Creating 3D apps & games using Babylon.JS

Microsoft

David Catuhe

Windows Clients Evangelist Lead

http://aka.ms/david

@deltakosh

David Rousset

Windows Clients Evangelist

http://aka.ms/davrous

@davrous

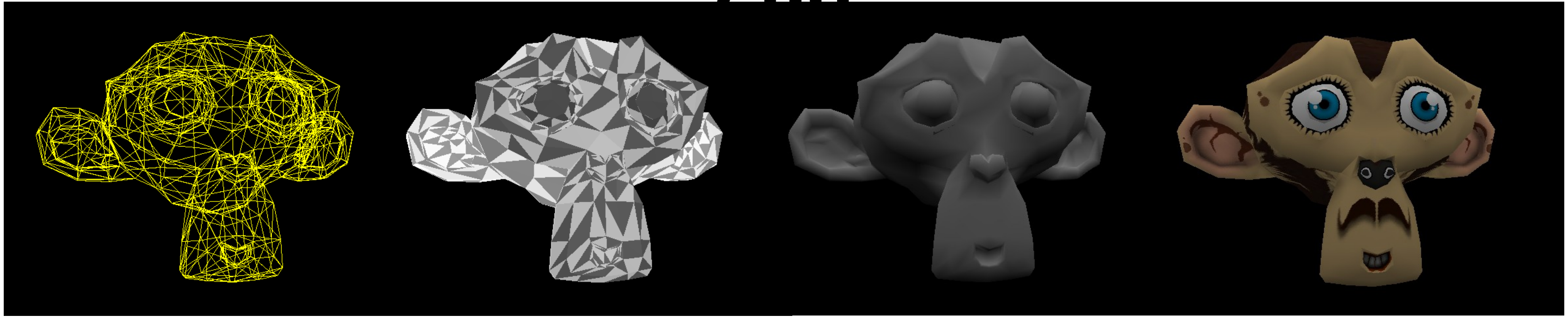
Agenda

- **Why building a WebGL 3D engine ?**
  - The old school way: Using the 2D canvas
  - The rise of GPUs
  - Using **WebGL** directly
- **Using Babylon.js to create 3D apps and games**
  - How to use **Babylon.js?**
  - Advanced features
- **What we’ve learned…**
  - Tracking and reducing the pressure on garbage collector
  - Performance first
  - Handling touch devices

Why building a **WebGL** 3D engine ?

The oldschool way: using **2D canvas**

Build a 3D "**Software**" engine that only uses the **CPU**



Soft Engine

DEMONSTRATION

The **rise** of GPUs

Hardware accelerated rendering:
2D Canvas, CSS3 animations

H264 & **JPG** hardware **decoding**

Accelerated 3D with **WebGL**

Using **WebGL** directly

Requires a **compatible** browser:

A new **context** for the canvas:

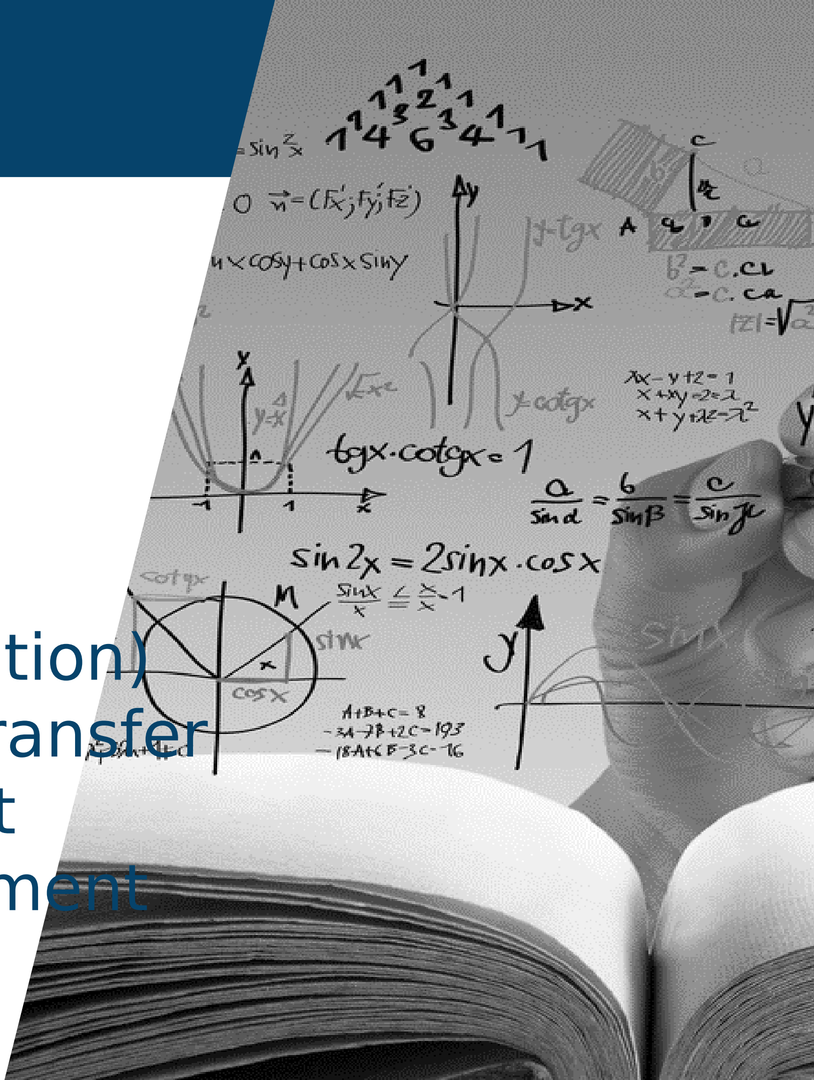
```
canvas.getContext("webgl", { antialias: true}) ||
canvas.getContext("experimental-webgl", { antialias: true});
```

Using **WebGL** directly

WebGL is a **low** level API

Need to handle **everything**
except the *rendering:*

- Shaders code (loading, compilation)
- Geometry creation, topology, transfer
- Shaders variables management
- Texture and resources management
- Render loop



WebGL 101

DEMONSTRATION

Using **Babylon.js** to create 3D apps & games

How to use **Babylon.js** ?

Open source project (Available on Github)

http://www.babylonjs.com
https://github.com/babylonjs/babylon.js

How to use it? **Include** one file and you’re ready to go!

```
<script
src="babylon.js"></script>
```

To start Babylon.js, you’ve just need to create an **engine** object:

```
var engine = new BABYLON.Engine(canvas,
true);
```

How to use **Babylon.js** ?

Babylon.js is a **scene graph**: All complex features are abstracted for **YOU**!

```
var scene = new BABYLON.Scene(engine);

var camera = new BABYLON.FreeCamera("Camera", new BABYLON.Vector3(0, 0, -10), 
scene);
var light0 = new BABYLON.PointLight("Omni0", new BABYLON.Vector3(0, 100, 100), 
scene);
var sphere = BABYLON.Mesh.createSphere("Sphere", 16, 3, scene);
```

Handling **rendering** can be done in one line:

```
        engine.runRenderLoop(function() 
        { scene.render(); });
```

