



DevCon5

**HTML5 & Mobile App
Developer Conference**

December 10-11, 2013
Hilton Los Angeles/Universal City,
California

www.devconfive.com

@DevConFive
#DevCon5



Achieving seamless 3D Content Integration within web apps

Ryan Paul - MontageJS - Developer Relations
ryan@montagejs.org

Fabrice Robinet - MontageJS & COLLADA Chair
fabricerobinet@me.com

KHRONOS
GROUP



@DevConFive
#DevCon5



Menu

- Context
- Importing : From inter-exchange formats to WebGL
- Manipulating: from WebGL to CSS
- Using WebGL with MontageJS
- Demo

@DevConFive
#DevCon5



Context

On the Web, **content is king**

- Images have **JPEG, PNG...**
- Audio has **MP3...**
- 3D ???

@DevConFive
#DevCon5



Importing

From inter-exchange formats to WebGL

- targets ?
- expectations ?
- what's common today ?

@DevConFive
#DevCon5



Importing targets

- Phones & Tablets
- Desktop
- ... on the Web !.

@DevConFive
#DevCon5



Importing expectations

- ease of implementation
 - should be < 5 lines of client code to load a scene - ideally 0 :)
 - fully set up. Including all lights, cameras, animations...
- efficiency (bandwidth, performance...)
- should map well to underlying graphics API (e.g WebGL)
 - Forward directly datas to WebGL (indices, vertices, shaders and even types and enum values)
- flexibility - negotiate content:
 - Rest3D : <https://github.com/amd/rest3d>

@DevConFive
#DevCon5



Importing

what's common today ?

- many lines of code to configure a scene:
 - http://threejs.org/examples/#webgl_materials_cars
 - configure lights
 - configure materials...
 - 3D artists should be able to fully setup scenes - not only programmers.
 - COLLADA contains all this info, but is too heavy for our targets...

@DevConFive
#DevCon5



Importing

There is a missing a piece to bridge the gap between
inter-exchange formats and WebGL

.... a delivery format.

@DevConFive
#DevCon5



Importing



@DevConFive
#DevCon5



Importing

glTF: what's that ?

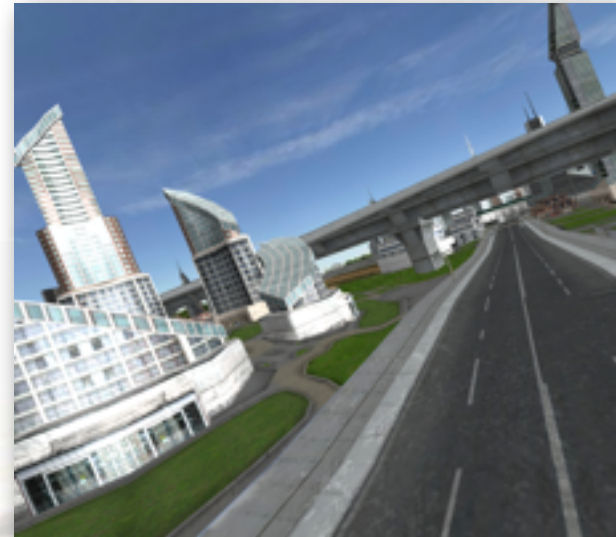
- **g**raphic **l**ibrary **T**ransmission **F**ormat
- runtime asset format for GL based APIs
- COLLADA WG initiative, not yet a ratified specification
- compact representation of a scene:
 - JSON for node hierarchy, shaders
 - binary for mesh and animation data.
- specification and implementation (converter) in the open:
<https://github.com/khronosgroup/glTF>

@DevConFive
#DevCon5



Importing glTF: **MontageJS** viewer

- demo: <https://github.com/fabrobinet/glTF-webgl-viewer.git>



Models from www.3drt.com

Importing

glTF: design principles

- ease of implementation (JSON + typed arrays)
- efficient (performance, bandwidth)
- offload processing from runtime
- runtime neutral - can be used by any framework or app
- built on top of GL Based APIs and Typed Arrays

