WebGL

Now Is The Time To Start Paying Attention

FITC Toronto 2012

Dustin Malik, Erik Oros

http://developer.blackberry.com/

Erik Oros

Background

- Application Development Consultant with Developer Relations team.
- Initial focus was with BlackBerry Java.
- Transitioned to HTML5 and loving it.

On The Web

- @BlackBerryDev ^EO
- @WaterlooErik
- oros (BlackBerry Support Forms)

Dustin Malik

- Work for RIM in Developer Relations
- 7 Years of Flash Experience at RIM (Primarily Flash IDE)
- 1 Month with Developer Relations
- Twitter - @dustinmalik

WebGL

What Is It? What Can It Do For You?

What Is It?

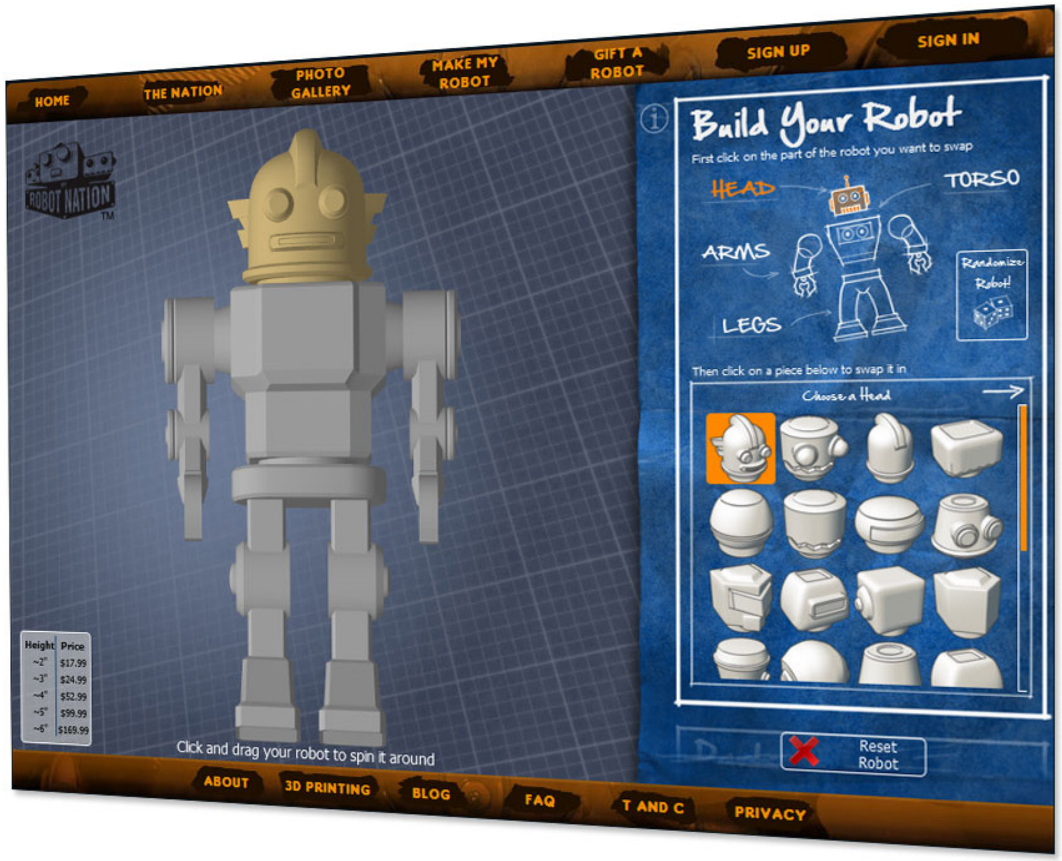
WebGL is OpenGL ES 2.0 in your Web Browser

- Khronos open standard
- Hardware accelerated graphics for the web
- It’s fully integrated. It is NOT a plugin.
- Version 1.0 released March 2011

Where Can I Use It?

It runs in any standards compliant browser.

- Chrome
- Firefox
- Safari
- Opera



First click on the part of the robot you want to swap

Then click on a piece below to swap it in

Height	Price
~2"	$17.99
~3"	$24.99
~4"	$52.99
~5"	$99.99
~6"	$169.99

Click and drag your robot to spin it around

What About Mobile Devices?

- Runs on BlackBerry PlayBook
- Will run on future BlackBerry 10 devices
- iOS and Android support is coming soon

Why Is It Cool?

