WebGL

Gregg Tavares

Google™

WebGL: What is it?

It’s OpenGL ES 2.0 in your Web Browser

- It’s fully integrated. It is NOT a plugin.
- It runs in any standards compliant browser.
  - Chrome
  - Firefox
  - Opera
  - Safari
  - Android: Coming Soon
  - iOS: Coming Soon
  - Internet Explorer: PLEASE!!!

What is this OpenGL ES 2.0 thingy?

It's OpenGL with all the cruft removed.

- No glVertex, glColor, etc... Must use vertex buffers.
- No fixed function pipeline... Must use shaders.

Why WebGL ROCKS!!!

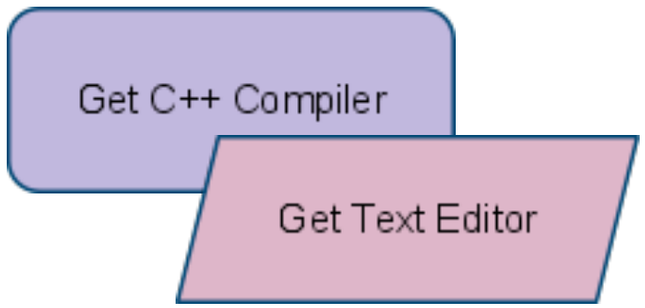
Making a graphics app in C++

Oh b0y! I'm g0ing 70 m4k3 a 1337 d3m0!

