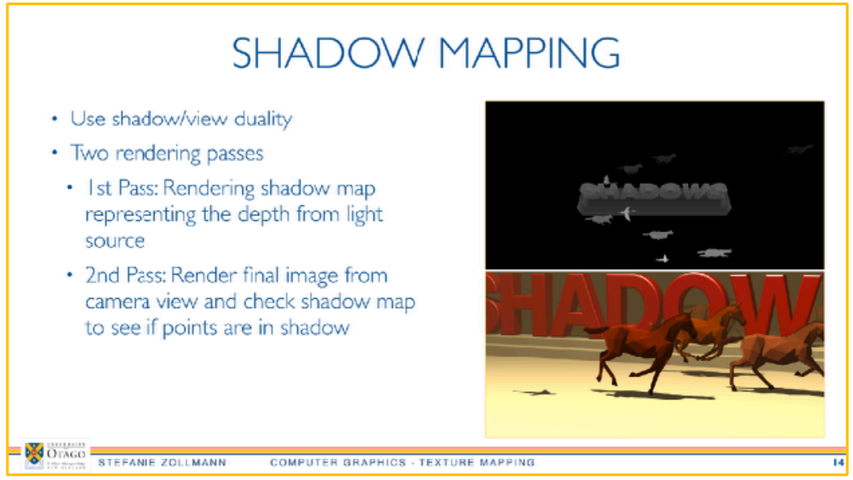
Lecture 21

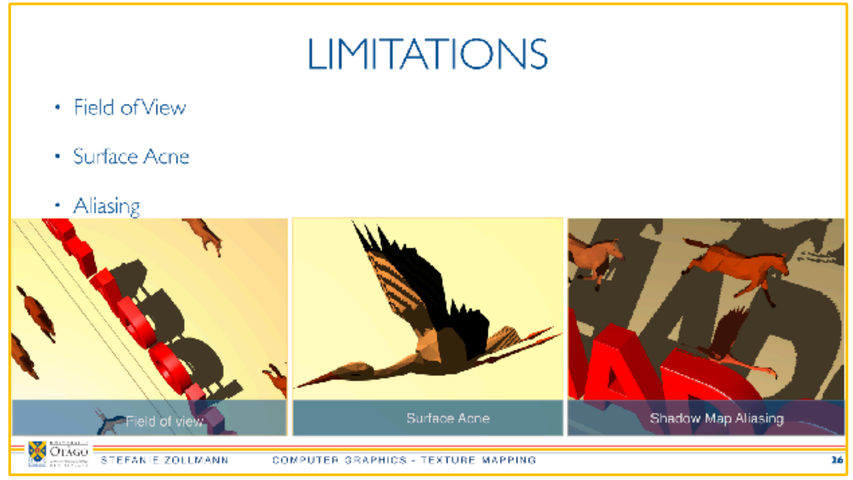
WebGL

Stefanie Zollmann

LAST TIME





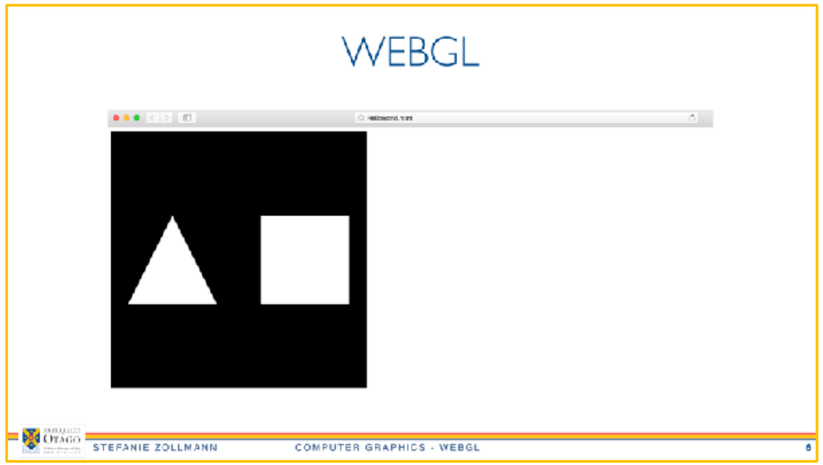


Shadows

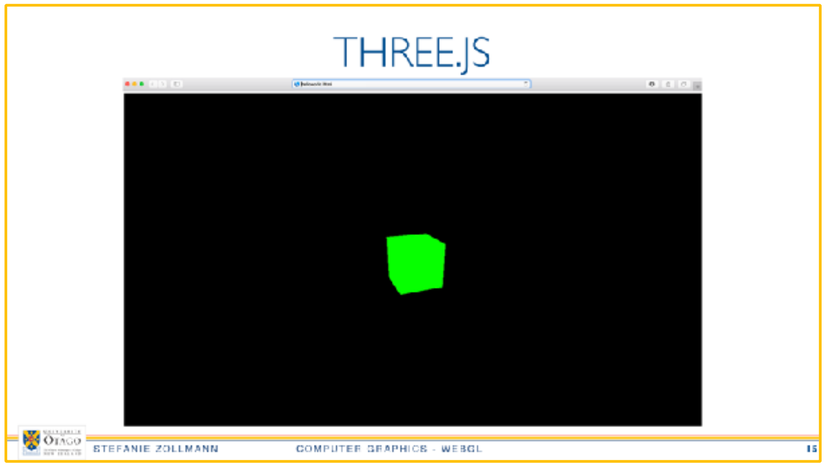
Methods

Limitations & Improvements

TODAY



WEBGL



THREE.JS

INTERACTIVE DEMOS

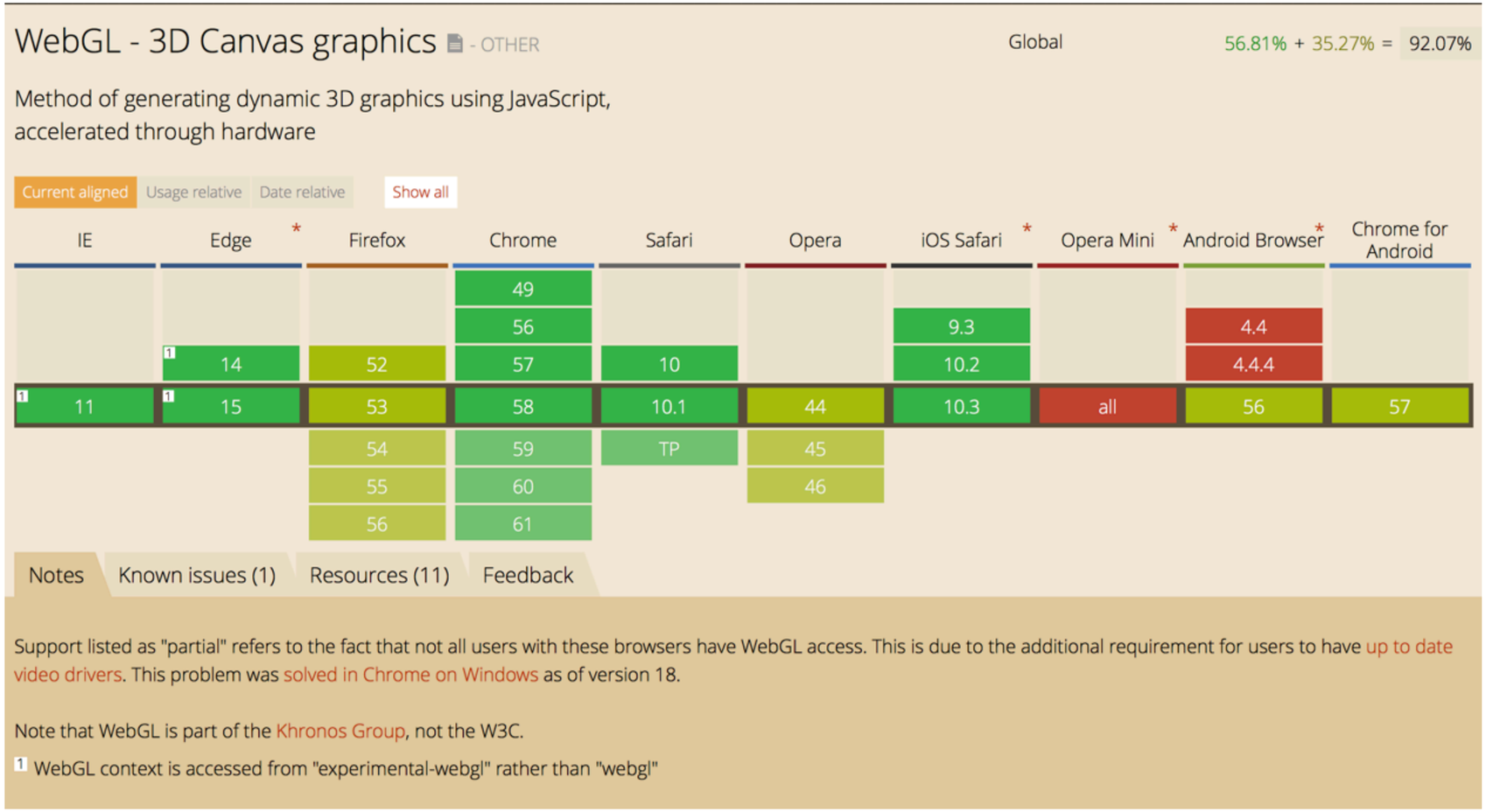