- Rapid Prototyping
  - You can write it in JavaScript and view it live in a web browser vs writing in C++ and compiling.
- Cross platform
- Many JavaScript developers out there

Using WebGL

The Basic Steps

You can use WebGL by following this outline:

1. Create a Canvas
2. Get a WebGL context
3. Setup a GLSL program
4. Buffer Geometry
5. Setup Shaders
6. Render

1. Create A Canvas

```
<canvas id="myCanvas" width="1024" height="512” ></canvas>
```

2. Get A WebGL Context

```
var canvas = document.getElementById( “myCanvas");
var gl = canvas.getContext("experimental-webgl");
```

3. Setup A GLSL Program

```
var vertexShader = createShaderFromScriptElement(gl, "2d-vertex-shader");

var fragmentShader = createShaderFromScriptElement(gl, "2d-fragment-
shader");

var program = createProgram(gl, [vertexShader, fragmentShader]);

gl.useProgram(program);

var positionLocation = gl.getAttribLocation(program, "a_position");
```

4. Buffer Geometry

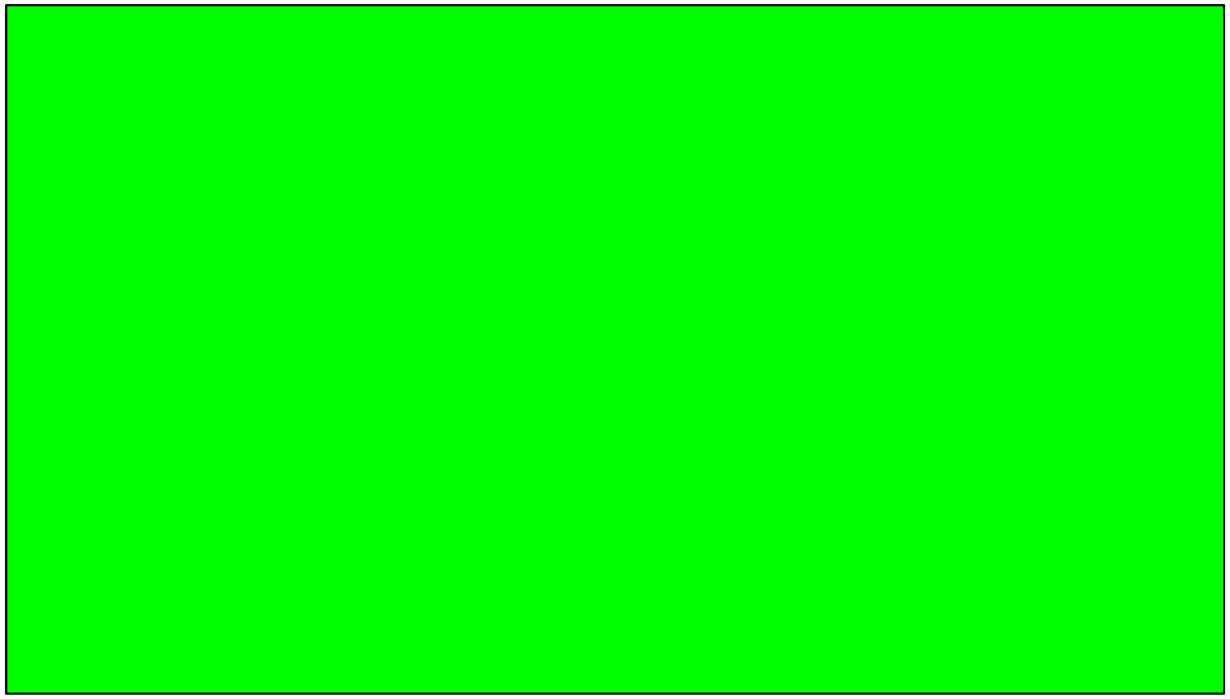
```
var buffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, buffer);
gl.bufferData(
    gl.ARRAY_BUFFER,
    new Float32Array([
        -1.0, -1.0,
         1.0, -1.0,
        -1.0,  1.0,
        -1.0,  1.0,
         1.0, -1.0,
         1.0,  1.0]),
    gl.STATIC_DRAW);
gl.enableVertexAttribArray(positionLocation);
gl.vertexAttribPointer(positionLocation, 2, gl.FLOAT, false, 0, 0);
```

5. Setup Shaders

```
<script id="2d-vertex-shader" type="x-shader/x-vertex">
attribute vec2 a_position;

void main() {
  gl_Position = vec4(a_position, 0, 1);
}
</script>

<script id="2d-fragment-shader" type="x-shader/x-fragment">
void main() {
  gl_FragColor = vec4(0,1,0,1);  // green
}
</script>
```

6. Render

```
gl.drawArrays(gl.TRIANGLES, 0, 6);
```



Using WebGL With Three.js

The Basic Steps

With Three.js we can minimize the complexity of WebGL.
The Three.js workflow is as follows:

1. Import Three.js
2. Setup scene
3. Setup camera
4. Add geometry with material
5. Render scene

1. Import Three.js

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

2. Setup Scene

```
scene = new THREE.Scene();
```

3. Setup Camera

```
scene = new THREE.Scene();

camera = new THREE.PerspectiveCamera( 75, window.innerWidth /
window.innerHeight, 1, 10000 );
camera.position.z = 1000;
scene.add( camera );
```

4. Add Geometry With Material

```
scene = new THREE.Scene();

camera = new THREE.PerspectiveCamera( 75, window.innerWidth /
window.innerHeight, 1, 10000 );
camera.position.z = 1000;
scene.add( camera );

geometry = new THREE.CubeGeometry( 1024, 600, 0 );
material = new THREE.MeshBasicMaterial( { color: 0x00FF00, wireframe: false }
);
mesh = new THREE.Mesh( geometry, material );
scene.add( mesh );
```

