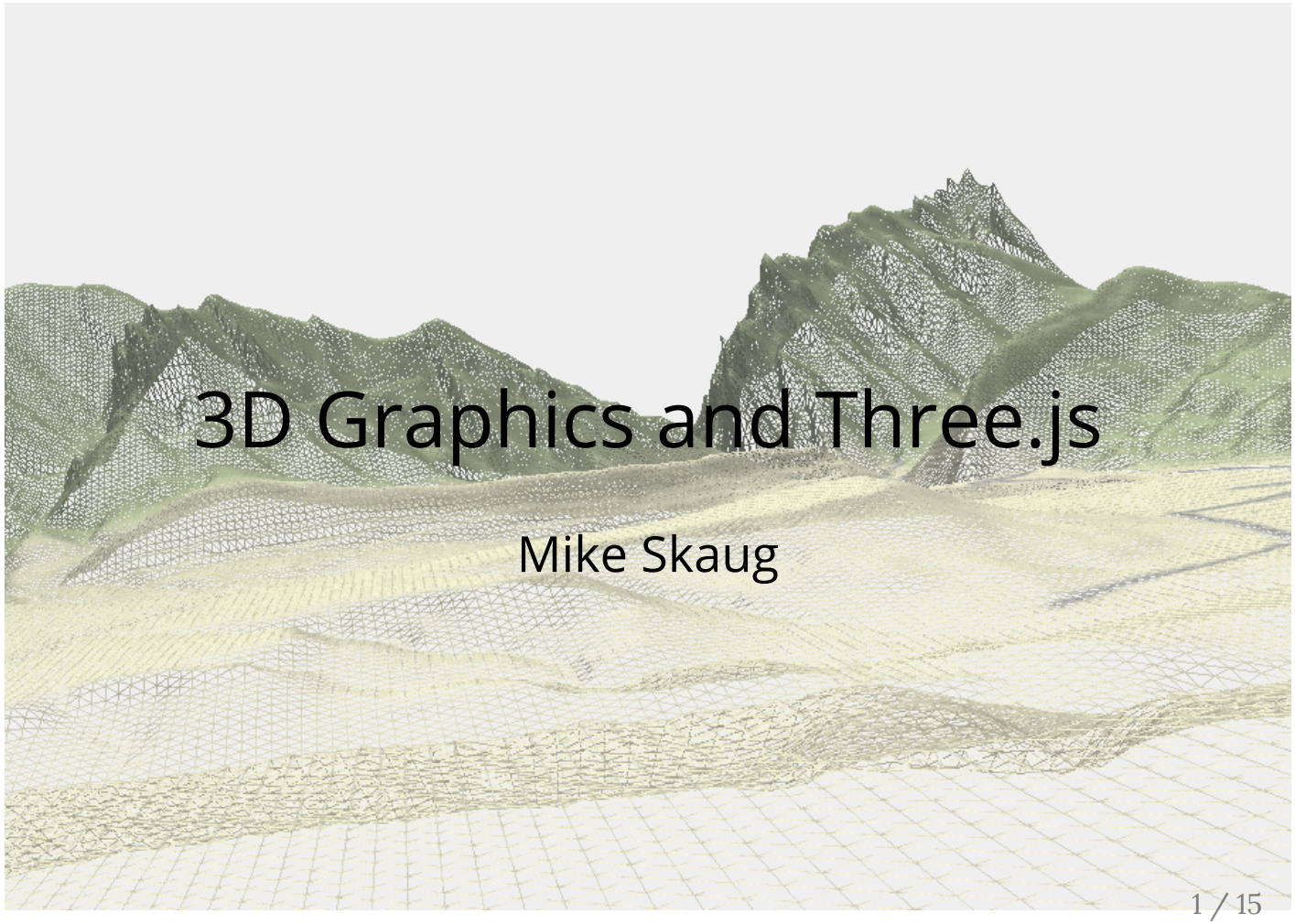
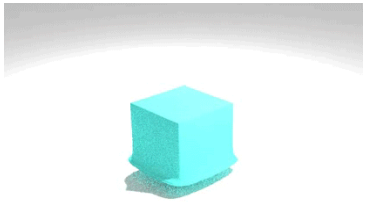
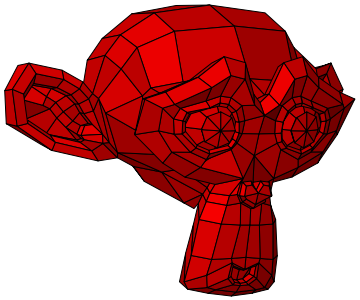

3D Graphics and Three.js
Mike Skaug
1 / 15

3D Graphics

Animation, games, modelling, ...