WEBGL

- Derivative of OpenGL
- More specifically, OpenGL ES version 2.0
- 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

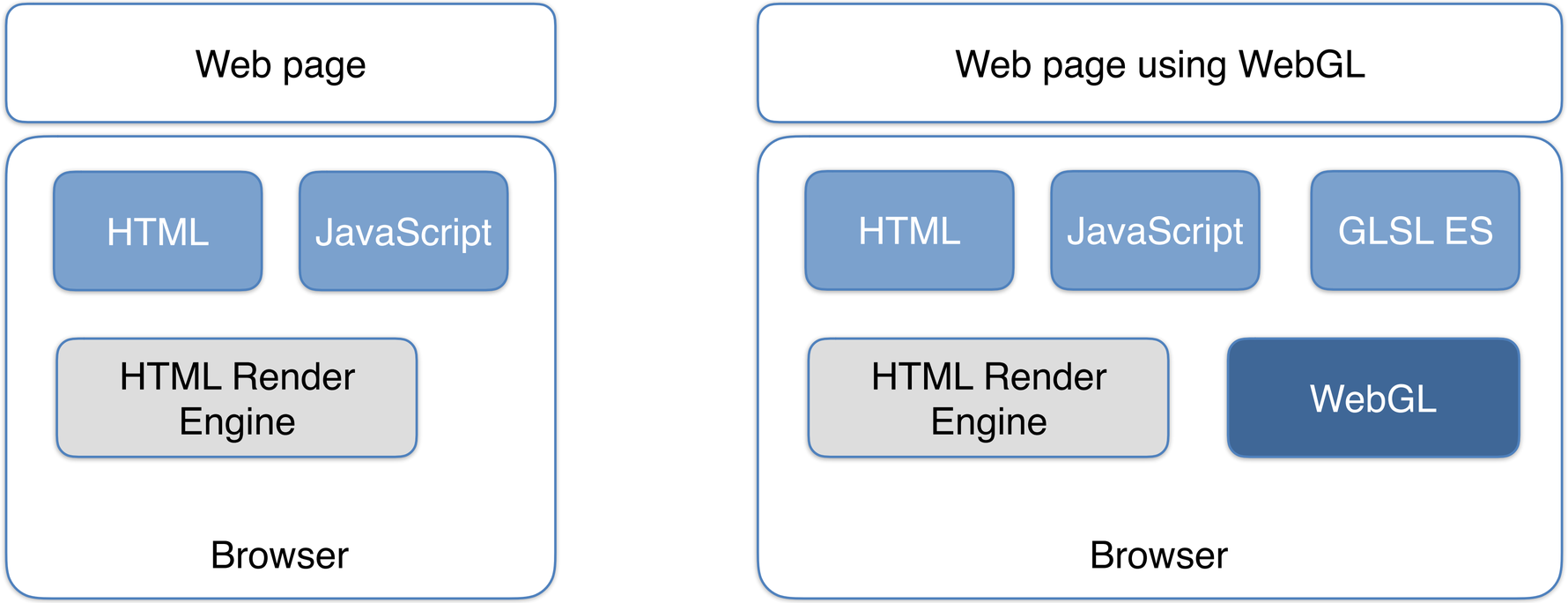
- Inside an HTML <canvas> element.
- Development is likely to involve HTML, CSS, JavaScript, GLSL, in addition to WebGL's OpenGL-style naming.

