WebGL, HTML5… and how the mobile web was won

image: http://www.bitrebels.com

tony parisi
Vizi, inc.
march 25, 2014

a wild web of app development

+

=

image:
http://www.arvindsaba.blogspot.com/


HTML
5
... and one platform to rule them all...

the web browser as platform

- fast JavaScript virtual machines
- hardware-accelerated compositing
- animation support
- Workers, WebSockets, local storage, local databases
- new language initiatives: Dart, Typescript, asm.js
- mobile-inspired features: location, touch, device orientation…
- mobile platforms rapidly adopting all HTML5 features in browsers and embedded WebView controls - near-ubiquity

breakthrough applications

60FPS **ported in 5 days** **Unreal native C++ engine -> JavaScript** **Emscripten + asm.js**

bringing the power of the GPU to the web

- WebGL hardware-accelerated 3D rendering
- CSS 3D hardware-accelerated transforms, transitions, and animations

the 3D API standard for the web

- OpenGL ES™ in a browser
- JavaScript API bindings
- shipped since early 2011
- supported in all modern browsers
  - desktop Safari, Firefox, Chrome, Opera, Internet Explorer
  - iOS mobile Safari – iAds only
  - Android – mobile Chrome, mobile Firefox
  - Blackberry, Tizen, Firefox OS, Amazon FireOS (Kindle Fire HDX)
  - Surface (Windows 8.1)
  - NVIDIA Shield
- over 1B served

100,000 Stars Google Experiment
http://workshop.chromeexperiments.com/stars/

how WebGL works

- JavaScript API to OpenGL ES (version 2.0)
  - draw to a canvas element using a special context
  - low-level drawing – buffers, primitives, textures and shaders
  - accelerated by graphics hardware (GPU)
  - can draw 2D as well as 3D graphics
- there is ***no file format; no markup language; no DOM.***

a simple WebGL program

1. create a `<canvas>` element
2. obtain a drawing context
3. initialize the viewport
4. create one or more buffers
5. create one or more matrices
6. create one or more shaders
7. initialize the shaders
8. draw one or more primitives

create the canvas, context and viewport

```
function initWebGL(canvas) {
        var gl = null;
        var msg = "Your browser does not support WebGL, " +
                "or it is not enabled by default.";
        try
        {
            gl = canvas.getContext("webgl");
        }
        catch (e)
        {
            msg = "Error creating WebGL Context!: " + e.toString();
        }

        if (!gl)
        {
                alert(msg);
                throw new Error(msg);
        }
    return gl;
  }

  function initViewport(gl, canvas)
  {
    gl.viewport(0, 0, canvas.width, canvas.height);
  }
```

detect WebGL

set WebGL drawing region

buffers and typed arrays

```
var vertexBuffer;

vertexBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuffer);

var verts = [
        // Front face
        -1.0, -1.0,  1.0,
         1.0, -1.0,  1.0,
         1.0,  1.0,  1.0,
        -1.0,  1.0,  1.0,

        // Back face
        -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.bufferData(gl.ARRAY_BUFFER, new Float32Array(verts), gl.STATIC_DRAW);
```

WebGL drawing functions use buffers of data

new low-level data type stores arrays of floats and ints compactly

shaders

```
var vertexShaderSource =

        "    attribute vec3 vertexPos;\n" +
        "    attribute vec2 texCoord;\n" +
        "    uniform mat4 modelViewMatrix;\n" +
        "    uniform mat4 projectionMatrix;\n" +
        "    varying vec2 vTexCoord;\n" +
        "    void main(void) {\n" +
        "                    // Return the transformed and projected vertex value\n" +
        "      gl_Position = projectionMatrix * modelViewMatrix * \n" +
        "          vec4(vertexPos, 1.0);\n" +
        "      // Output the texture coordinate in vTexCoord\n" +
        "      vTexCoord = texCoord;\n" +
        "    }\n";

var fragmentShaderSource =
        "    precision mediump float;\n" +
        "    varying vec2 vTexCoord;\n" +
        "    uniform sampler2D uSampler;\n" +
        "    void main(void) {\n" +
        "    // Return the pixel color: always output white\n" +
      "    gl_FragColor = texture2D(uSampler, vec2(vTexCoord.s, vTexCoord.t));\n" +
"}\n";
```