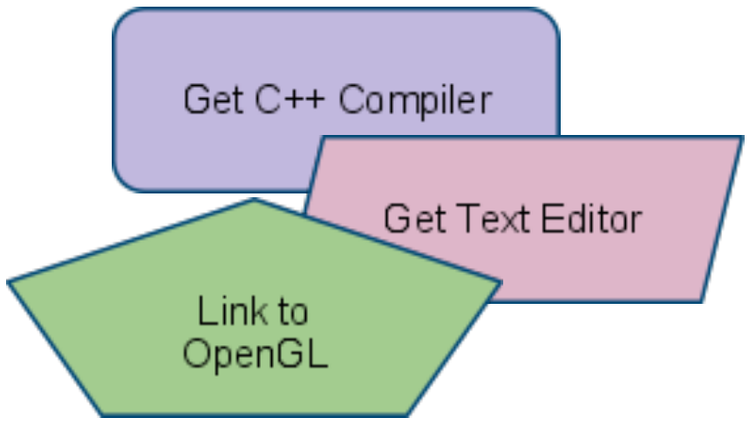
Making a graphics app in C++

Get C++ Compiler

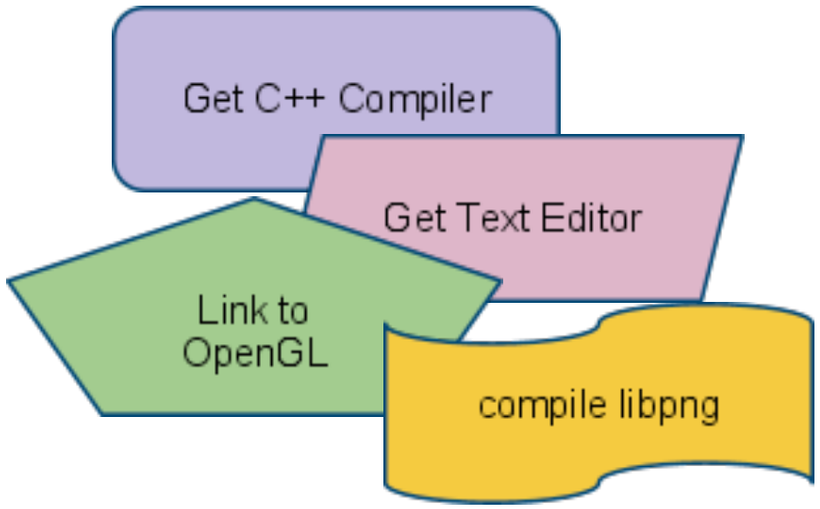
Making a graphics app in C++



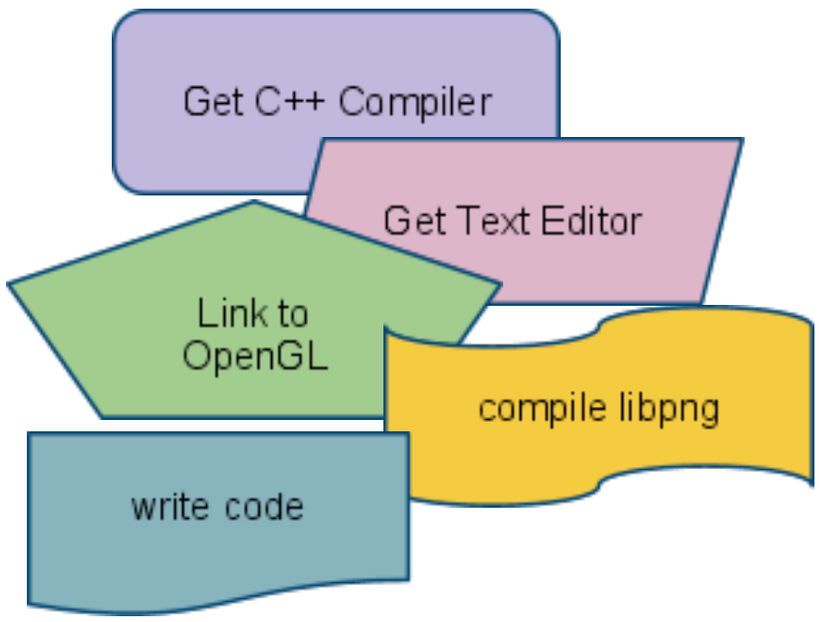
Making a graphics app in C++



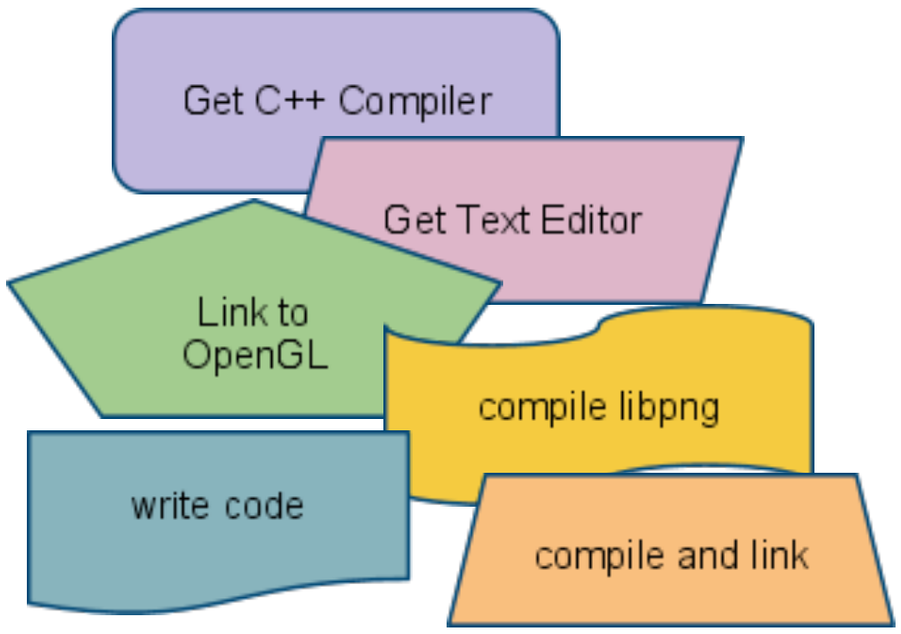
Making a graphics app in C++



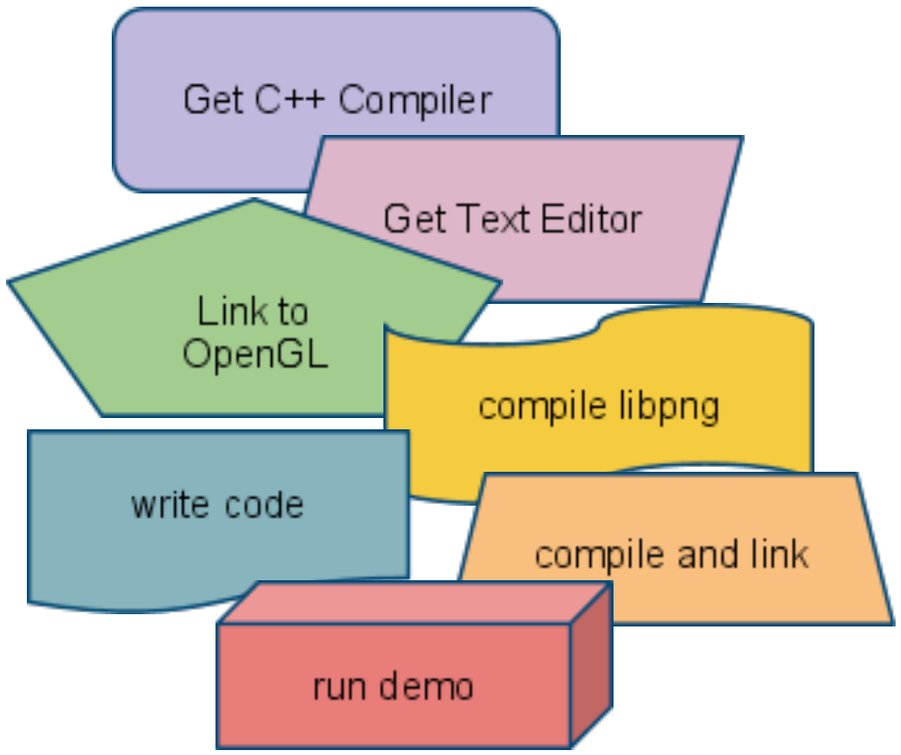