SUPPORT OF WEBGL

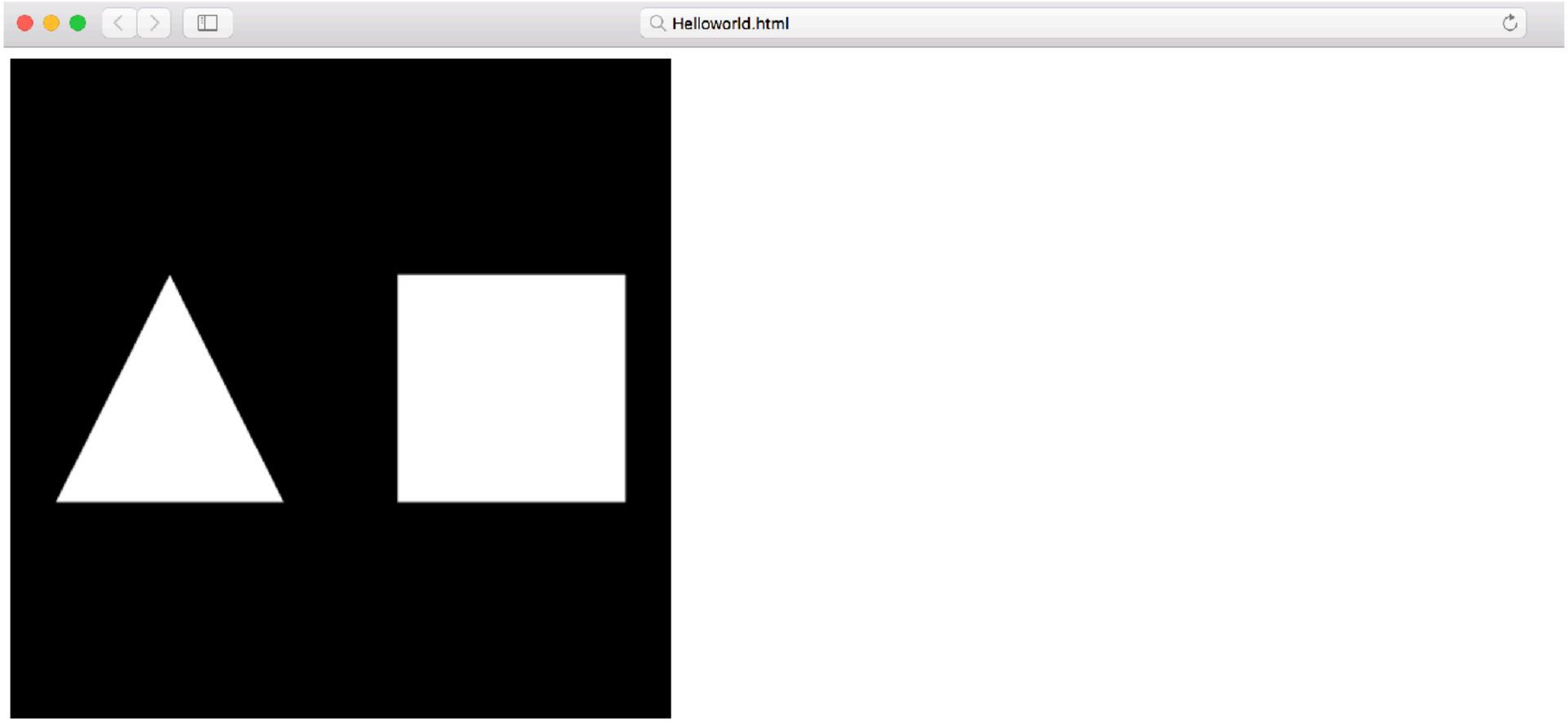


http://caniuse.com/#feat=webgl

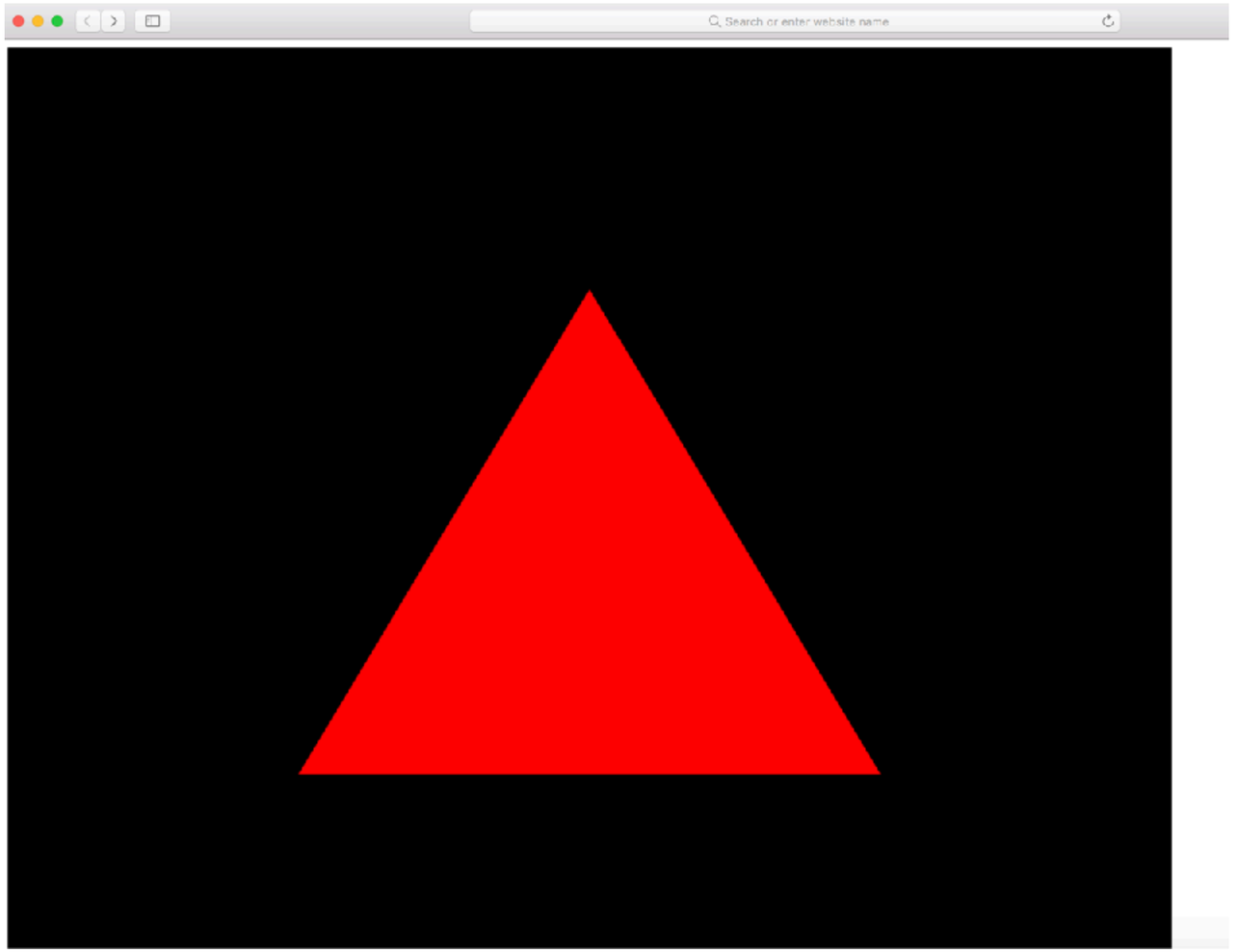
WEBGL APPLICATION STRUCTURE



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

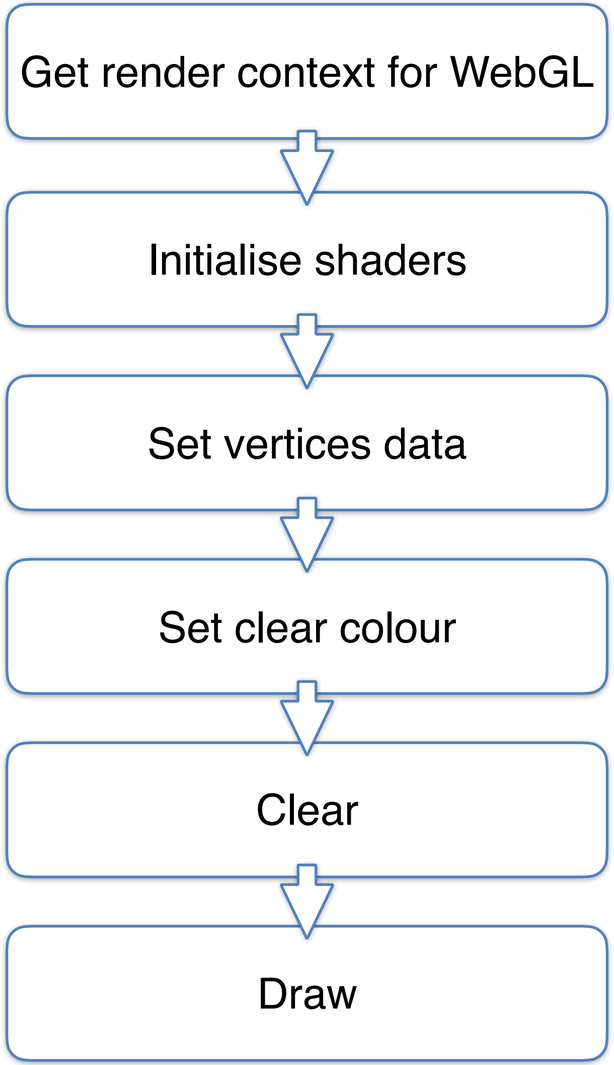
WEBGL "HELLO TRIANGLE"

```
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