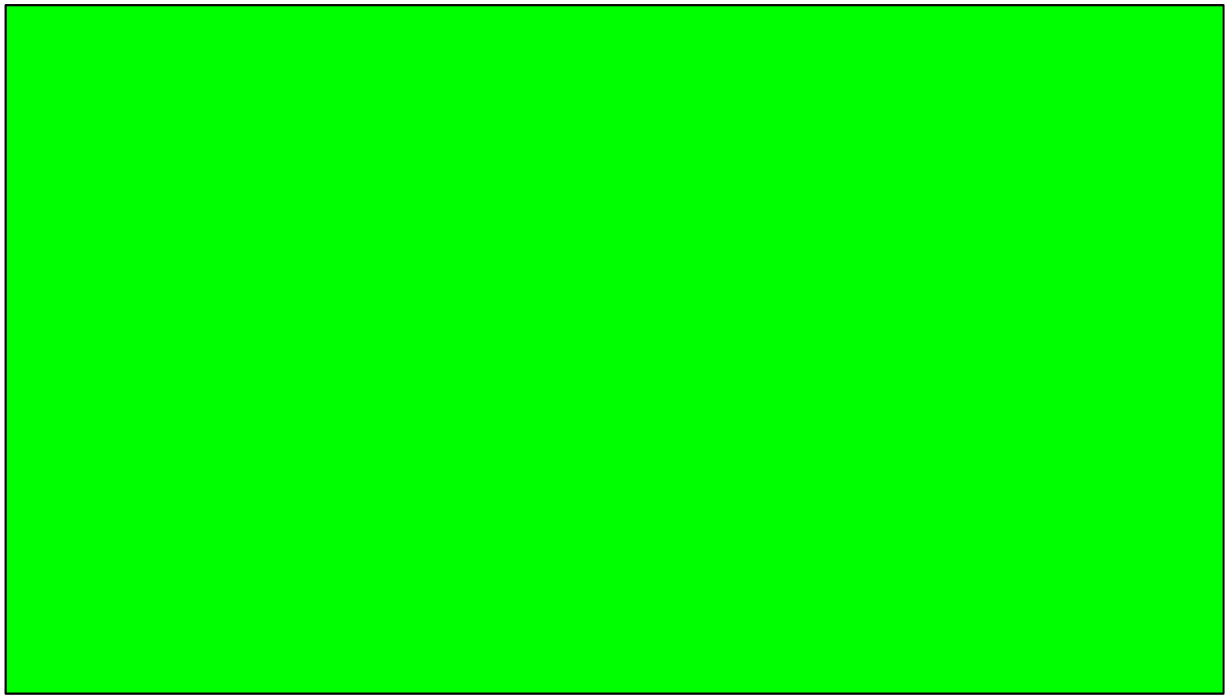
5. Render Scene

```
scene = new THREE.Scene();

camera = new THREE.PerspectiveCamera( 75, window.innerWidth /
window.innerHeight, 1, 10000 );
camera.position.z = 1000;
scene.add( camera );

geometry = new THREE.CubeGeometry( 1024, 600, 0 );
material = new THREE.MeshBasicMaterial( { color: 0x00FF00, wireframe: false }
);
mesh = new THREE.Mesh( geometry, material );
scene.add( mesh );

renderer = new THREE.CanvasRenderer();
renderer.setSize( window.innerWidth, window.innerHeight );
document.body.appendChild( renderer.domElement );
renderer.render( scene, camera );
```

Same Result, Fewer Lines Of Code

Raw WebGL = 47 lines of code

Three.js = 26 lines of code



J3D

J3D was created by Bartek Drozdz as an experiment vs a production environment.

- Still actively contributing in Github
- Has support to export Unity scenes
- github.com/drojdjou/J3D