Making a graphics app in C++

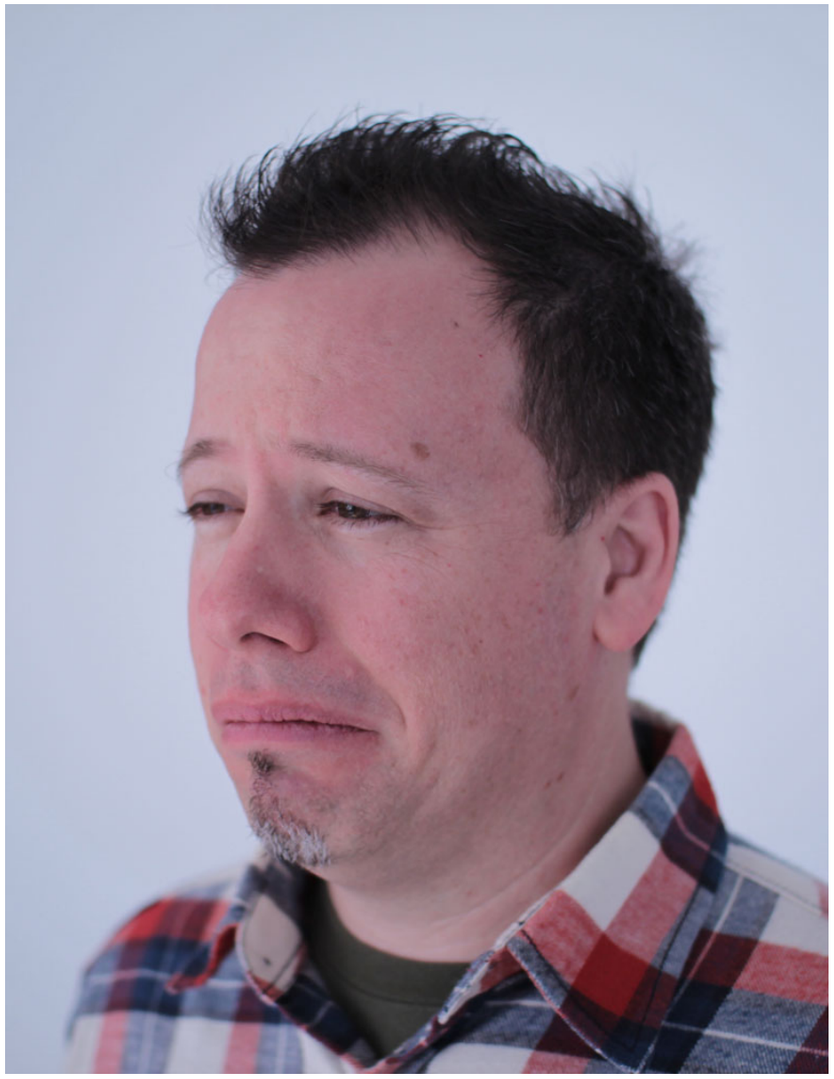


Making a graphics app in C++

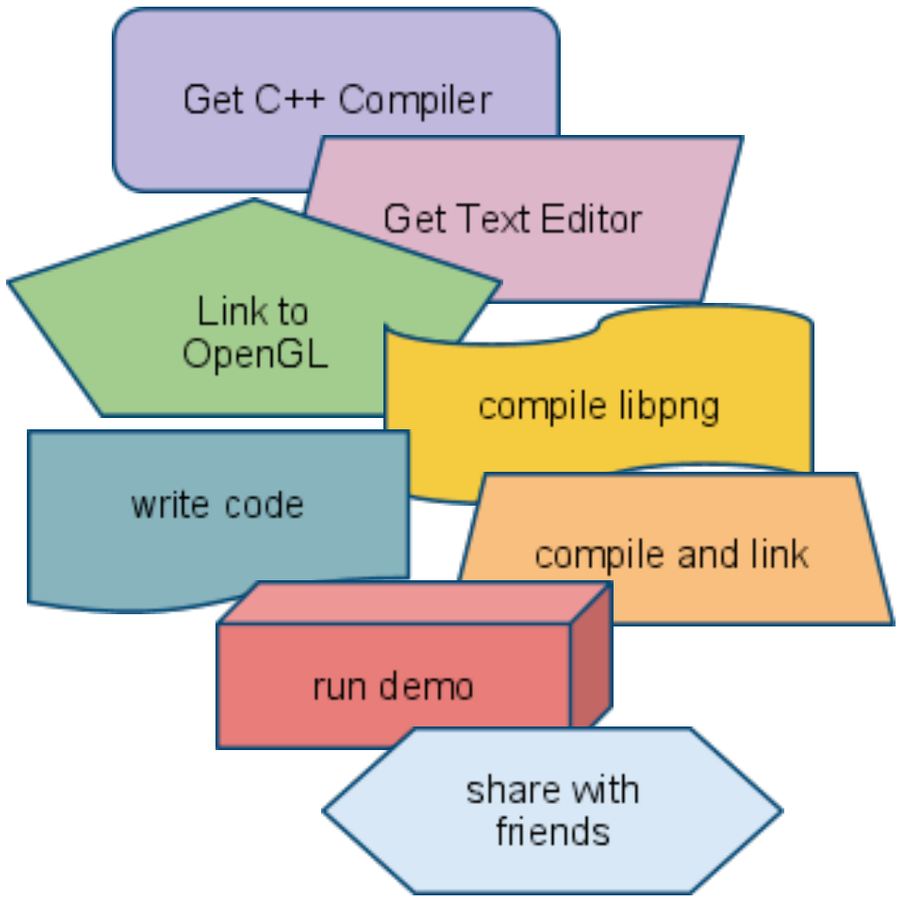


Making a graphics app in C++



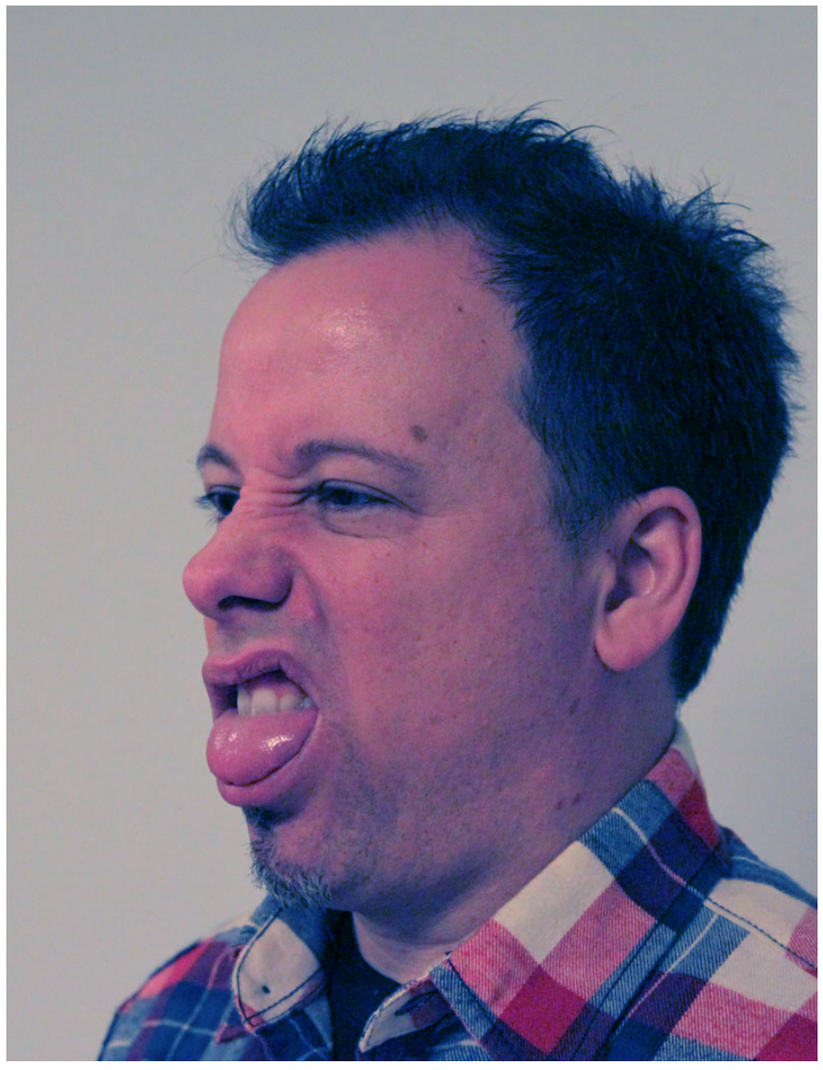


Making a graphics app in C++

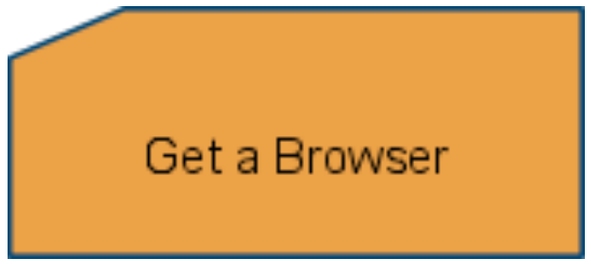


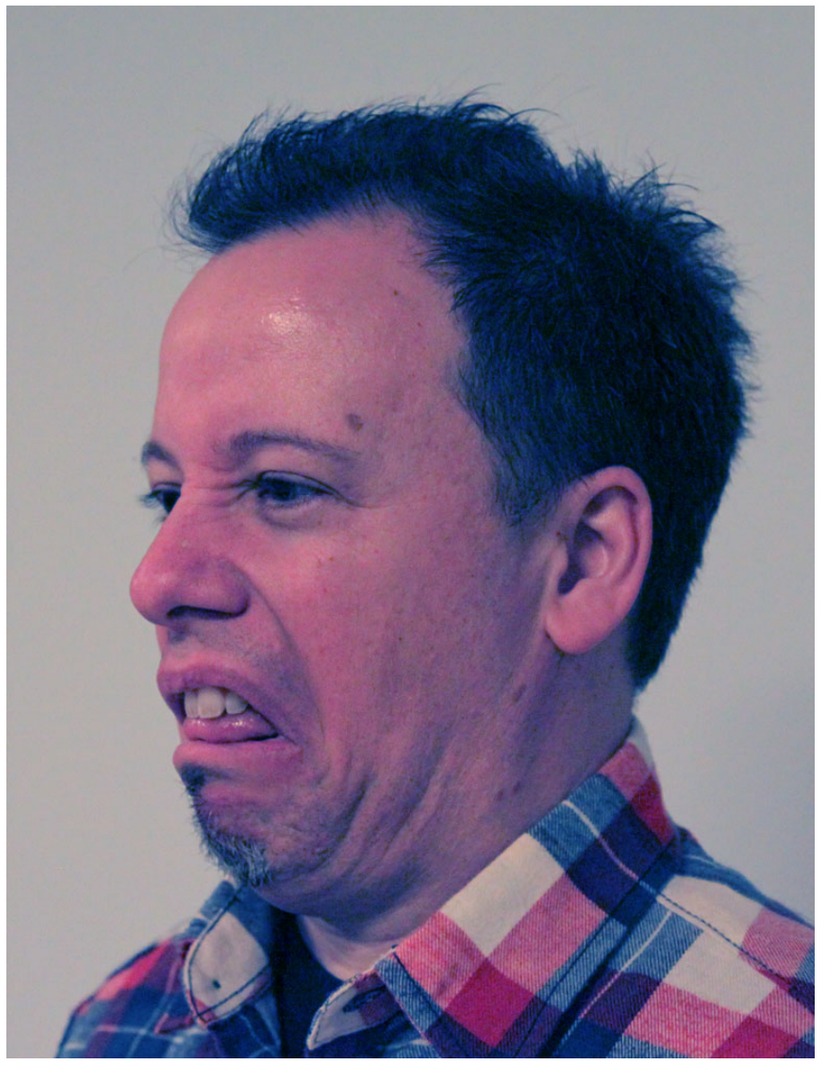
Making a graphics app in WebGL

lol wut!?!