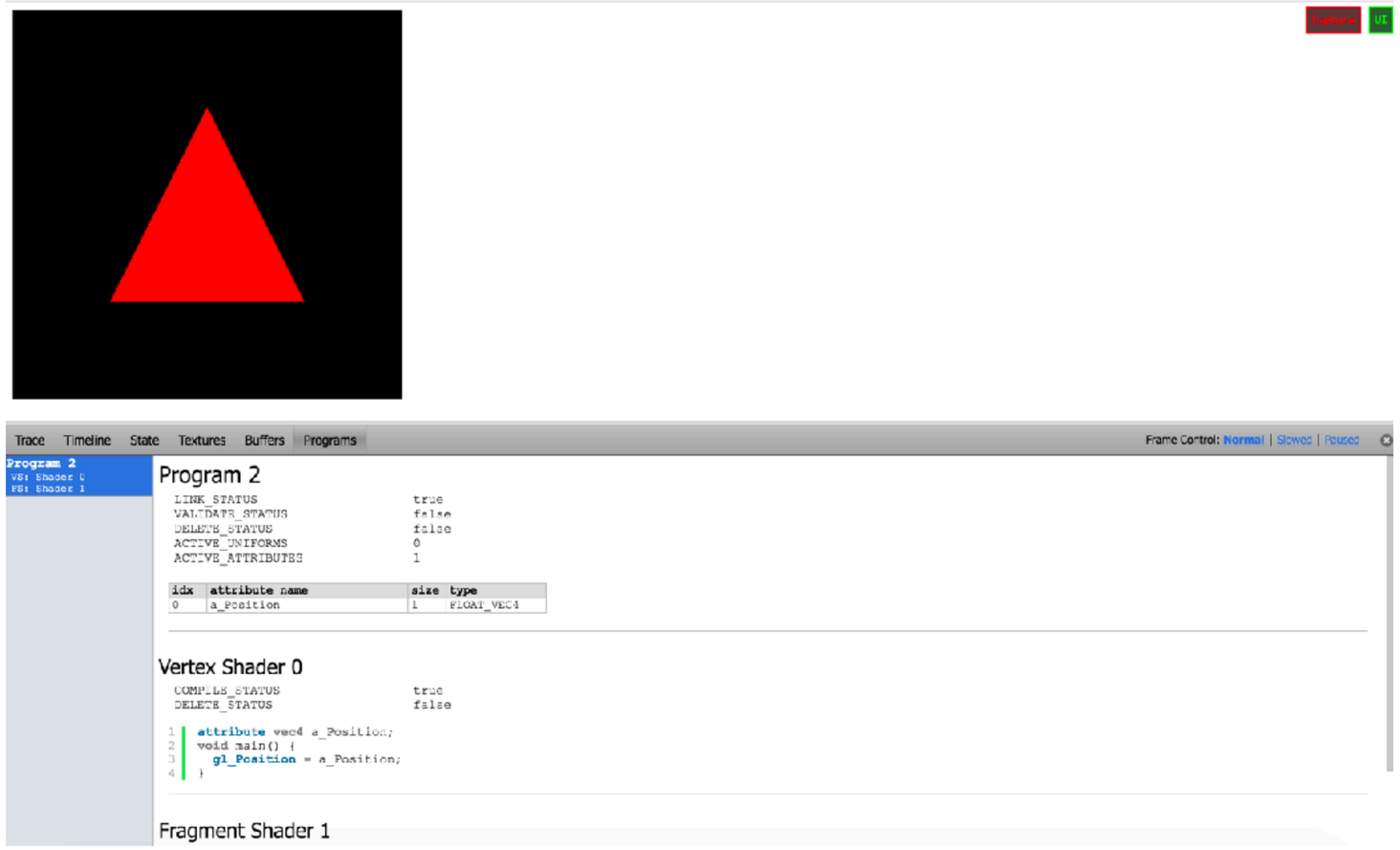
WEBGL SHADERS

```
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



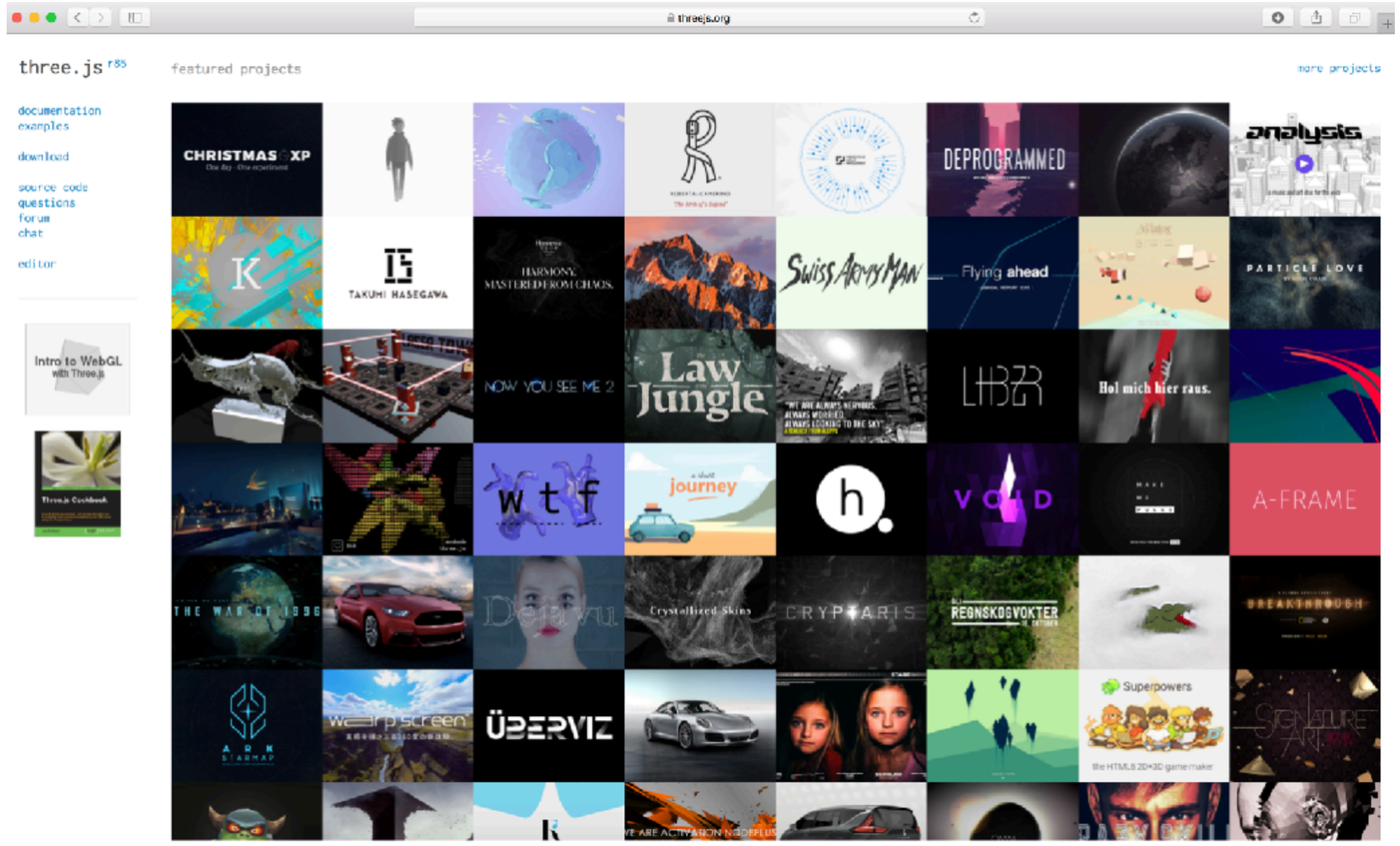
WebGL inspector

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 WebGL
- Website: threejs.org
- Library is in single javascript file