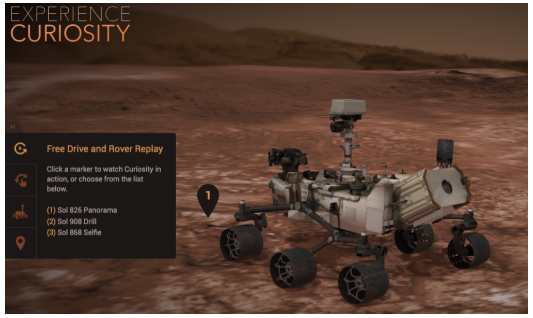
Sofware: Blender, Maya, AutoCAD, Cinema4D, ...

3D Graphics - in a browser

- Javascript API for 3D graphics introduced in 2011.
- Uses GPU to render to a `<canvas>`
- Relatively low level, shaders still written in GLSL

3D Graphics - in a browser

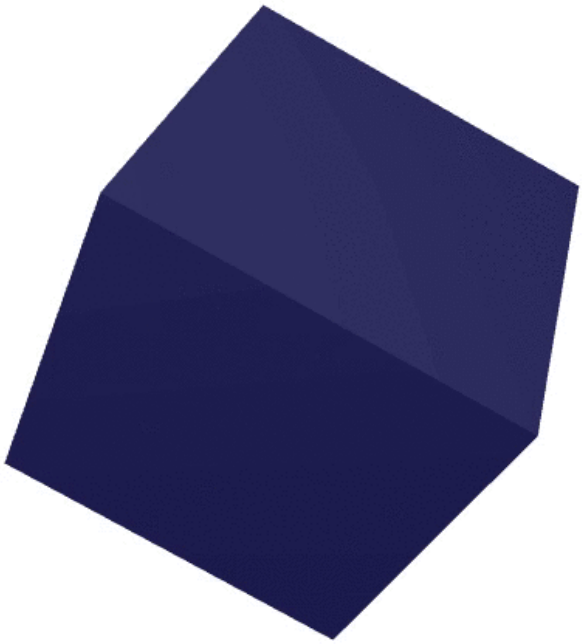
- Javascript API for 3D graphics introduced in 2011.
- Uses GPU to render to a `<canvas>`
- Relatively low level, shaders still written in GLSL



But you have to write code like this:

```
<script id="shader-vs" type="x-shader/x-vertex"></script>
<script id="shader-fs" type="x-shader/x-fragment"></script>

...

function initBuffers() {
        pyramidVertexPositionBuffer = gl.createBuffer();
        gl.bindBuffer(gl.ARRAY_BUFFER, pyramidVertexPositionBuffer);
...
mat4.identity(mvMatrix);
mat4.translate(mvMatrix, [-1.5, 0.0, -8.0]);
mvPushMatrix();
mat4.rotate(mvMatrix, degToRad(rPyramid), [0, 1, 0]);
...
```

To get something like this:



Higher Level Frameworks

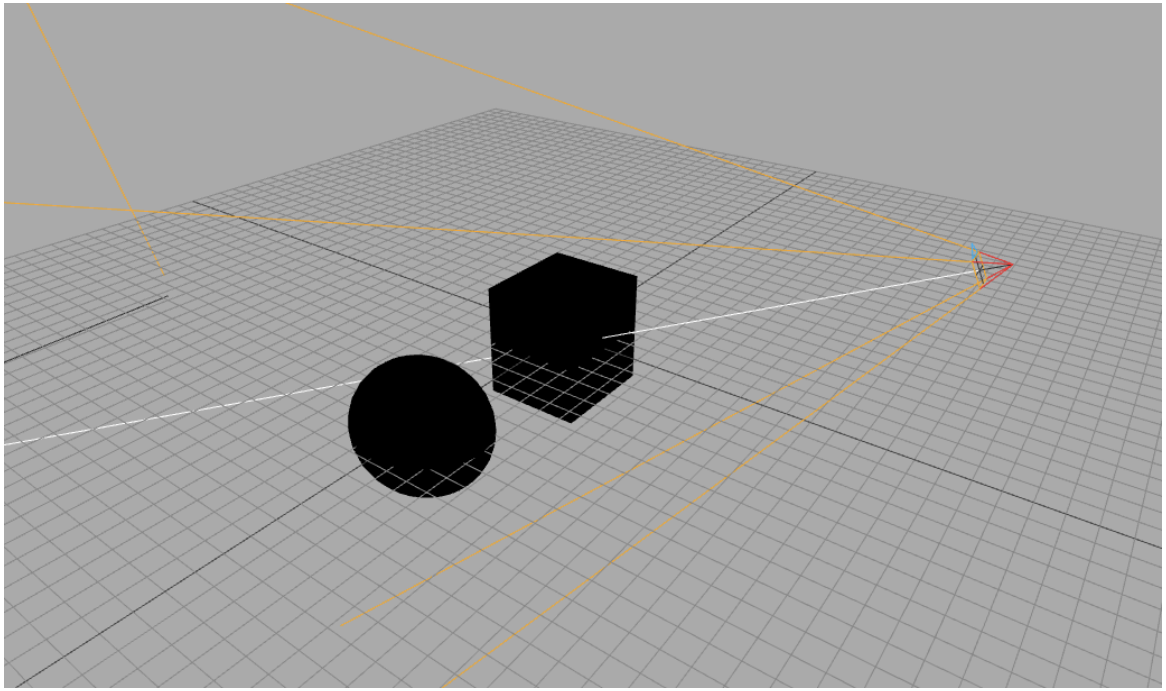
three.js r84

- lots of development starting in 2009
- uses WebGl as well as other renderers

- focused on game development
- a little yonger, released 2013

Lights, Camera, Objects...

- 3D modelling borrows from cinematography



Simple Scene

Create a scene and a camera

```
let THREE = require('three');

let scene = new THREE.Scene();
scene.background = new THREE.Color('rgb(80%, 80%, 80%)');
let camera = new THREE.PerspectiveCamera(75, 1, 0.1, 1000); # or OrthographicCamer
camera.position.z = 30;
```

Create renderer and add it to the DOM

```
let renderer = new THREE.WebGLRenderer();
renderer.setSize(500, 500);
document.getElementById('3d-scene').appendChild(renderer.domElement);
```

Create an object and add it to the scene

```
let geometry = new THREE.TorusKnotGeometry(10, 3, 100, 16);
let material = new THREE.MeshPhongMaterial({color: 0xFF7F00});
let torusKnot = new THREE.Mesh(geometry, material);
scene.add(torusKnot);
```

Simple Scene

Add lights

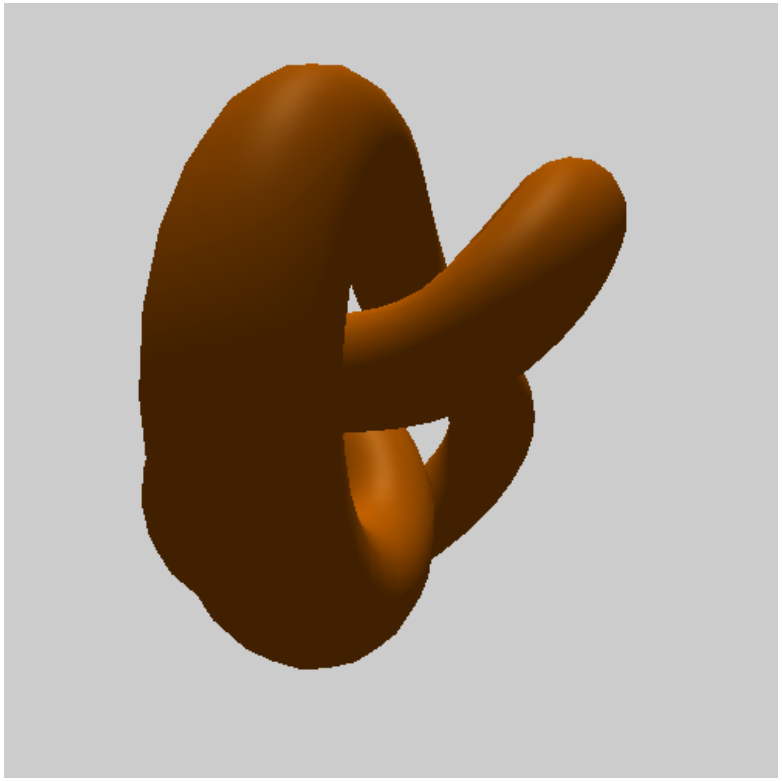
```
let light = new THREE.AmbientLight(0x404040);
scene.add(light);

let directionalLight = new THREE.DirectionalLight(0xffffff, 0.5);
scene.add(directionalLight);
```

