3d in the Browser

<canvas>, WebGL, and Three.js

Greg Tatum
webdev for Cubic

gregtatum.com
@TatumCreative

What is WebGL?

A new web standard to leverage the power of the graphics card to draw crazy stuff in the browser.

The control code for WebGL is written in **JavaScript**.

The shading is handled by compiled **shader code**. written in **GLSL** (based on the syntax of **C**)

What is <canvas>?

The canvas element lets you draw on a **bitmap image right within the browser** without effecting any other DOM elements or causing re-paints.

You can render in both **2d and 3d** on canvas using the standard 2d context, or the WebGL context (where supported).

Browser events are limited, so user interaction can be a bit trickier compared to SVG drawing, which does effect the DOM.

What is Three.js?

An open source javascript 3d library that can render scenes using a variety of technologies including WebGL.

It gets rid of a lot of the pain and technical know-how of interacting with a WebGL context.

threejs.org

threejs.org/examples/

threejs.org/docs/

github.com/mrdoob/three.js

Demos

Mapping Small Arms and Ammunition

Visualizing World Population

3d Car Visualizer

Digital Landscapes

Find Your Way To Oz Source Code: https://code.google.com/p/oz-experiment/

Can I Use <canvas>?

Canvas (basic support) - Candidate Recommendation

Method of generating fast, dynamic graphics using JavaScript

*Usage stats:	Global
Support:	81.36%
Partial support:	4.81%
Total:	86.17%

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
								2.1		
								2.2		
						3.2		2.3		
						4.0-4.1		3.0		
	8.0			5.1		4.2-4.3		4.0		
	9.0			6.0		5.0-5.1		4.1		
	10.0	26.0	31.0	6.1		6.0-6.1		4.2-4.3	7.0	
Current	11.0	27.0	32.0	7.0	19.0	7.0	5.0-7.0	4.4	10.0	10.0
Near future		28.0	33.0		20.0					
Farther future		29.0	34.0		21.0					
3 versions ahead		30.0	35.0							

IE8 - http://flashcanvas.net/

Can I Use WebGL?

WebGL - 3D Canvas graphics - Other

Method of generating dynamic 3D graphics using JavaScript, accelerated through hardware

*Usage stats:	Global
Support:	40.83%
Partial support:	21.8%
Total:	62.63%

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
								2.1		
								2.2		
						3.2		2.3		
						4.0-4.1		3.0		
	8.0			5.1		4.2-4.3		4.0		
	9.0			6.0		5.0-5.1		4.1		
	10.0	26.0	31.0	6.1		6.0-6.1		4.2-4.3	7.0	
Current	11.0	27.0	32.0	7.0	19.0	7.0	5.0-7.0	4.4	10.0	10.0
Near future		28.0	33.0		20.0					
Farther future		29.0	34.0		21.0					
3 versions ahead		30.0	35.0							

IE8 - fuggedaboutit

Drawing in 3d

The hard way – working with a WebGL context

Drawing in 3d

The easier way - working with Three.js

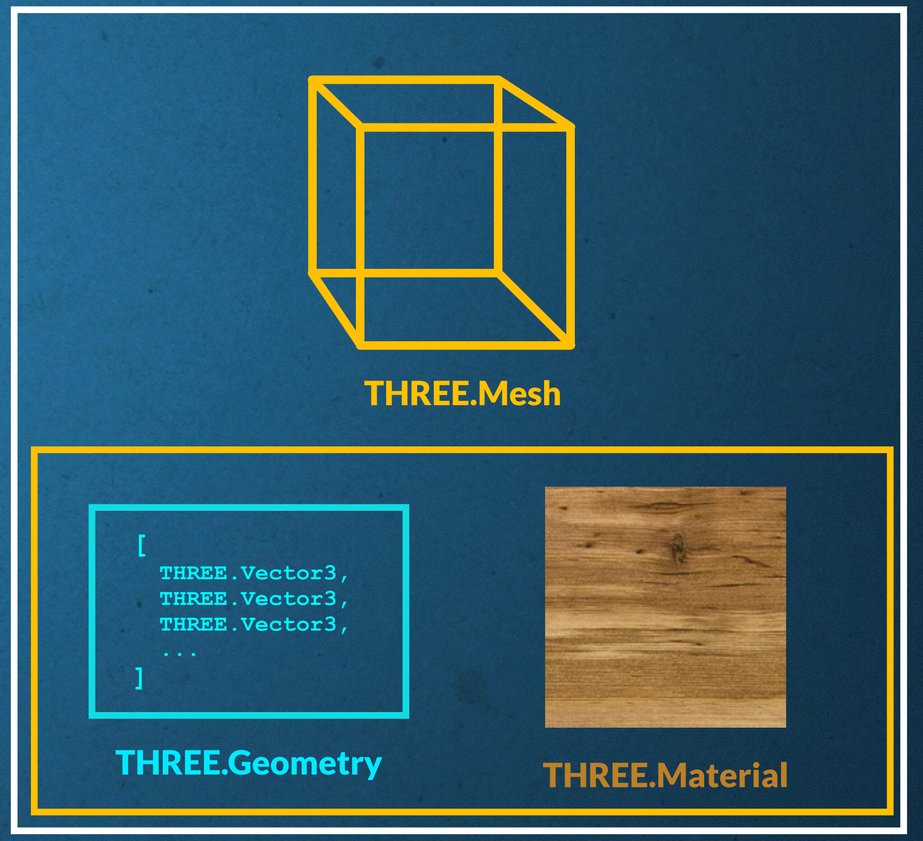
THREE.Scene

THREE.PointLight

THREE.Object3d

THREE.PerspectiveCamera

THREE.Object3d



THREE.Mesh

THREE.Geometry

THREE.Material

THREE.Object3d

Typed Data in JavaScript

```
//Create a buffer of 16 bytes
var buffer = new ArrayBuffer(16);

//ArrayBuffer with a Float32 view
var floats = new Float32Array(4); //16 bytes

// [0, 0, 0, 0]

floats[1];                  // 0

floats[1] = 5745.55;        //5745.55

floats[2] = "67";           //67

floats;                     //[0, 5745.5498046875, 67, 0]
```

ArrayBuffer Views

Type	Size	Description	Equivalent C type
`Int8Array`	1	8-bit twos complement signed integer	`signed char`
`Uint8Array`	1	8-bit unsigned integer	`unsigned char`
`Uint8ClampedArray`	1	8-bit unsigned integer	`unsigned char`
`Int16Array`	2	16-bit twos complement signed integer	`short`
`Uint16Array`	2	16-bit unsigned integer	`unsigned short`
`Int32Array`	4	32-bit twos complement signed integer	`int`
`Uint32Array`	4	32-bit unsigned integer	`unsigned int`
`Float32Array`	4	32-bit IEEE floating point number	`float`
`Float64Array`	8	64-bit IEEE floating point number	`double`

A peek at what I'm working on

EvolveJS.com

Three.js - The Code

github.com/mrdoob/three.js

Three.js - The Site

threejs.org

The WebGL Spec

http://www.khronos.org/registry/webgl/specs/latest/1.0/

The Future

Graphics card leveraged rendering –
the **browser** vs **native environments**

Imagine some applications

- 3d visualization of open data
- WebSockets for realtime interaction
- Universal gaming not tied to an app store
- GPGPU - Super computing in the browser
- What other distributed collaborative networks can be created?