THREE.JS

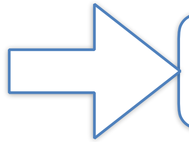
- Features:
  - Scenes: For adding and removing objects during runtime
  - Cameras: Different camera models (perspective, orthographic)
  - Lights: Ambient, direction, point and spot lights
  - Shadows: Control shadow casting and receiving
  - Materials: Phong, textures
  - Shaders: GLSL
  - Objects: 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;
```

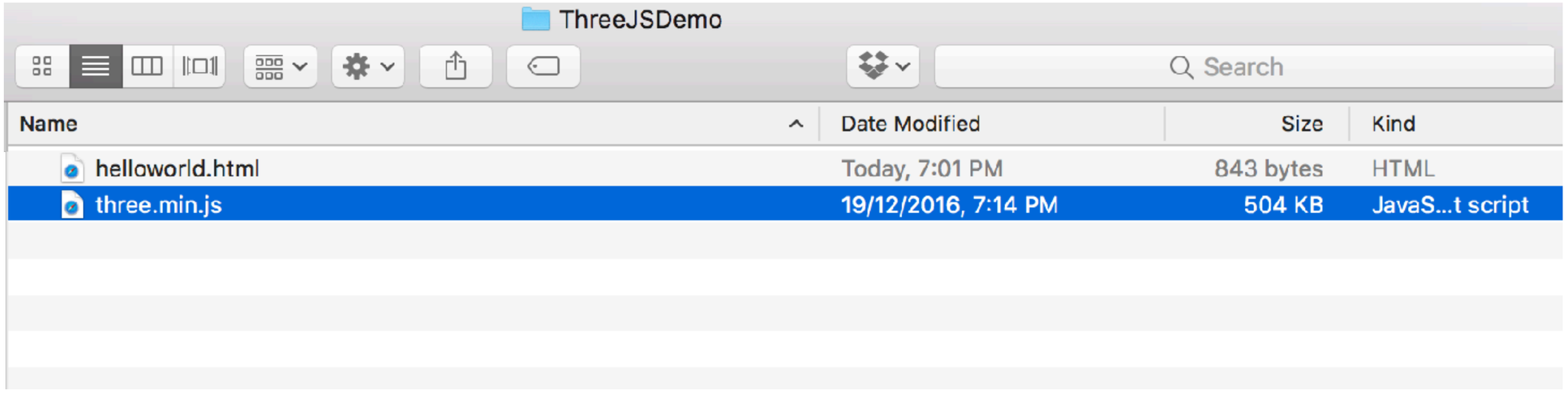
HELLO CUBE: RENDER LOOP

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

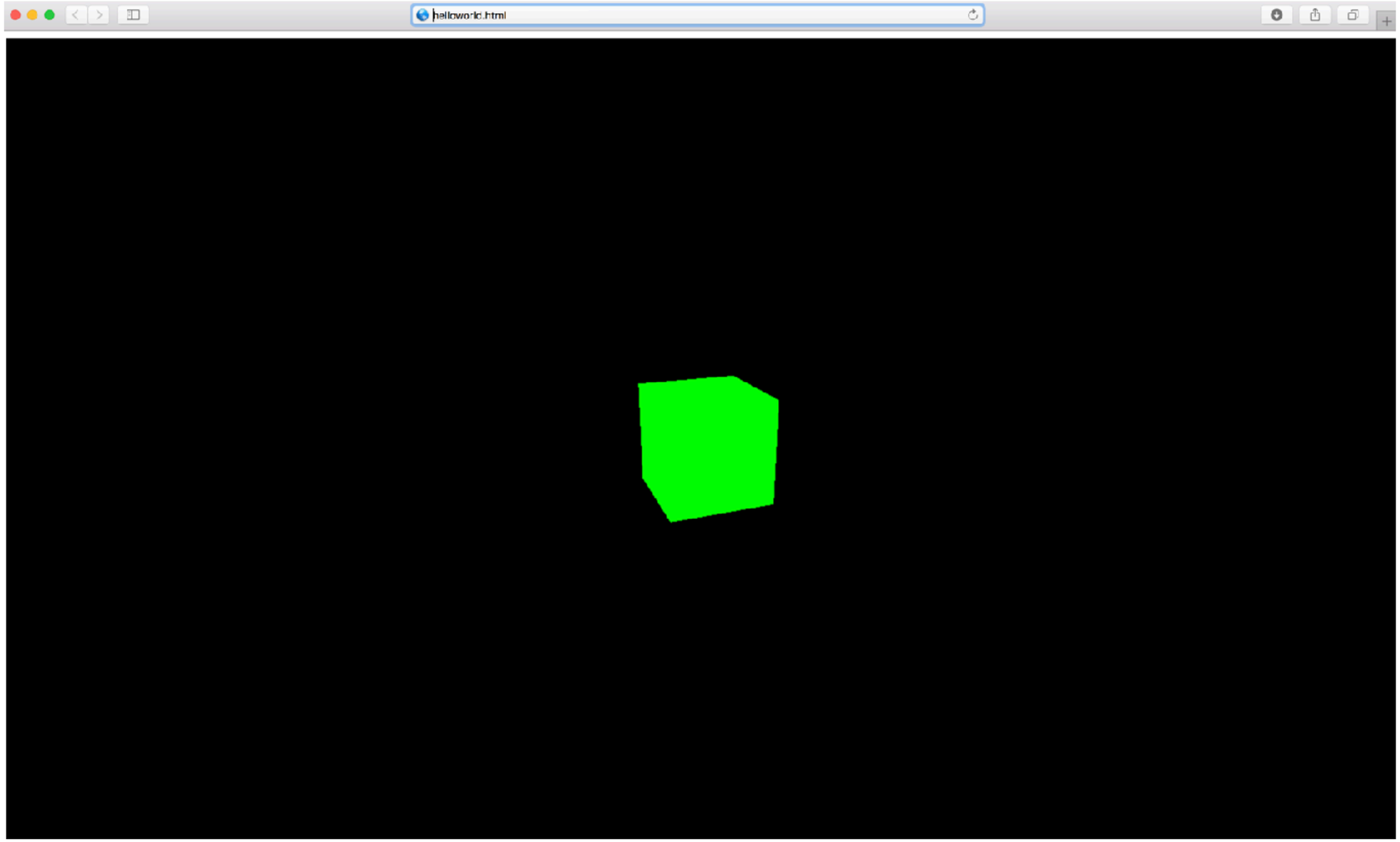
```
function render() {
    requestAnimationFrame( render );