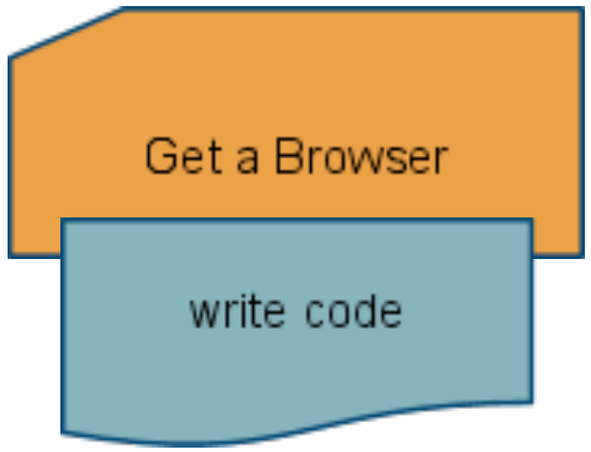


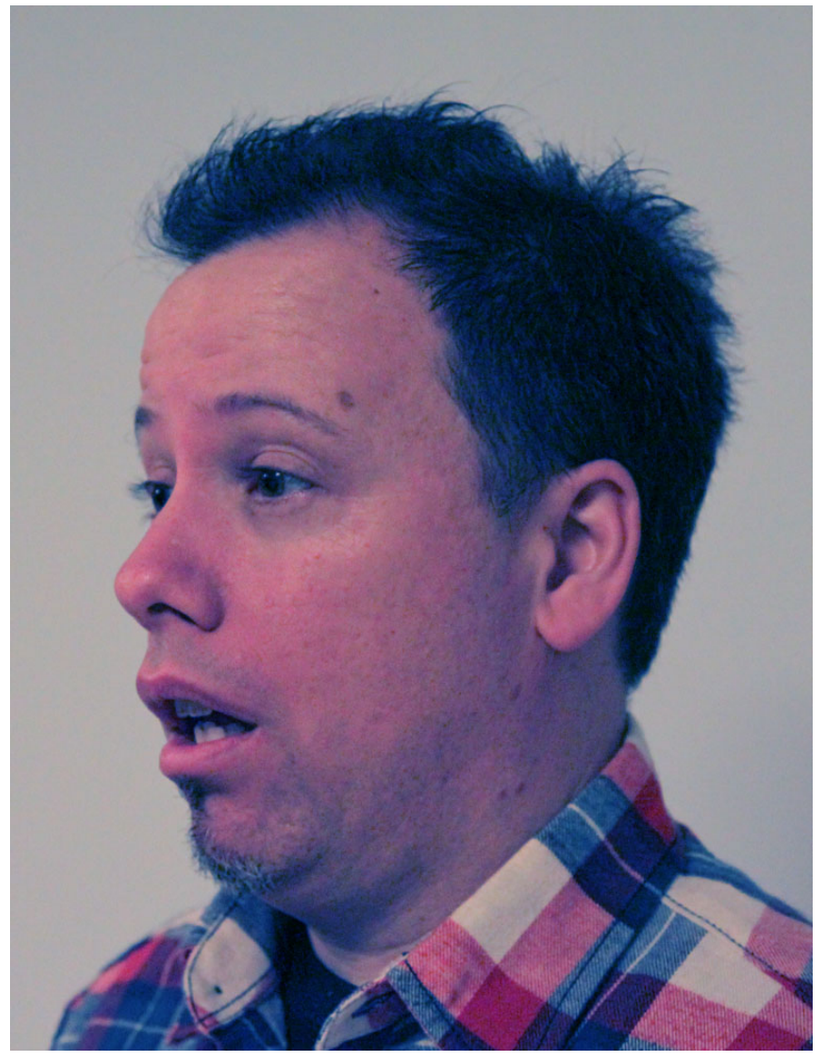
Making a graphics app in WebGL



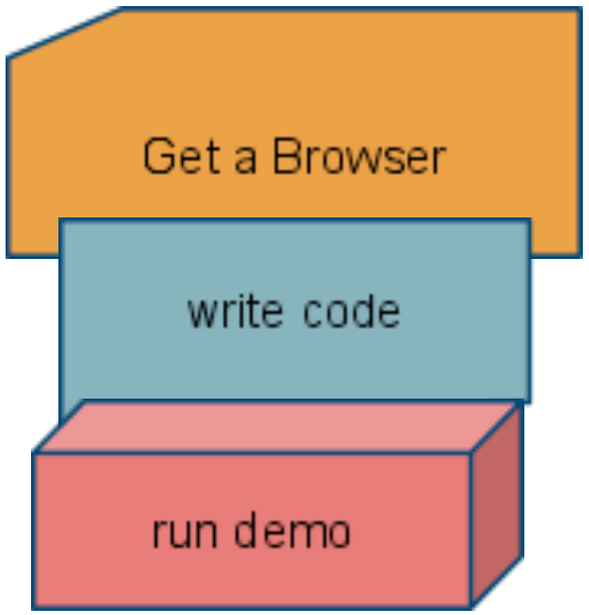


Making a graphics app in WebGL

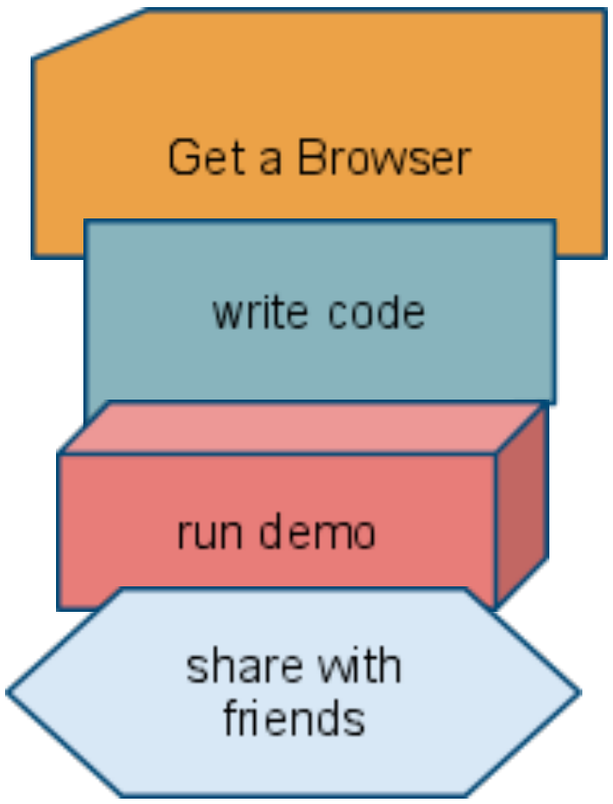


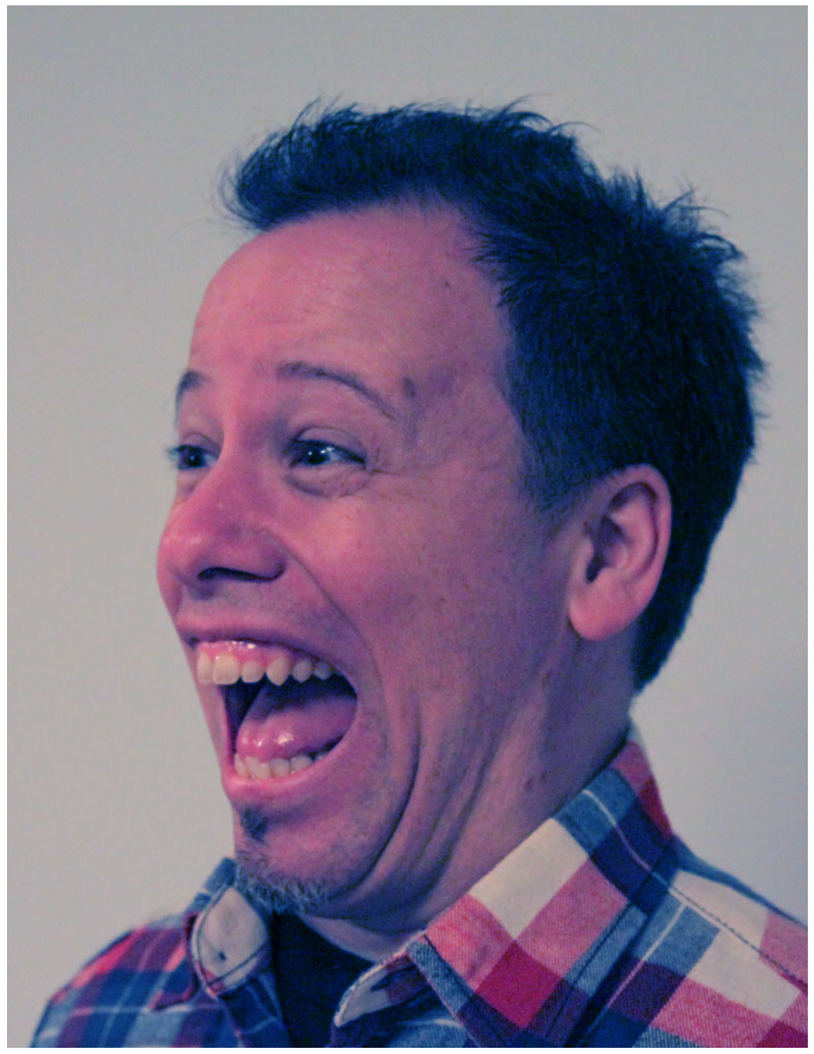


Making a graphics app in WebGL



Making a graphics app in WebGL





WebGL Code

Make a Canvas

```
<html>
    <head>
        <script>
        </script>
    </head>
    <body>
        <canvas id="myCanvas"
            width="800" height="600”>
        </canvas>
    </body>
</html>
```

Startup: Set onload

```
<script>
window.onload = main;

function main() {
}
</script>
```

Create a Context

```
<script>
window.onload = main;

function main() {
  canvas = document.getElementById("myCanvas");
  gl = canvas.getContext("webgl");
}
</script>
```

Create a Context

```
<script>
window.onload = main;

function main() {
  canvas = document.getElementById("myCanvas");
  gl = canvas.getContext("webgl");
}
</script>

// actually
//   gl = canvas.getContext(
//                   "experimental-webgl");
```

Step 1: The Smallest WebGL Program

```
<script>
window.onload = main;

function main() {
  canvas = document.getElementById("myCanvas");
  gl = canvas.getContext("webgl");
  gl.clearColor(1, 0, 1, 1);
  gl.clear(gl.COLOR_BUFFER_BIT);
}
</script>
```

Step 2: Our Demo

Looks like normal OpenGL!!

```
var vbo = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, vbo);
gl.bufferData(
    gl.ARRAY_BUFFER,
    new Float32Array([1.0, 2.0, 3.0, ...]),
    gl.STATIC_DRAW);
...
function render()
  ...
  gl.drawArrays(gl.TRIANGLES, 0, numTriangles);
  window.requestAnimationFrame(render, canvas);
}
render();
```

Step 3: Live Debugging

```
var g_lightColor = [1, 1, 1, 1];
```

Step 3: Live Debugging

```
var g_lightColor = [1, 1, 1, 1];
```

Open the debugger

Step 3: Live Debugging

```
var g_lightColor = [1, 1, 1, 1];
```

Open the debugger

```
g_lightColor = [1,0,0,1]
```

Step 4: Shader Editing

```
<textarea cols="60" rows="10" id="shader"></textarea>
<input type="button" id="apply" value="Apply"></input>
```

Step 4: Shader Editing

```
<textarea cols="60" rows="10" id="shader"></textarea>
<input type="button" id="apply" value="Apply"></input>