the vertex shader transforms model-space positions into screen space

the fragment shader outputs a color value for each pixel

drawing

```
function draw(gl, obj) {

    // clear the background (with black)
    gl.clearColor(0.0, 0.0, 0.0, 1.0);
    gl.enable(gl.DEPTH_TEST);
    gl.clear(gl.COLOR_BUFFER_BIT  | gl.DEPTH_BUFFER_BIT);

    // set the shader to use
    gl.useProgram(shaderProgram);

    // connect up the shader parameters: vertex position, texture coordinate,
    // projection/model matrices and texture
    // set up the buffers
    gl.bindBuffer(gl.ARRAY_BUFFER, obj.buffer);
    gl.vertexAttribPointer(shaderVertexPositionAttribute, obj.vertSize, gl.FLOAT, false, 0, 0);
    gl.bindBuffer(gl.ARRAY_BUFFER, obj.texCoordBuffer);
    gl.vertexAttribPointer(shaderTexCoordAttribute, obj.texCoordSize, gl.FLOAT, false, 0, 0);
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, obj.indices);

    gl.uniformMatrix4fv(shaderProjectionMatrixUniform, false, projectionMatrix);
    gl.uniformMatrix4fv(shaderModelViewMatrixUniform, false, modelViewMatrix);

    gl.activeTexture(gl.TEXTURE0);
    gl.bindTexture(gl.TEXTURE_2D, webGLTexture);
    gl.uniform1i(shaderSamplerUniform, 0);

    // draw the object
    gl.drawElements(obj.primtype, obj.nIndices, gl.UNSIGNED_SHORT, 0);
}
```

clear the canvas

set the shader

set up buffers for vertices and texture coordinates

pass transform and projection matrices to the shader

set the texture and pass to the shader

draw the object

engines and frameworks

game engines/IDEs

- Goo Engine
  http://www.gootechnologies.com/
- Verold http://verold.com/
- Turbulenz https://turbulenz.com/
- PlayCanvas
  http://www.playcanvas.com/
- Artillery Engine
  https://artillery.com/
- Sketchfab https://sketchfab.com/
- **Unreal**
  https://www.unrealengine.com/
- **Unity** **http://unity3d.com/#unity-5**

scene graph libraries/page frameworks

- Three.js
  http://threejs.org/
- SceneJS
  http://scenejs.org/
- BabylonJS
  http://www.babylonjs.com/
- Vizi
  https://github.com/tparisi/Vizi
- Voodoo.js
  http://www.voodoojs.com/
- PhiloGL
  http://www.senchalabs.org/philogl/
- tQuery
  http://jeromeetienne.github.io/tquery/

glTF: a "JPEG for 3D"

- full-featured: scene layout, cameras, lights, animations
- JSON for scene structure; binary buffers for model data
- intended for WebGL and OpenGL ES mobile applications

model from 3drt.com

CSS3 3D transforms

- translate, rotate, scale page elements with perspective

browser will render element in 3D perspective

```
.bk-list li {
        perspective: 1800px;
}
```

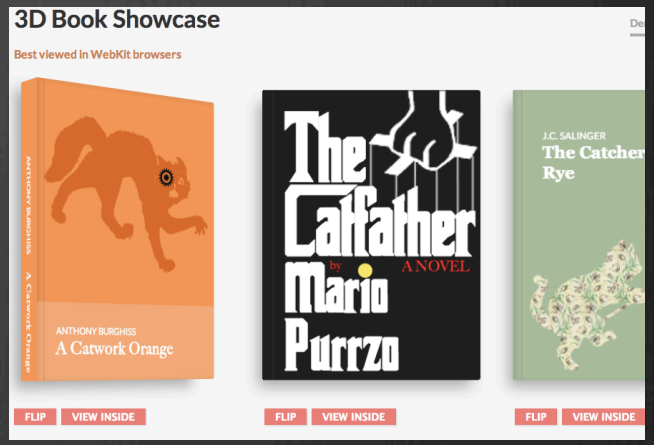
apply to child elements

```
.bk-list li .bk-front {
        transform-style: preserve-3d;
        transform-origin: 0% 50%;
        transform: translate3d(0,0,20px);
}
```

add origin to translation

apply 35 degree rotation about Y axis

```
.bk-list li .bk-book.bk-bookdefault:hover {
        transform: rotate3d(0,1,0,35deg);
}
```



http://tympanus.net/codrops/2013/01/08/3d-book-showcase/

even 2D graphics are accelerated if 3D transform functions are used e.g.

translate3d(10, 20, 0);