    cube.rotation.x += 0.01;
    cube.rotation.y += 0.01;
    renderer.render(scene, camera);
}
render();
```

SETUP FILES

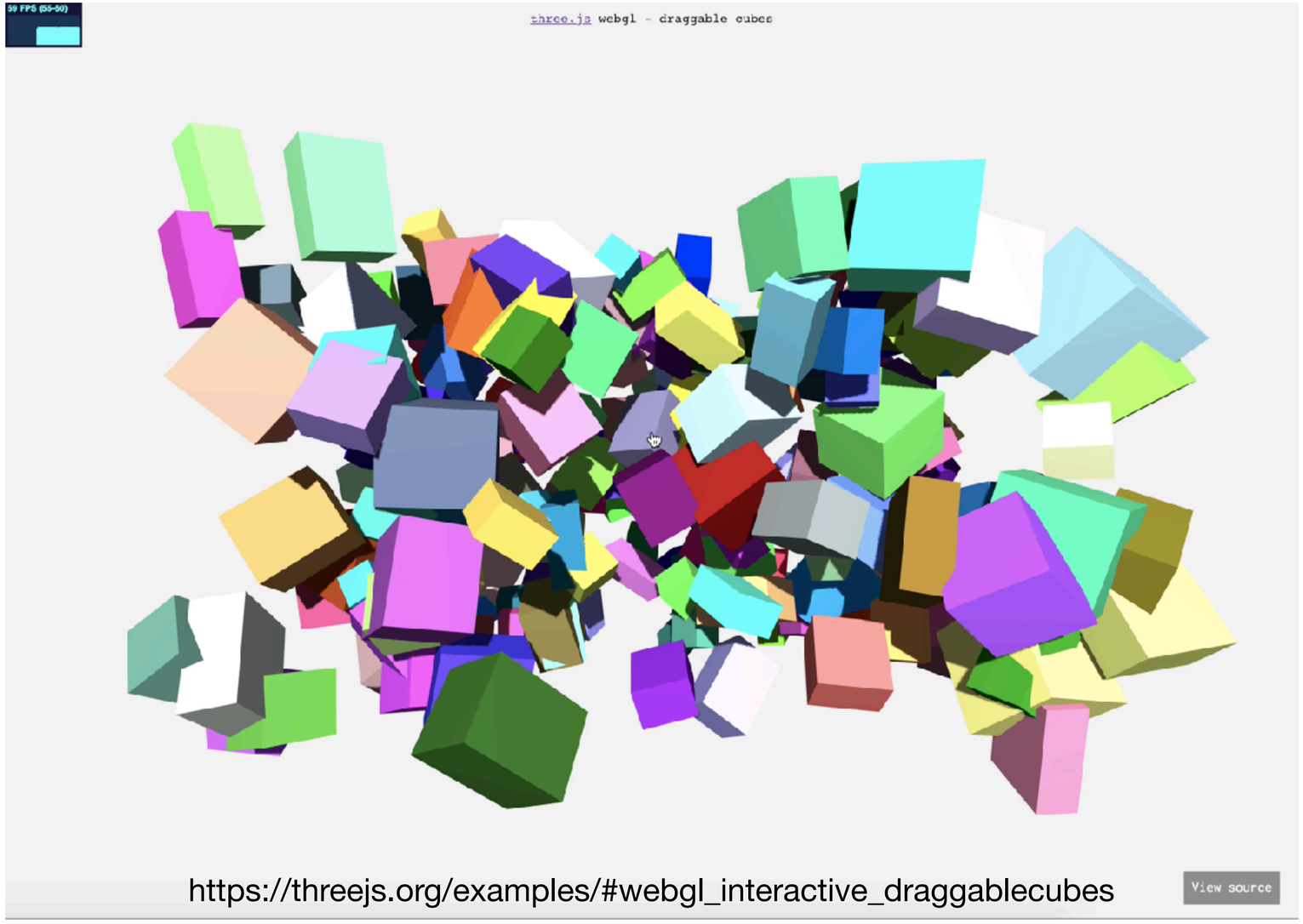
- Put the html and the THREEJS library in the same folder
- If no additional geometries or textures are loaded, html files can be opened directly
- Otherwise: Run files from a local web server:
  - Access e.g. http://localhost/ThreeJSDemo/helloworld.html



HELLO CUBE: DEMO



INTERACTIVE DEMOS



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

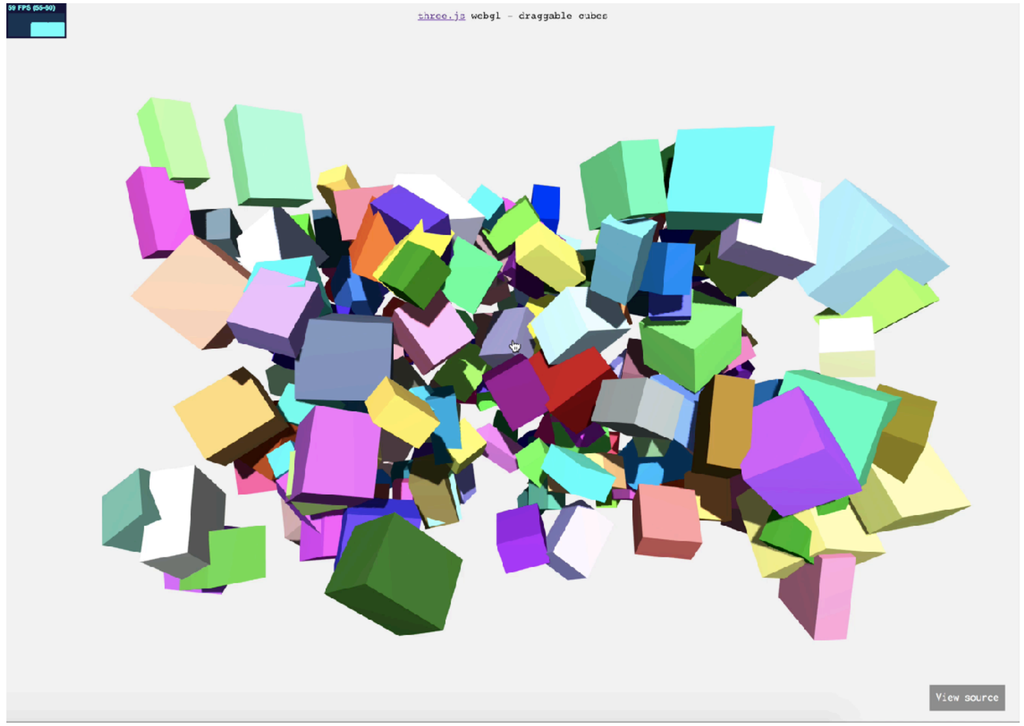
INTERACTIVE: OBJECT PICKING

- 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 );
```