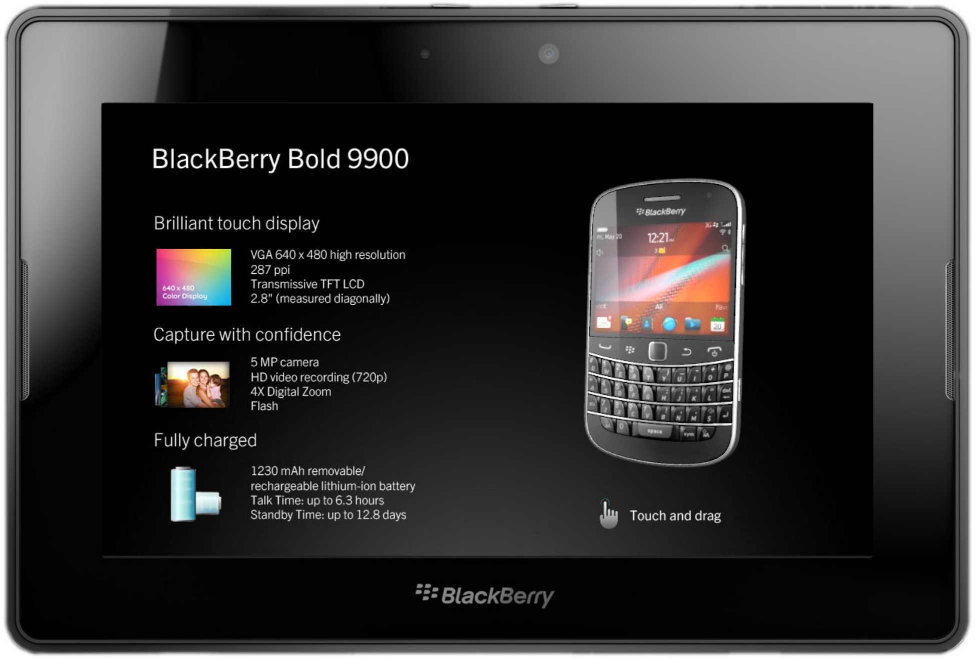
Demo

http://www.everyday3d.com/j3d/demo/008_Lightmap.html

Inka3D

Inka3D allows you to export Maya scenes into WebGL.

- Does all the heavy lifting
- Allows you to easily manipulate the models you export from Maya using JavaScript
- Not open source ☹
- Inka3d.com



BlackBerry Bold 9900

Brilliant touch display

VGA 640 x 480 high resolution
287 ppi
Transmissive TFT LCD
2.8" (measured diagonally)

Capture with confidence

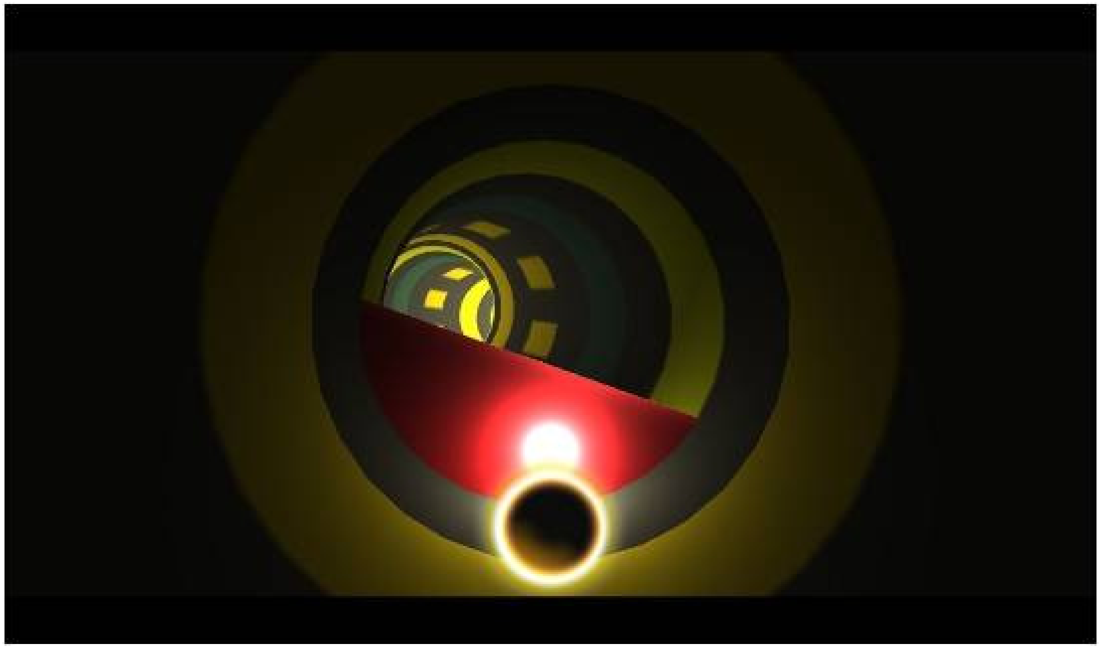
5 MP camera
HD video recording (720p)
4X Digital Zoom
Flash

Fully charged

1230 mAh removable/
rechargeable lithium-ion battery
Talk Time: up to 6.3 hours
Standby Time: up to 12.8 days