Hello World with **Babylon.js**

DEMONSTRATION

Advanced features

Blender exporter
Design & render

Offline support
IndexedDB

Complete **collisions** engine

Network optimizations
Incremental loading

Unleash babylon.js

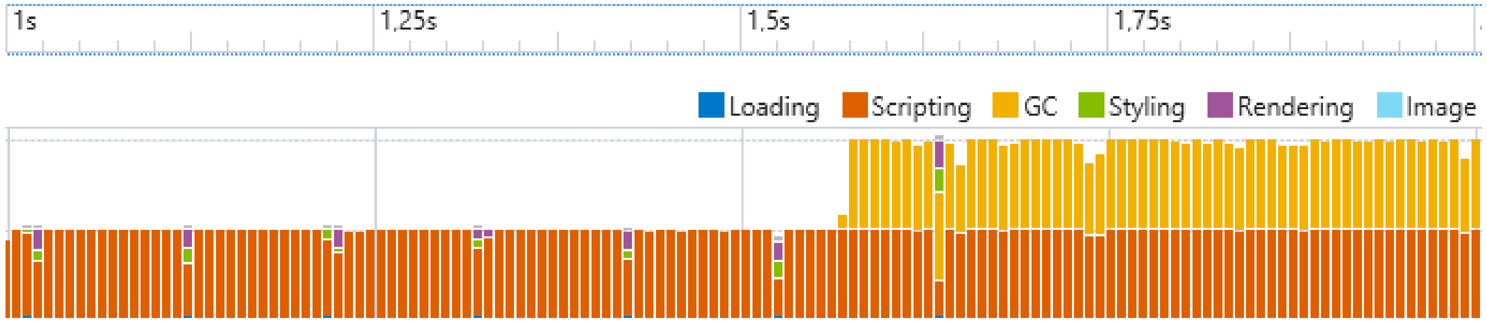
DEMONSTRATION

What we’ve **learned** ?

Tracking & reducing the **pressure** on GC

A **3D engine** is a place where matrices, vectors and quaternions live.
And there may be tons of them!

Pressure is huge on the **garbage collector**



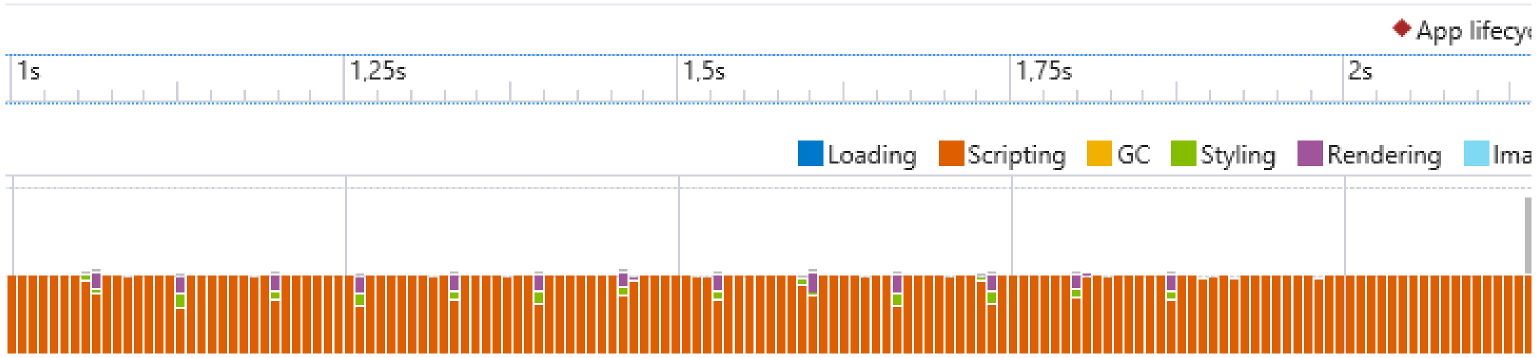
Tracking & reducing the **pressure** on GC

Maximum reuse of mathematical entities

- Pre-instantiate
- Stock variables

GC friendly arrays (able to reset size at no cost)

When the scene is up and running, aiming at **no allocation** at all



Using **F12** to reduce memory **pressure**

DEMONSTRATION

Performance first

Efficient shaders
Do only what is REALLY required

Complete cache system
Update WebGL only when required

Scene partitioning
Frustum / submeshes / octrees

Handling **touch** devices

Hand.js and the **TouchCamera**

DEMONSTRATION

Questions ?

@deltakosh / @davrous