@DevConFive
#DevCon5



Importing

glTF: understanding the content pipeline



@DevConFive
#DevCon5



Importing

rendering steps:
1. set GL states



gl.enable
gl.disable
gl.depthMask
gl.stencilFunc
...

gl.useProgram

gl.uniformXXX

gl.drawElements

Overview

technique

pass

states

geometry

primitive



```
"woodTechnique": {  
  "passes": {  
    "defaultPass": {  
      "states": {  
        "enable": [],  
        "disable": [],  
        "depthMask": 1,  
        "stencilFunc": [..]  
      }  
    }  
  }  
}
```

@DevConFive
#DevCon5



Importing

rendering steps:
2. use program



gl.enable
gl.disable
gl.depthMask
gl.stencilFunc
...

gl.useProgram

gl.uniformXXX

gl.drawElements

Overview

technique

pass

instanceProgram

geometry

primitive



```
"woodTechnique": {  
  "passes": {  
    "defaultPass": {  
      "instanceProgram": {  
        "program": "woodProgram",  
      }  
    }  
  },  
  "programs" : {  
    "woodProgram" : {  
      "fragmentShader" : "woodFS",  
      "vertexShader" : "woodVS"  
    }  
  }  
}
```

@DevConFive
#DevCon5



Importing

rendering steps:

3. set uniforms and attributes



```
gl.enable  
gl.disable  
gl.depthMask  
gl.stencilFunc  
...
```

```
gl.useProgram  
gl.uniformXXX
```

```
gl.drawElements
```

Overview

```
technique
```

```
pass
```

```
instanceProgram
```

```
geometry
```

```
primitive
```



```
"woodTechnique": {  
  parameters: {  
    "shininess": { " value ": 100, ... }  
  },  
  "passes": {  
    "defaultPass": {  
      "instanceProgram": {  
        "program": "woodProgram",  
        "uniforms": {  
          "u_shininess": "shininess",  
        }  
      }  
    }  
  }  
}
```

@DevConFive
#DevCon5



Importing

rendering steps:
4. draw geometry



gl.enable
gl.disable
gl.depthMask
gl.stencilFunc
...

gl.useProgram
gl.uniformXXX

gl.drawElement

Overview

technique

pass

instanceProgram

geometry

primitive



```
"meshes": {  
  "table_0": {  
    "name": "kitchenTable",  
    "primitives": [  
      {  
        "attributes": {  
          "NORMAL": "attribute_25",  
          "POSITION": "attribute_23",  
        },  
        "indices": "indices_21",  
        "material": "woodMaterial",  
        "primitive": 4  
      }  
    ]  
  }  
}
```

@DevConFive
#DevCon5



Importing

glTF: compression

- compression is an extension in glTF
- glTF & Open3DGC Integration
 - geometry: POSITION, NORMAL, multiple sets...
 - animations: TRS (Translation, Rotation, Scale)
 - skinning: weights and bone indices
- Open3DGC: Efficient Implementation of MPEG 3D Graphics Codecs
 - Encoders in C++ for fast server side compression
 - Decoders in javascript and C++ to target mobile and web applications

Importing glTF: compression

Results for “virtual city” scene

@DevConFive
#DevCon5



Importing glTF: compression

Results for “virtual city” scene

	geometry	animation	total
COLLADA			raw: 5469 kb gzip: 1612 kb
glTF	680 kb	1297 kb	raw: 1977 kb gzip: 878 kb
glTF Open3DGC- binary	198 kb	297 kb	raw: 496 kb gzip: 429 kb
glTF Open3DGC- ascii	355 kb	533 kb	raw: 888 kb gzip: 390 kb

@DevConFive
#DevCon5



Importing what's next ?

We want more control !

Displaying 3D Content is nice, but we'd like to change how it looks and have it react to user interactions...

@DevConFive
#DevCon5



Manipulating

From WebGL to CSS

- when possible, rely on CSS instead of adding APIs.
on-going MontageJS effort to let you apply CSS Styles on your 3D Content rendered by WebGL.

But still...