TunnelTilt

In The Beginning

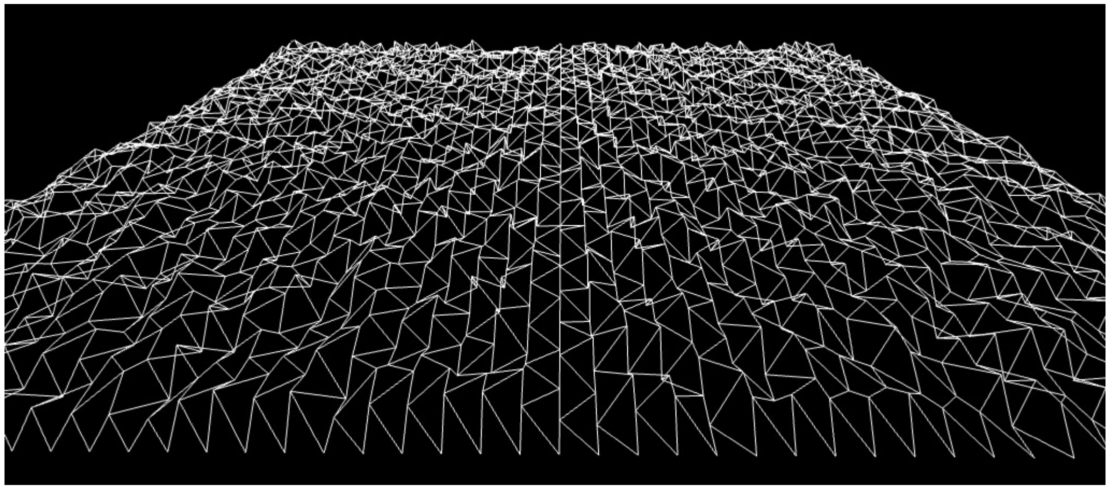


- First WebGL demo released for PlayBook OS 2.0
- Available on BlackBerry App World
- Open-sourced on www.github.com/blackberry
- Key Features: Accelerometer and Collision Detection
- Honourable mention: Balloon Gunner 3D

PeaksAndValleys

My WebGL Journey

Frameworks vs Pure WebGL

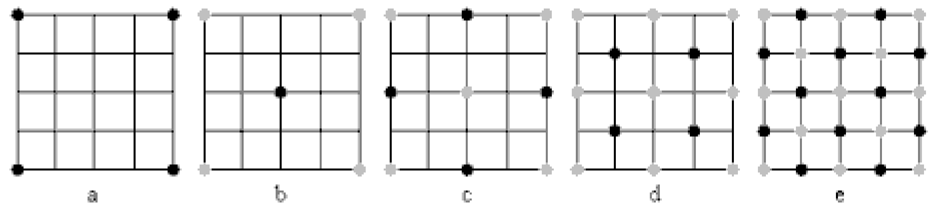


- three.js by mrdoob (Github)
- glMatrix by toji (Github)
- virtualjoystick.js by jeromeetienne (Github)
- bullet.js by adambom (Github)
- jiglibjs by jiglibjs.org

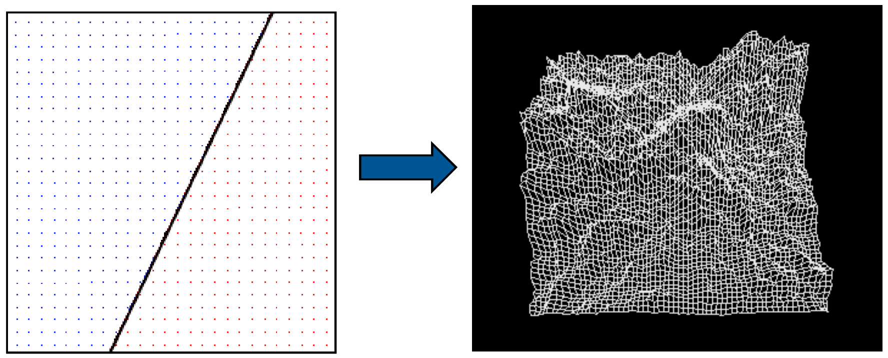
Frameworks will be revisited.

Generating The Landscape

Diamond-Square Fractal Algorithm[1]



Fault Algorithm[2]



Raised Zones

- Generate a square.
- Raise all included vertices.
- Repeat.
- Average with neighbours.

1 - http://www.gameprogrammer.com/fractal.html

2 - http://www.lighthouse3d.com/opengl/terrain/index.php3?fault

Generating The Skybox

- Two easy steps:
  - Paint the inside of a box
  - Place box on head
- Works in real life too!

- Many pre-generated textures
- Size limitations: 1024 x 1024

- The magic happens with:

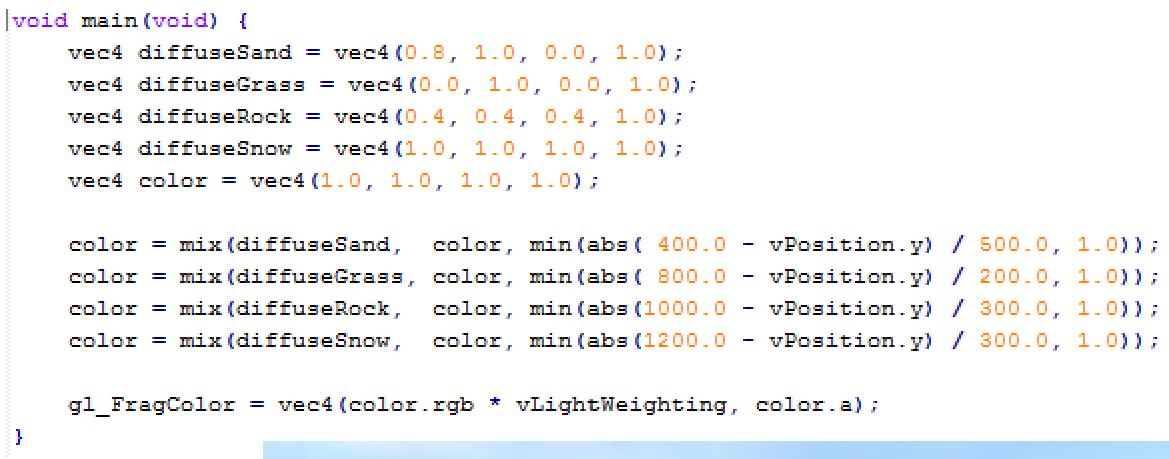
  gl.disable(gl.DEPTH_TEST);

  Don’t forget to turn this back on.