<script type="something-not-javascript" id="foo">
...shader code goes here...
</script>
```

Step 4: Shader Editing

```
<textarea cols="60" rows="10" id="shader"></textarea>
<input type="button" id="apply" value="Apply"></input>

<script type="something-not-javascript" id="foo">
...shader code goes here...
</script>
```

```
source = document.getElementById("foo").text;
```

Step 4: Shader Editing

```
<textarea cols="60" rows="10" id="shader"></textarea>
<input type="button" id="apply" value="Apply"></input>

<script type="something-not-javascript" id="foo">
...shader code goes here...
</script>

source = document.getElementById("foo").text;

var textarea = document.getElementById("shader");
var button = document.getElementById("apply");
```

Step 4: Shader Editing

```
<textarea cols="60" rows="10" id="shader"></textarea>
<input type="button" id="apply" value="Apply"></input>

<script type="something-not-javascript" id="foo">
...shader code goes here...
</script>

source = document.getElementById("foo").text;

var textarea = document.getElementById("shader");
var button = document.getElementById("apply");

button.onclick = function() {
  var shader = compileShader(textarea.value);
  ...
};
```

Step 5: UI

```
<div id="uiContainer">
    <div id="red"></div>
    <div id="green"></div>
    <div id="blue"></div>
</div>
```

Step 5: UI

```
<div id="uiContainer">
    <div id="red"></div>
    <div id="green"></div>
    <div id="blue"></div>
</div>
```

```
$("#red").slider({ slide: handleSlider(0) });
$("#green").slider({ slide: handleSlider(1) });
$("#blue").slider({ slide: handleSlider(2) });
```

Step 5: UI

```
<div id="uiContainer">
    <div id="red"></div>
    <div id="green"></div>
    <div id="blue"></div>
</div>
```

```
$("#red").slider({ slide: handleSlider(0) })
$("#green").slider({ slide: handleSlider(1) })
$("#blue").slider({ slide: handleSlider(2) })

function handleSlider(index) {
  return function(event, ui) {
    g_lightColor[index] = ui.value / 100;
  }
}
```

Step 6: Real time code updating.

```
setLightColor(lightColor, count);
```

Step 6: Real time code updating.

```
setLightColor(lightColor, count);

function setLightColor(dest, count) {
  dest[0] = 1;
  dest[1] = 1;
  dest[2] = 1;
}
```

Step 6: Real time code updating.