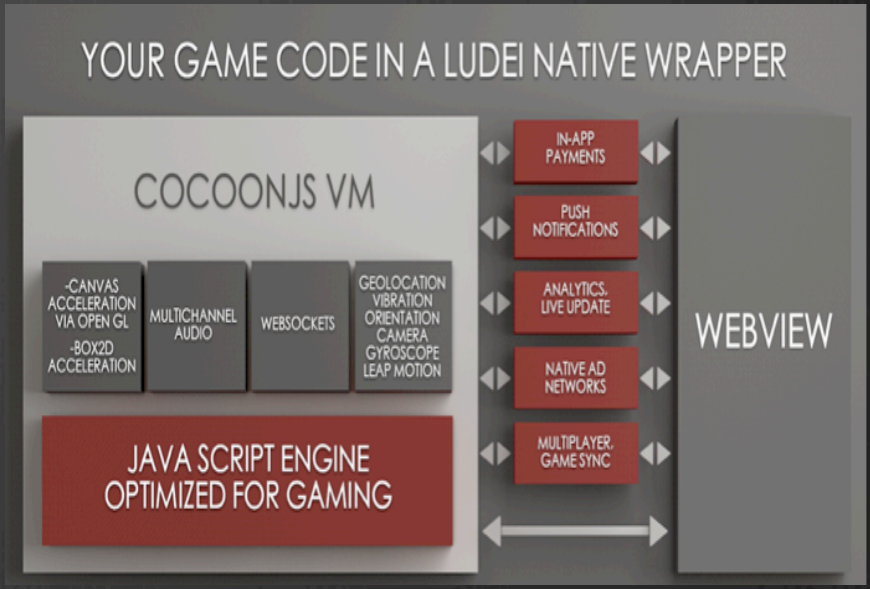
winning the mobile 3d web

- HTML5 runs in **all** mobile browsers, and as embedded WebView components in apps
- WebGL is supported in **most** mobile environments
  - iOS mobile Safari – iAds only
  - Android – mobile Chrome, mobile Firefox
  - Blackberry, Tizen, Firefox OS, Amazon FireOS (Kindle Fire HDX)
  - Surface (Windows 8.1)
  - NVIDIA Shield
- CSS 3D transforms are supported in **all** mobile environments

cross-browser HTML5 and WebGL

- desktop HTML5 and WebGL
  - all browsers support all features, nearly identically
- mobile WebGL
  - iOS - mobile Safari – iAds only
  - Android – mobile Chrome, Firefox
  - Amazon Silk, Kindle Fire OS
  - Blackberry, Tizen, Firefox OS
  - NVIDIA Shield

hybrid app development
use CocoonJS™
http://ludei.com/
or Impact Ejecta



the tipping point

- Microsoft now fully supports WebGL in IE and Windows mobile.
- Kindle Fire HDX: at $229, the 7" is probably the best multimedia device deal on the planet… thanks in part to WebGL.
- Sony built the whole PS4 user interface out of WebGL. 4.2M seats in one whack… and growing.
- the 2013 NORAD Tracks Santa site saw 48.8% WebGL success across all browsers & platforms for 20M visitors, an increase of 146% over 2012.
- Opera Devices SDK – WebGL coming soon to a Bang & Olufsen TV near you!
- pro game middleware (Unreal, Unity) fully on board
- CSS 3D transforms are supported on all platforms
- can Apple be far behind… ?

on the frontier…

image: http://www.bitrebels.com

- WebGL 2!
  - based on GL ES 3
  - will contain popular WebGL extensions
  - just getting under way – no dates yet

- WebCL
  - Heterogeneous parallel computing JavaScript
  - based on OpenCL 1.1
  - experimental –browser extensions and custom Chromium builds
  - no ship dates announced yet

stay in touch…

contact information
tparisi@gmail.com
skype: auradeluxe
http://twitter.com/auradeluxe
http://www.tonyparisi.com/
http://www.learningwebgl.com/

SF WebGL Meetup
http://www.meetup.com/WebGL-Developers-Meetup/

book source code
https://github.com/tparisi/WebGLBook
https://github.com/tparisi/Programming3DApplications

get Vizi
https://github.com/tparisi/Vizi

get the books!
WebGL: Up and Running
http://www.amazon.com/dp/144932357X
Programming 3D Applications with HTML and WebGL
http://www.amazon.com/Programming-Applications-HTML5-WebGL-Visualization/dp/1449362966