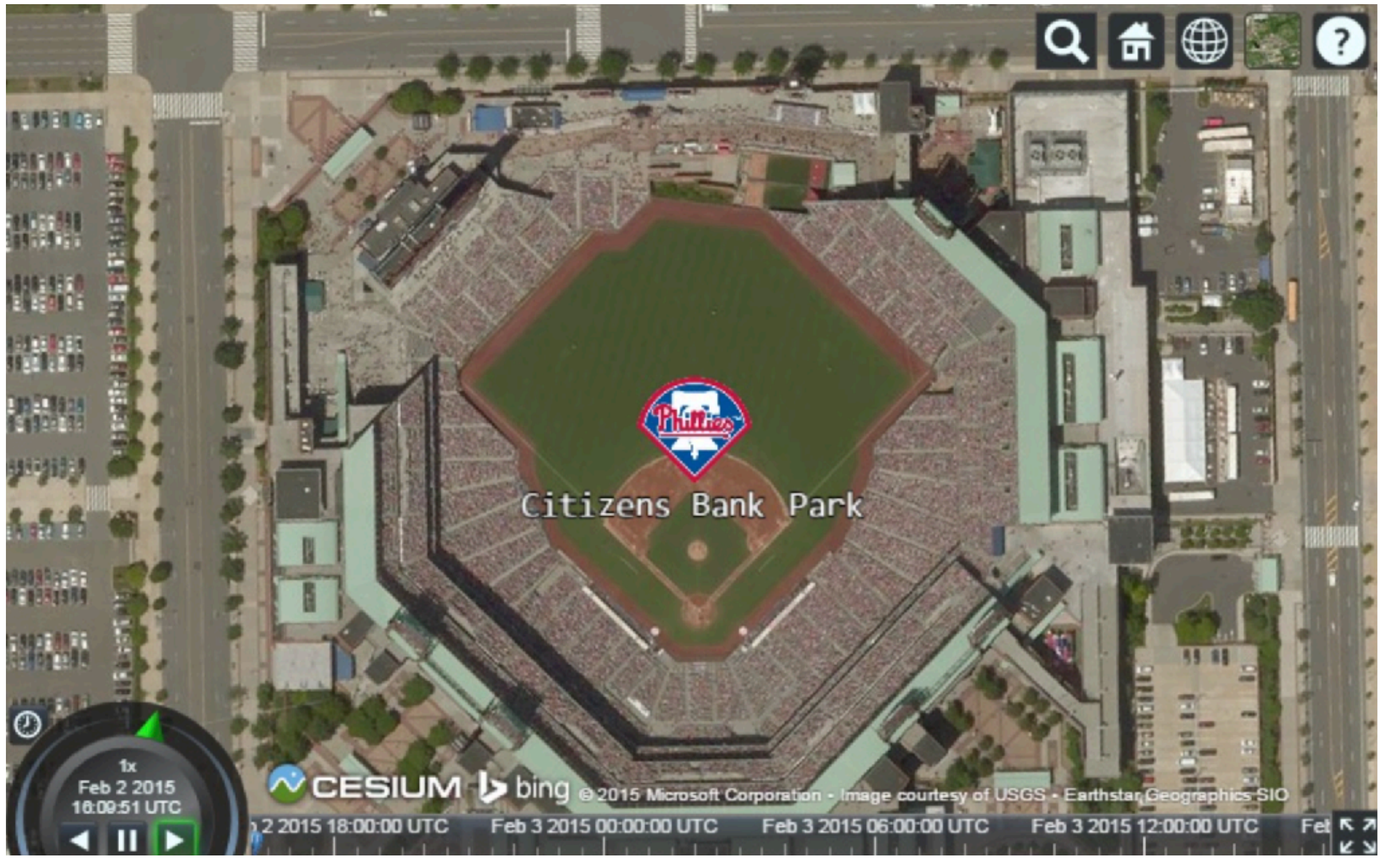
INTERACTIVE: OBJECT PICKING

```
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

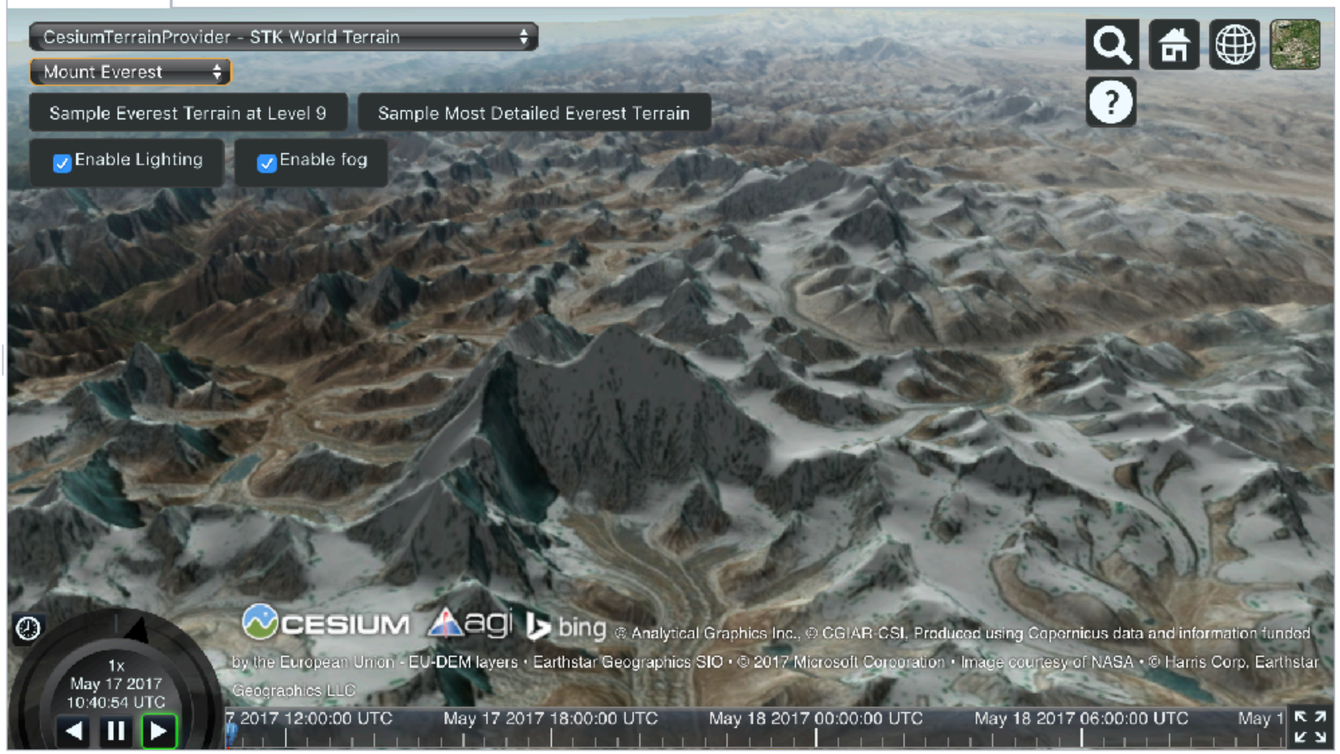
New | Run (F8) | Suggest (Ctrl-Space) | Info | Save As | Share | Import Gist | Open in New Window | View as Thumbnail | CESIUM

Search Gallery

JavaScript code | HTML body & CSS