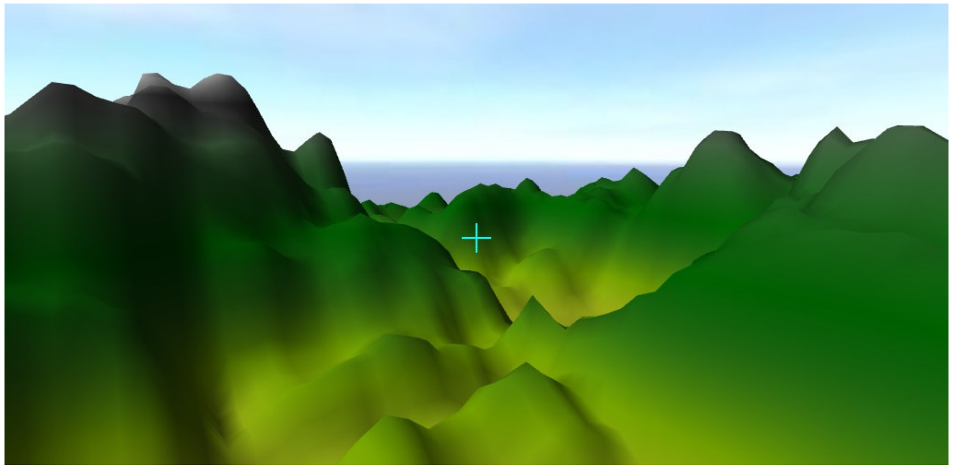
Image from: http://neverwateraweed.blogspot.ca/2011/08/hey-box-head.html

Blending Colours And Textures

- Fragment shader blends our colours and shadows.



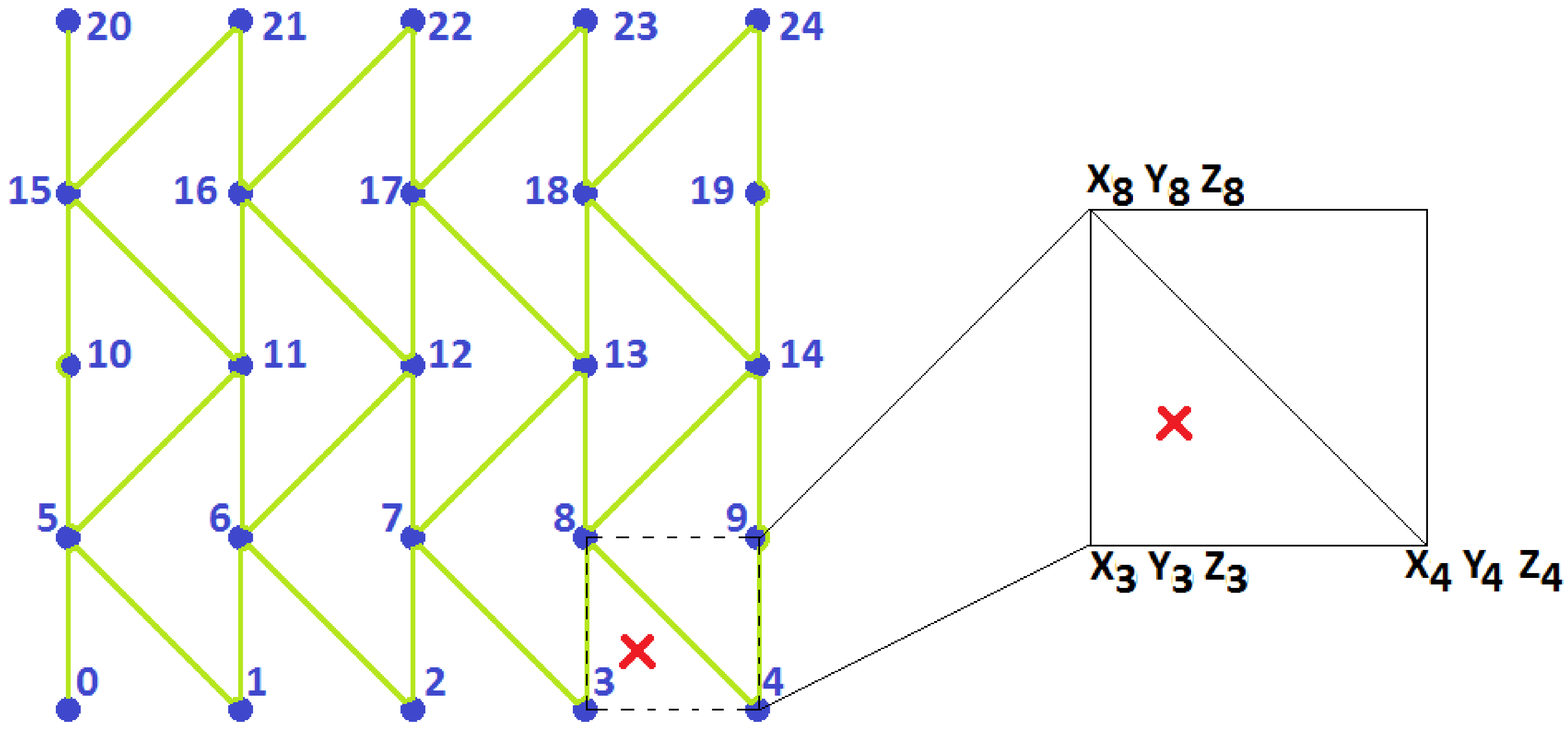
```
void main(void) {
    vec4 diffuseSand = vec4(0.8, 1.0, 0.0, 1.0);
    vec4 diffuseGrass = vec4(0.0, 1.0, 0.0, 1.0);
    vec4 diffuseRock = vec4(0.4, 0.4, 0.4, 1.0);
    vec4 diffuseSnow = vec4(1.0, 1.0, 1.0, 1.0);
    vec4 color = vec4(1.0, 1.0, 1.0, 1.0);

    color = mix(diffuseSand,  color, min(abs( 400.0 - vPosition.y) / 500.0, 1.0));
    color = mix(diffuseGrass, color, min(abs( 800.0 - vPosition.y) / 200.0, 1.0));
    color = mix(diffuseRock,  color, min(abs(1000.0 - vPosition.y) / 300.0, 1.0));
    color = mix(diffuseSnow,  color, min(abs(1200.0 - vPosition.y) / 300.0, 1.0));

    gl_FragColor = vec4(color.rgb * vLightWeighting, color.a);
}
```