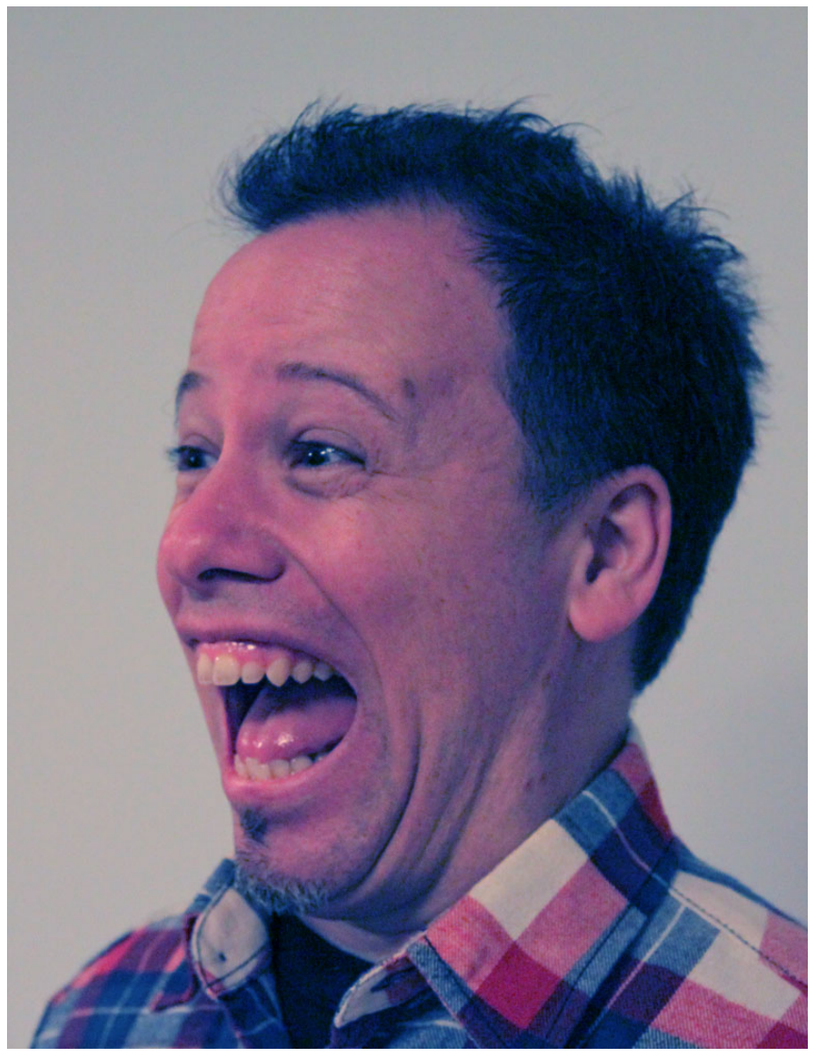
```
setLightColor(lightColor, count);

function setLightColor(dest, count) {
  dest[0] = 1;
  dest[1] = 1;
  dest[2] = 1;
}
```

```
function setLightColor(dest, count) {
  dest[0] = count / 5 % 1;
  dest[1] = count / 10 % 1;
  dest[2] = count / 15 % 1;
}
```

Yeah!!!

- OMG! They weren't kidding.
- WebGL ROXOR!



WebGL Demos

And the techniques used to make them.