Render loop

```
let render = () => {
  requestAnimationFrame(render);

  torusKnot.rotation.x += 0.01;
  torusKnot.rotation.y += 0.01;

  renderer.render(scene, camera);
};

render();
```

Lots of other lights, geometries, materials, etc

The Result



Pain Points

Documention!

- API has changed significantly in recent years - many resources out of date
- Documentation is sparse - nuts and bolts, but no concepts
- The mystery `three/examples/js` directory

Usage

- Limited set of geometry primitives
- No physics
- Difficult to integrate with other frameworks like React
- Do you actually need 3D?

ThreeJS Alternatives

WhitestormJS

- Really new (2016) wrapper around ThreeJS
- adds physics, ES6 style, modern tools

Unity

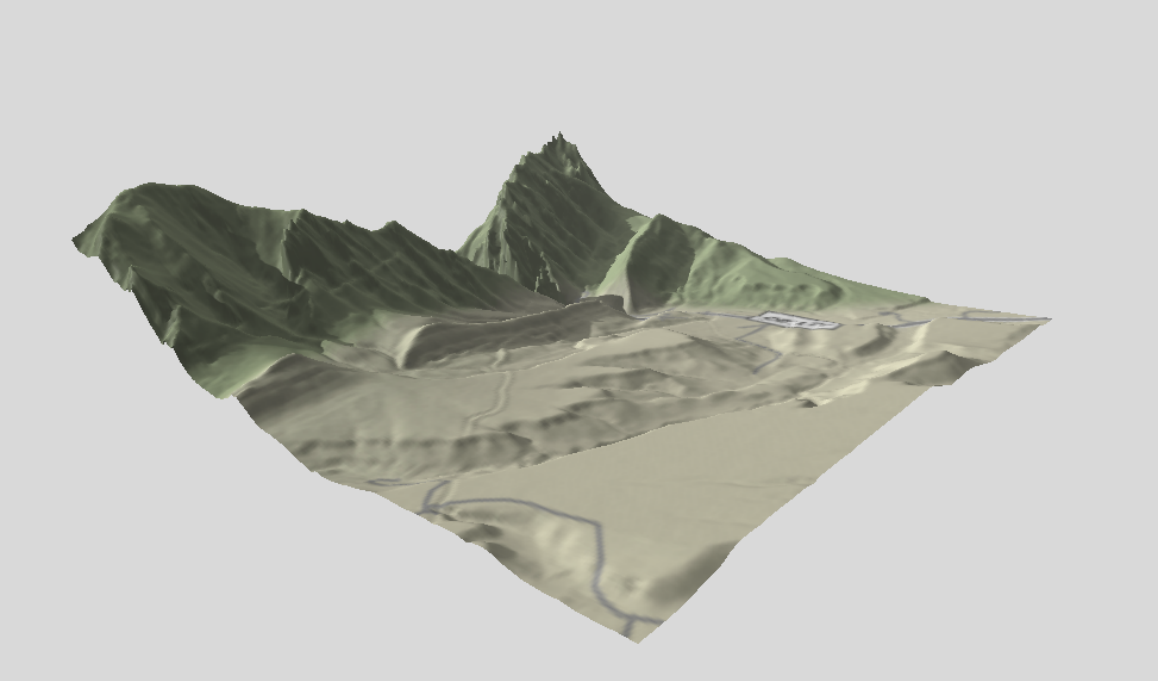
- Big platform
- desktop editor, deploy to mobile, web, desktop
- Targeted towards game development and VR

Cesium

- "An open-source JavaScript library for world-class 3D globes and maps"

Use Case

Simulating sunlight on the earth's surface



Thanks!

@mike_skaug

mike.skaug@gmail.com