Special thanks to : http://chandler.prallfamily.com/2011/06/blending-webgl-textures

Exploring The World

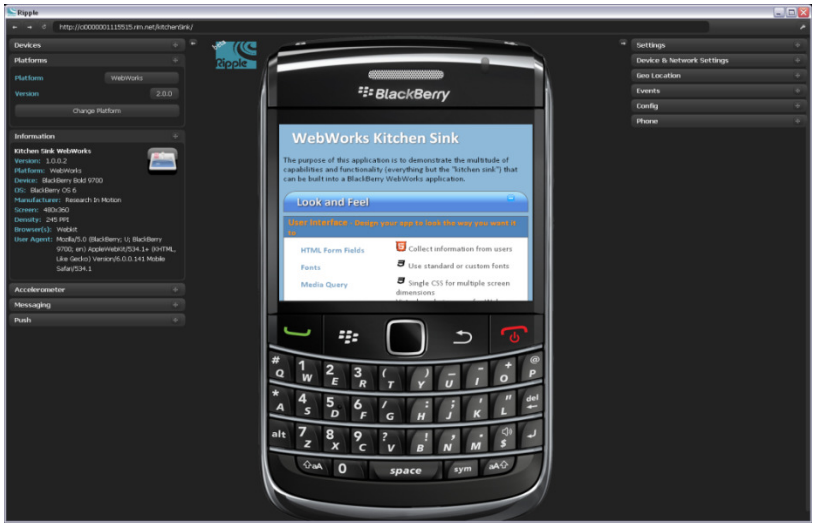
- Expanded on virtualjoystick.js for multi-touch
- Minimize triangles via gl.TRIANGLE_STRIP



Development Tools

http://developer.blackberry.com/html5

- BlackBerry WebWorks SDK
- Ripple Emulator
- Smartphone/Tablet Simulators
- Code Signing Keys



http://ci0000001115515.rim.net/kitchenSink/

Platform: WebWorks
Version: 2.0.0

Kitchen Sink WebWorks
Version: 1.0.0.2
Platform: WebWorks
Device: BlackBerry Bold 9700
OS: BlackBerry OS 6
Manufacturer: Research In Motion
Screen: 480x360
Density: 245 PPI
Browser(s): Webkit
User Agent: Mozilla/5.0 (BlackBerry; U; BlackBerry 9700; en) AppleWebKit/534.1+ (KHTML, Like Gecko) Version/6.0.0.141 Mobile Safari/534.1