- keep the DOM clean of 3D concepts that aren't core HTML concepts
- 3D elements stored/described in glTF

@DevConFive
#DevCon5



Manipulating

A simple example: set a color

- APIs vs Declarations
- WebGL

```
var location = gl.getUniformLocation(program, "backgroundColor");  
gl.uniform3f(location, 1, 1, 1);
```

- CSS

```
.whiteBackground { background-color: #FFFFFF }  
<div class = "whiteBackground">
```

@DevConFive
#DevCon5



Using WebGL in a MontageJS Application

MontageJS

- Modern MVC framework for building SPAs
- Component architecture
- Binding system
- Managed draw cycle

Display a 3D scene

```
"scene": {  
  "prototype": "mjs-volume/runtime/scene",  
  "properties": {  
    "path": "models/model.json"  
  }  
},  
"view": {  
  "prototype": "mjs-volume/ui/view.reel",  
  "properties": {  
    "element": { "#": "view" },  
    "scene": { "@": "scene" }  
  }  
}
```

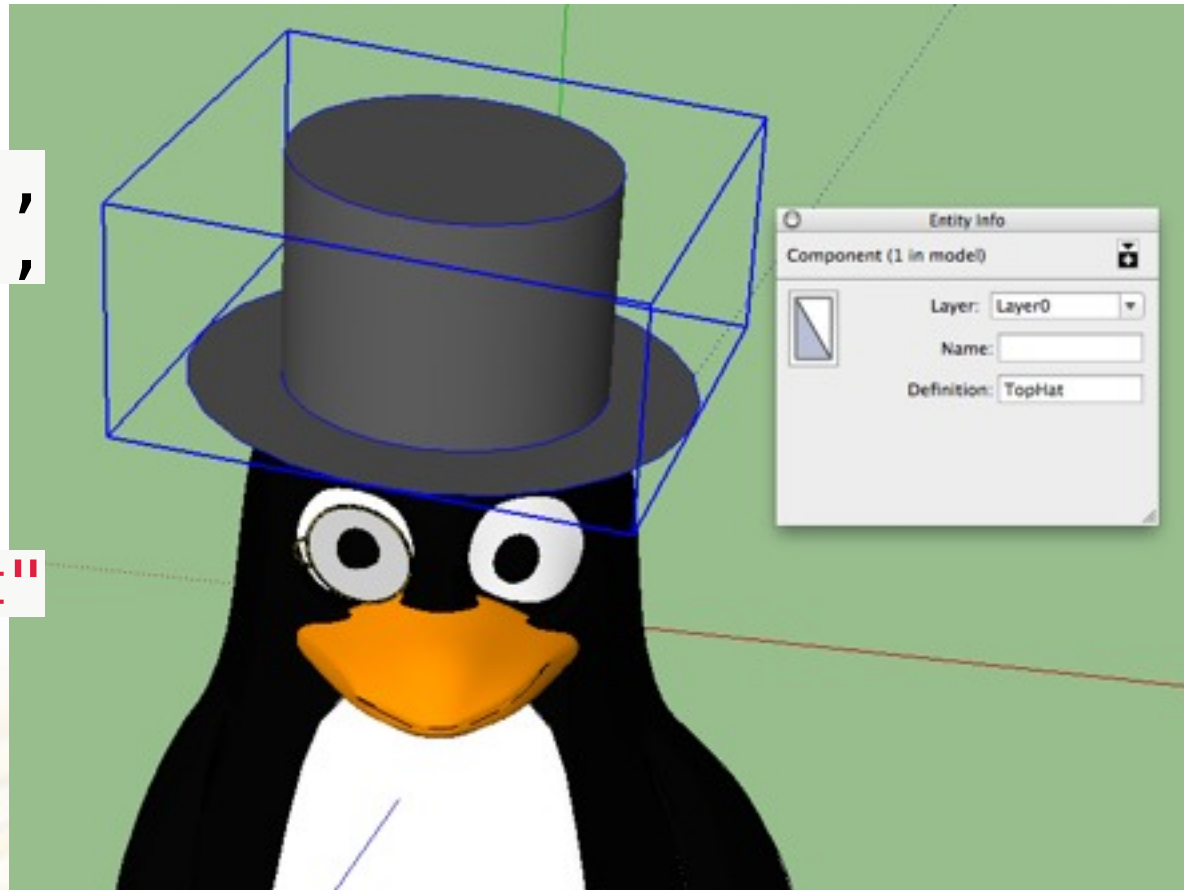
```
<div data-montage-id="view"></div>
```