Top 4 Reasons to make WebGL demos

Top 4 Reasons to make WebGL demos

#4: To promote WebGL

Top 4 Reasons to make WebGL demos

#4: To promote WebGL

#3: Because it’s fun

Top 4 Reasons to make WebGL demos

#4: To promote WebGL

#3: Because it's fun

#2: To make purddy pictures

Top 4 Reasons to make WebGL demos

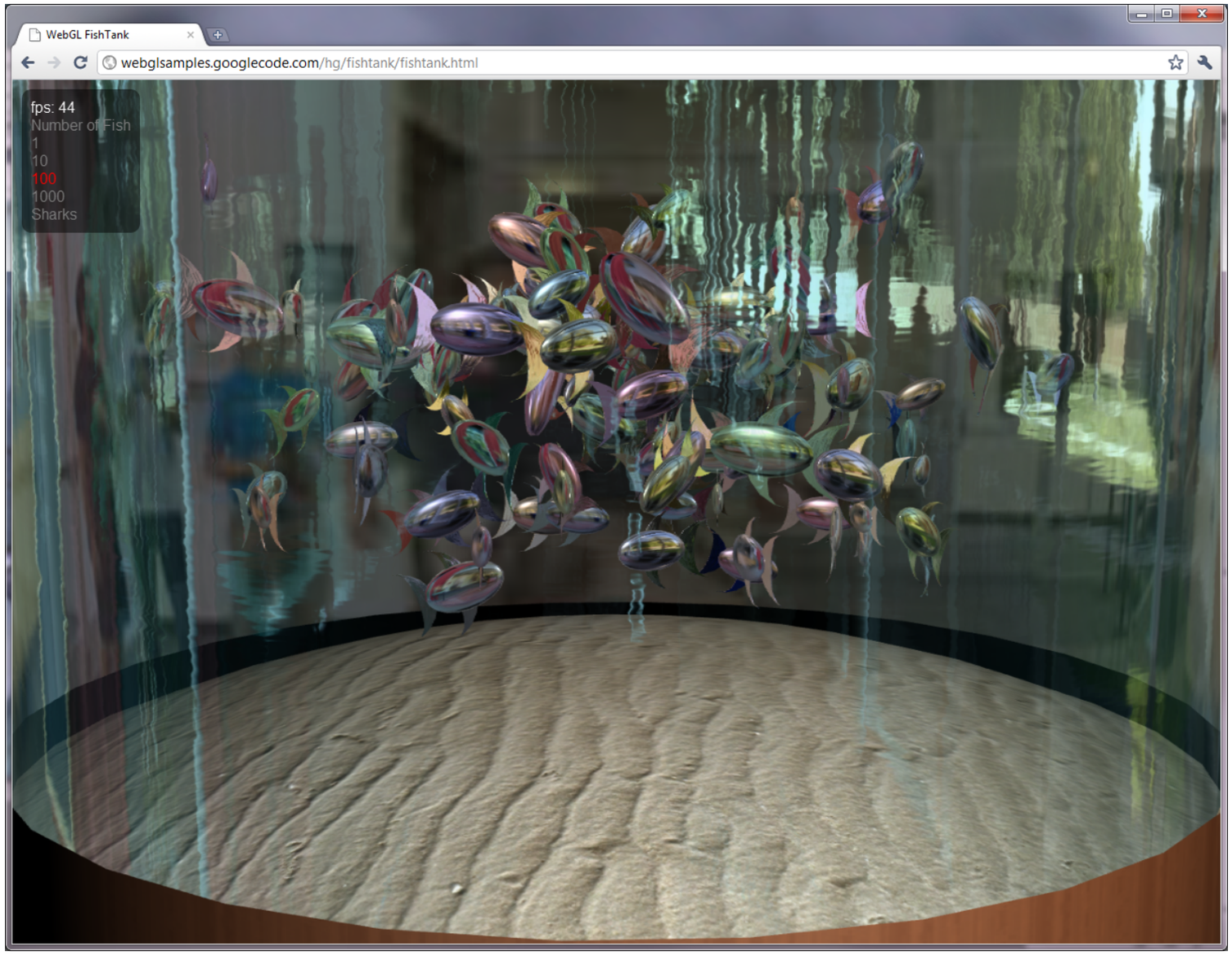
#4: To promote WebGL

#3: Because it's fun

#2: To make purddy pictures

#1: To find bugs

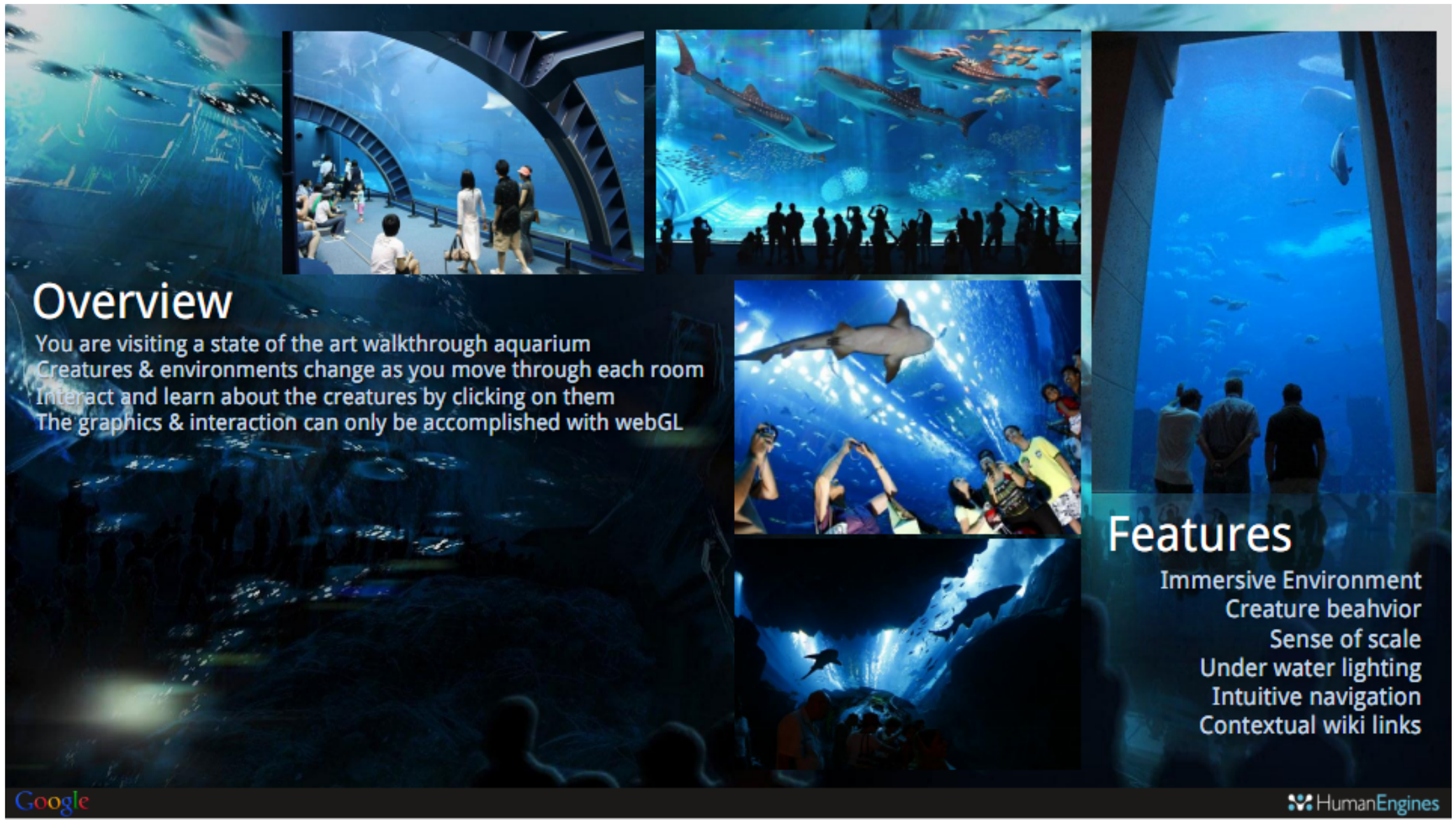
WebGL Aquarium: The original



Programmer Art FTW!