WebWorks Kitchen Sink

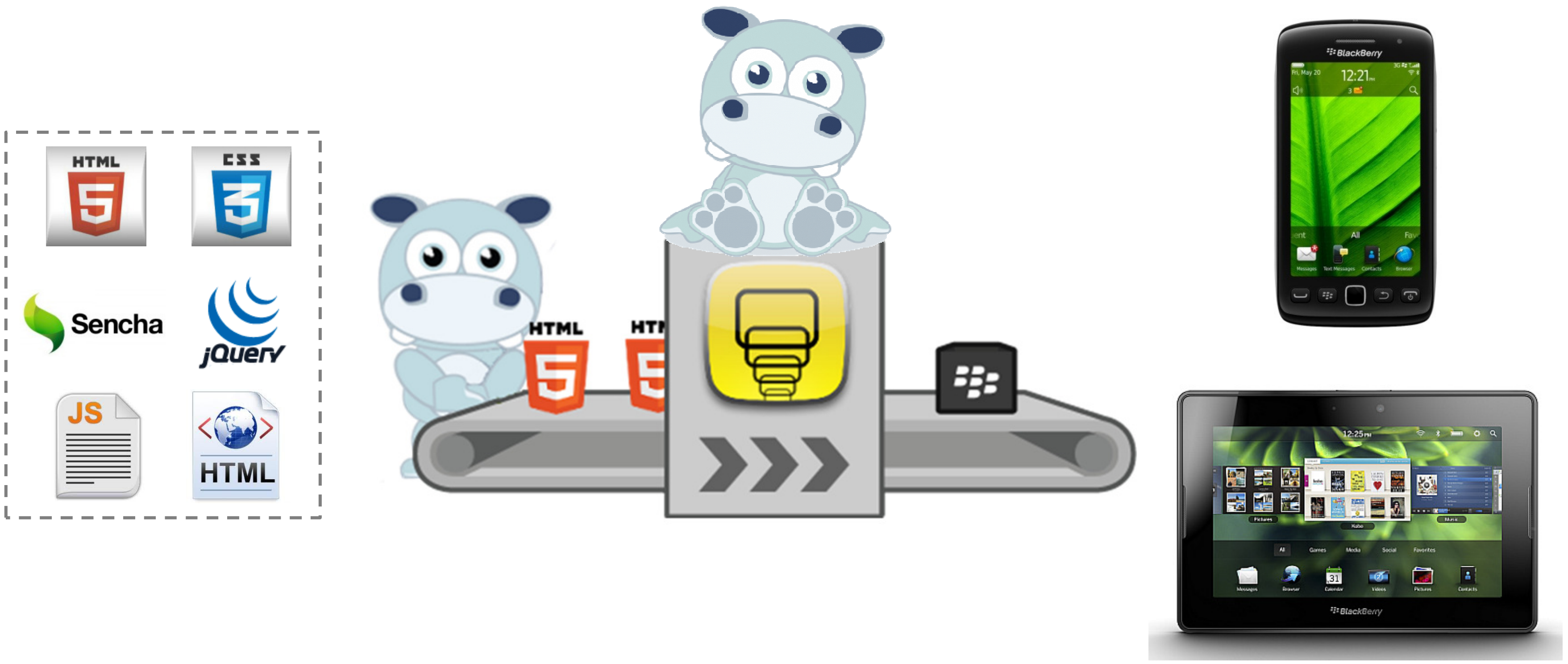
The purpose of this application is to demonstrate the multitude of capabilities and functionality (everything but the "kitchen sink") that can be built into a BlackBerry WebWorks application.

User Interface - Design your app to look the way you want it to

HTML Form Fields	Collect information from users
Fonts	Use standard or custom fonts
Media Query	Single CSS for multiple screen dimensions

BlackBerry

Putting It All Together



Web Assets
BlackBerry WebWorks
Applications

The Future of WebGL

The Spirit Of The Web

- Open, collaborative community.
- Use, enhance, contribute.

Comes naturally to seasoned web developers.

The Spirit Of The Web At RIM

- Many open source initiatives via Github
- Initial TunnelTilt sample open sourced
- Inka3D and PeaksAndValleys samples
  - Open source and tutorials to follow
- Contributions have begun to three.js
- Continued commitment to HTML5 for both end users and developers alike

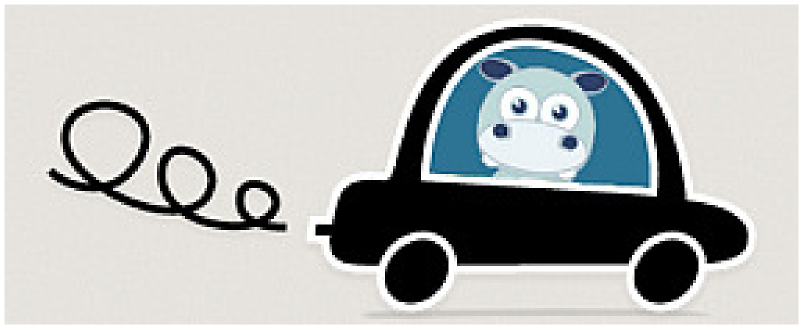
http://devblog.blackberry.com

http://www.github.com/blackberry

http://developer.blackberry.com

The Future of WebGL

- Depends on what happens today.
- Not sufficient to blindly implement technologies born on the desktop.
- APIs and Frameworks need to be brought over responsibly.
- Show a need for these technologies by participating in them today.
- Chrome experiments.



Thank You

Dustin Malik @dustinmalik

Erik Oros @WaterlooErik