```
var viewer = new Cesium.Viewer('cesiumContainer');

// set lighting to true
viewer.scene.globe.enableLighting = true;

var cesiumTerrainProviderMeshes = new Cesium.CesiumTerrainProvider({
    url : 'https://assets.agi.com/stk-terrain/world',
    requestWaterMask : true,
    requestVertexNormals : true
});
viewer.terrainProvider = cesiumTerrainProviderMeshes;

// setup alternative terrain providers
var ellipsoidProvider = new Cesium.EllipsoidTerrainProvider();

var vrTheWorldProvider = new Cesium.VRTheWorldTerrainProvider({
    url : 'http://www.vr-theworld.com/vr-theworld/tiles1.0.0/73/',
    credit : 'Terrain data courtesy VT MÄK'
});

Sandcastle.addToolbarMenu([{
    text : 'CesiumTerrainProvider - STK World Terrain',
    onselect : function() {
        viewer.terrainProvider = cesiumTerrainProviderMeshes;
        viewer.scene.globe.enableLighting = true;
    }
}, {
    text : 'CesiumTerrainProvider - STK World Terrain - no effects',
    onselect : function() {
        viewer.terrainProvider = new Cesium.CesiumTerrainProvider({
            url : 'https://assets.agi.com/stk-terrain/world'
        });
    }
}, {
    text : 'CesiumTerrainProvider - STK World Terrain w/ Lighting',
```

Cesium 1.33



Gallery | Console (1)

Showcases | Tutorials | Geometries | Beginner | DataSources | CZML | All

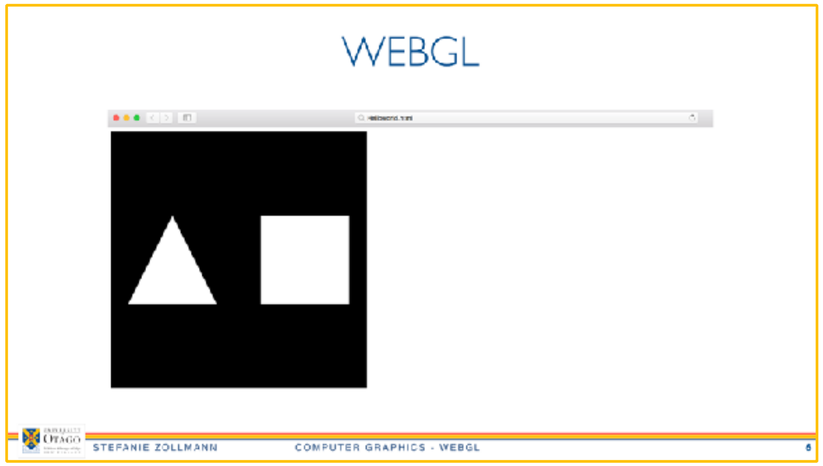
https://cesiumjs.org/Cesium/Apps/Sandcastle/index.html?src=Terrain.html&label=undefined

OTHER WEBGL FRAMEWORKS

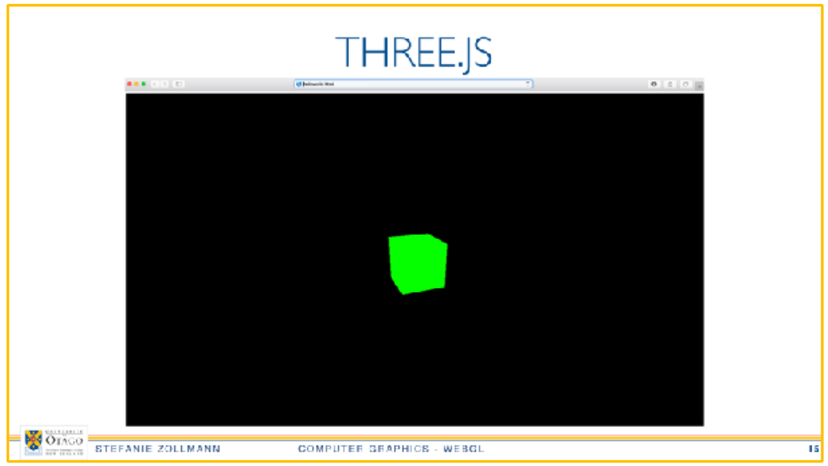
- Babylon.js:
  - JavaScript framework for building 3D games with HTML 5 and WebGL
  - Focus on games
  - Features like physics engine, shadows, assets management

http://www.babylonjs.com/demos/flat2009/

SUMMARY



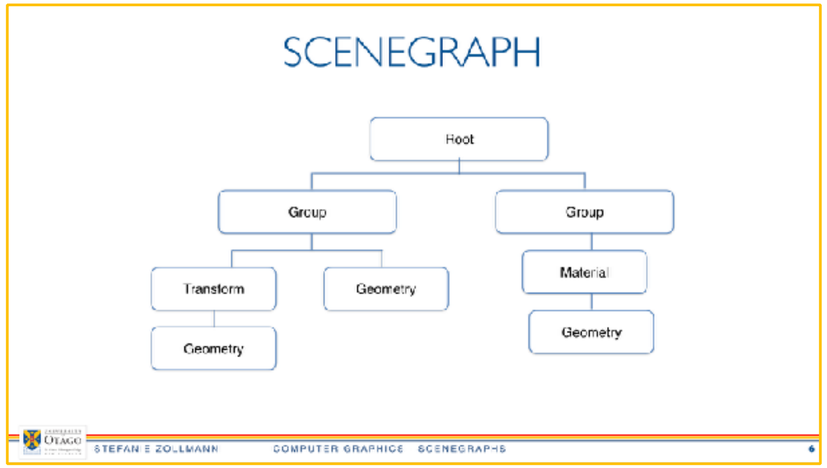
WEBGL

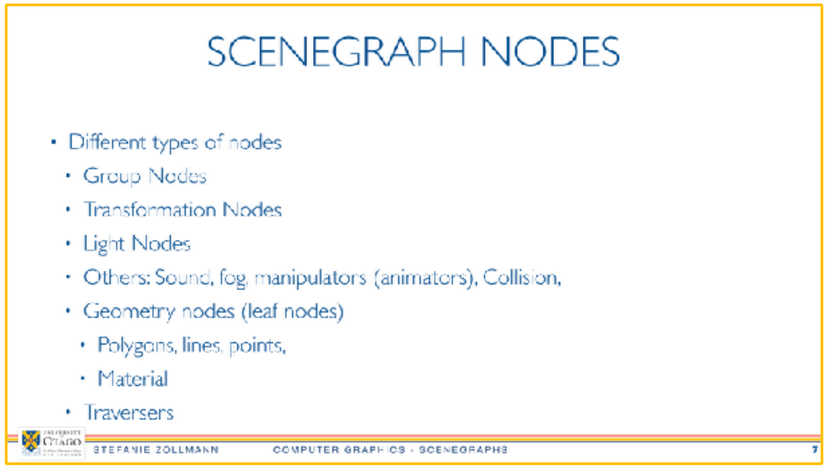


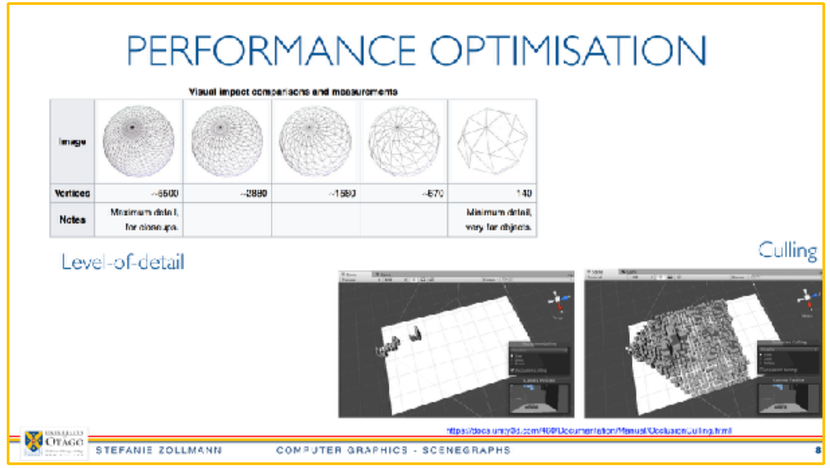
THREE.JS

INTERACTIVE DEMOS

NEXT TIME







SCENEGRAPH NODES OPTIMISATIONS

Thank You!

For more material visit

http://www.cs.otago.ac.nz/cosc342/