Expose 3D node to binding system

```
"node": {  
  "prototype": "mjs-volume/runtime/node",  
  "properties": {  
    "id": "ID511",  
    "scene": { "@": "scene" }  
  }  
}
```

Expose 3D node to binding system

```
"TopHat" : {  
  "children" : [ ],  
  "matrix" : [ ... ],  
  "meshes" : [  
    "ID490",  
    "ID500"  
  ],  
  "name" : "TopHat"  
},
```



Apply CSS to a node

```
"hat": {  
  "prototype": "mjs-volume/runtime/node",  
  "properties": {  
    "id": "TopHat",  
    "scene": { "@": "scene" }  
  },  
  "bindings": {  
    "classList.has('.grow: hover')": {  
      "<": "true"  
    }  
  }  
}
```

Apply CSS to a node *:hover* and *scale3d*

```
.grow:hover {  
  -webkit-transform: scale3d(1, 1, 1.5);  
  -webkit-transform-origin: 0% 50%;  
  transition: -webkit-transform 3s;  
}
```

Apply CSS to a node

rotateY and separate animation

```
.open {  
  -webkit-transform: rotateY(-90deg);  
}
```

```
.animate {  
  -webkit-transform-origin: 0% 50%;  
  transition-property: -webkit-transform;  
  transition-duration: 5s;  
}
```

transform-origin

```
.animate {  
  -webkit-transform-origin: 0% 50%;  
  transition-property: -webkit-transform;  
  transition-duration: 5s;  
}
```



Bind to Checkbox

```
"monocle": {  
  "prototype": "mjs-volume/runtime/node",  
  "properties": {  
    "id": "Monocle",  
    "scene": { "@": "scene" }  
  },  
  "bindings": {  
    "classList.has('.hide')": {  
      "<-": "!@checkbox.checked"  
    }  
  }  
}
```

Bind to Checkbox

```
<input type="checkbox" data-montage-id="checkbox">
```

Fancy

```
.hide {  
  visibility: hidden;  
}
```

Handle click events

```
"hat": {  
  "prototype": "mjs-volume/runtime/node",  
  "properties": {  
    "id": "TopHat",  
    "scene": { "@": "scene" }  
  }  
  "listeners": [  
    { "type": "action", "listener": {  
      "@": "owner" } }  
  ]  
},
```

Handle click events

```
var Component = require("montage/ui/  
component").Component;  
  
exports.Penguin = Component.specialize({  
  handleHatAction: {  
    value: function(event) {  
      alert("Do you like my fancy hat?");  
    }  
  }  
});
```

Control Camera

```
"view": {  
  "prototype": "mjs-volume/ui/view.reel",  
  "properties": {  
    "allowViewPointControl": false,  
    "element": { "#": "viewer" },  
    "scene": { "@": "scene" },  
    "viewPoint": { "@": "camera1" }  
  },  
  "bindings": {  
    "viewPoint": {  
      "<-": "@cameras.value.cam"  
    }  
  }  
},  
,
```

Control Camera

```
"camera1": {  
  "prototype": "mjs-volume/runtime/node",  
  "properties": {  
    "id": "node-Camera02",  
    "scene": { "@": "scene" }  
  }  
},
```

Control Camera

```
"cameras": {  
  "prototype": "digit/ui/select.reel",  
  "properties": {  
    "element": { "#": "cameras" },  
    "content": [  
      { "label": "Camera 1", "cam": { "@": "cam1" } },  
      { "label": "Camera 2", "cam": { "@": "cam2" } },  
      { "label": "Camera 3", "cam": { "@": "cam3" } },  
      { "label": "Camera 4", "cam": { "@": "cam4" } }  
    ],  
  },  
}
```

Demo

Thank You!
<http://montagejs.org>
[@montagejs](#)