WebGL Aquarium: The artist's version

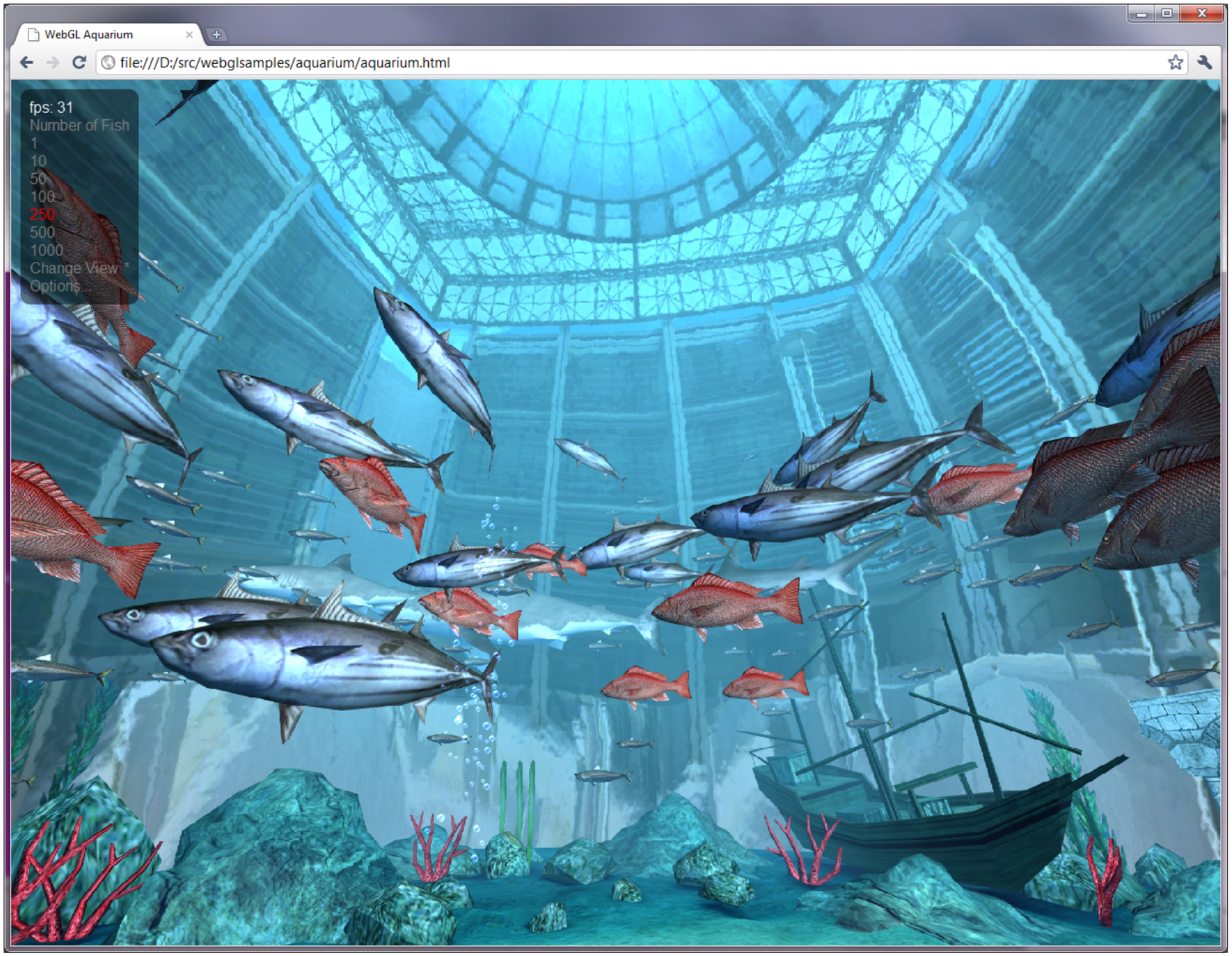
WebGL Aquarium: The artist's version



WebGL Aquarium: The artist's version

- Please kill me now (just kidding)

WebGL Aquarium



WebGL Aquarium: Details

For each frame

WebGL Aquarium: Details

For each frame

  for each fish (ie, 1000 times)

WebGL Aquarium: Details

For each frame

  for each fish (ie, 1000 times)

    compute fish position based on time

WebGL Aquarium: Details

For each frame

  for each fish (ie, 1000 times)

    compute fish position based on time

    compute next positon based on time

WebGL Aquarium: Details

For each frame
  for each fish (ie, 1000 times)
    compute fish position based on time
    compute next positon based on time
    use lookAt() to compute a world matrix

WebGL Aquarium: Details

For each frame
  for each fish (ie, 1000 times)
    compute fish position based on time
    compute next positon based on time
    use lookAt() to compute a world matrix
    compute a worldViewProjection

WebGL Aquarium: Details

For each frame
  for each fish (ie, 1000 times)
    compute fish position based on time
    compute next positon based on time
    use lookAt() to compute a world matrix
    compute a worldViewProjection
    compute a worldInverse

WebGL Aquarium: Details

For each frame
  for each fish (ie, 1000 times)
    compute fish position based on time
    compute next positon based on time
    use lookAt() to compute a world matrix
    compute a worldViewProjection
    compute a worldInverse
    compute a worldInverseTranspose

WebGL Aquarium: Details

For each frame
  for each fish (ie, 1000 times)
    compute fish position based on time
    compute next positon based on time
    use lookAt() to compute a world matrix
    compute a worldViewProjection
    compute a worldInverse
    compute a worldInverseTranspose
    set uniforms for all those values

WebGL Aquarium: Details

For each frame
  for each fish (ie, 1000 times)
    compute fish position based on time
    compute next positon based on time
    use lookAt() to compute a world matrix
    compute a worldViewProjection
    compute a worldInverse
    compute a worldInverseTranspose
    set uniforms for all those values
    gl.drawElements

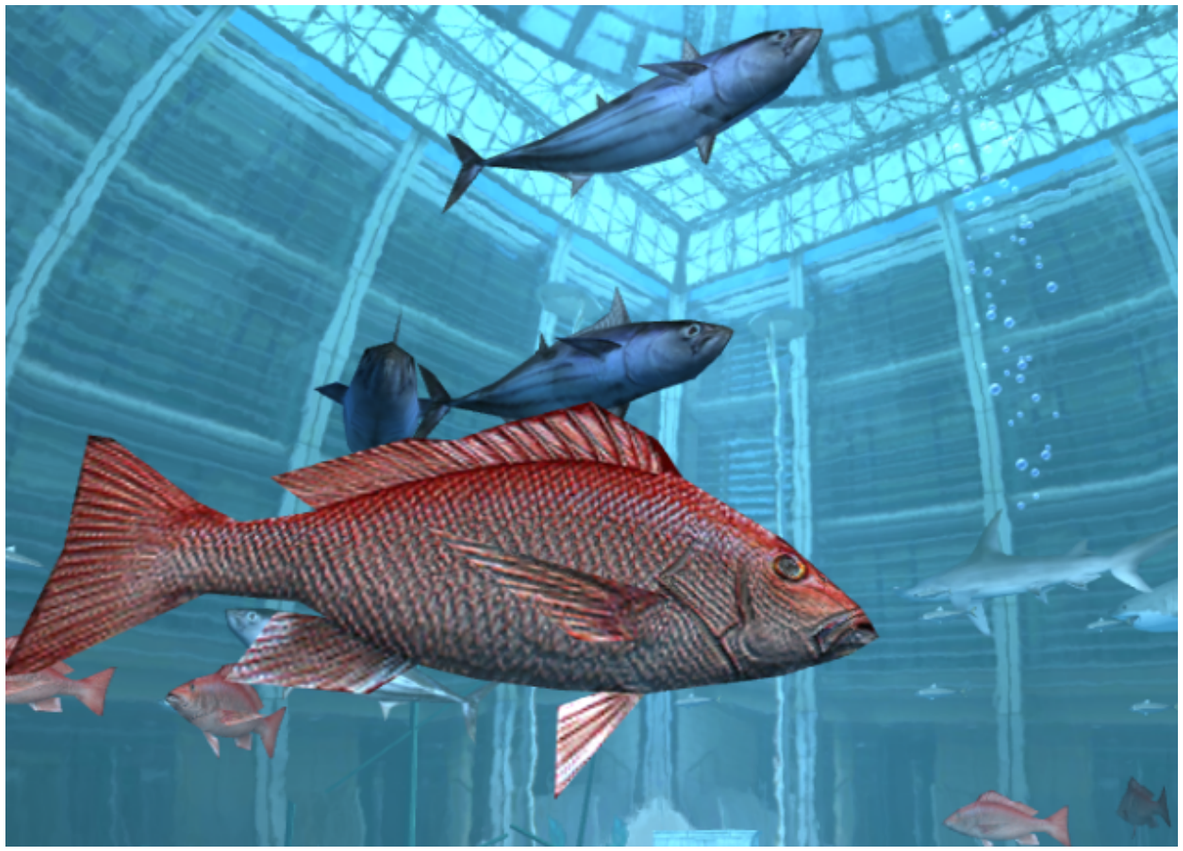
WebGL Aquarium: Details

```
For each frame
   for each fish (ie, 1000 times)
      compute fish position based on time
      compute next positon based on time
      // moved to Vertex Shader
      // use lookAt() to compute a world matrix
      // compute a worldViewProjection
      // compute a worldInverse
      // compute a worldInverseTranspose
      set uniforms for all those values
      gl.drawElements
```

WebGL Aquarium: Normal Maps

Put normal maps in .PNG files

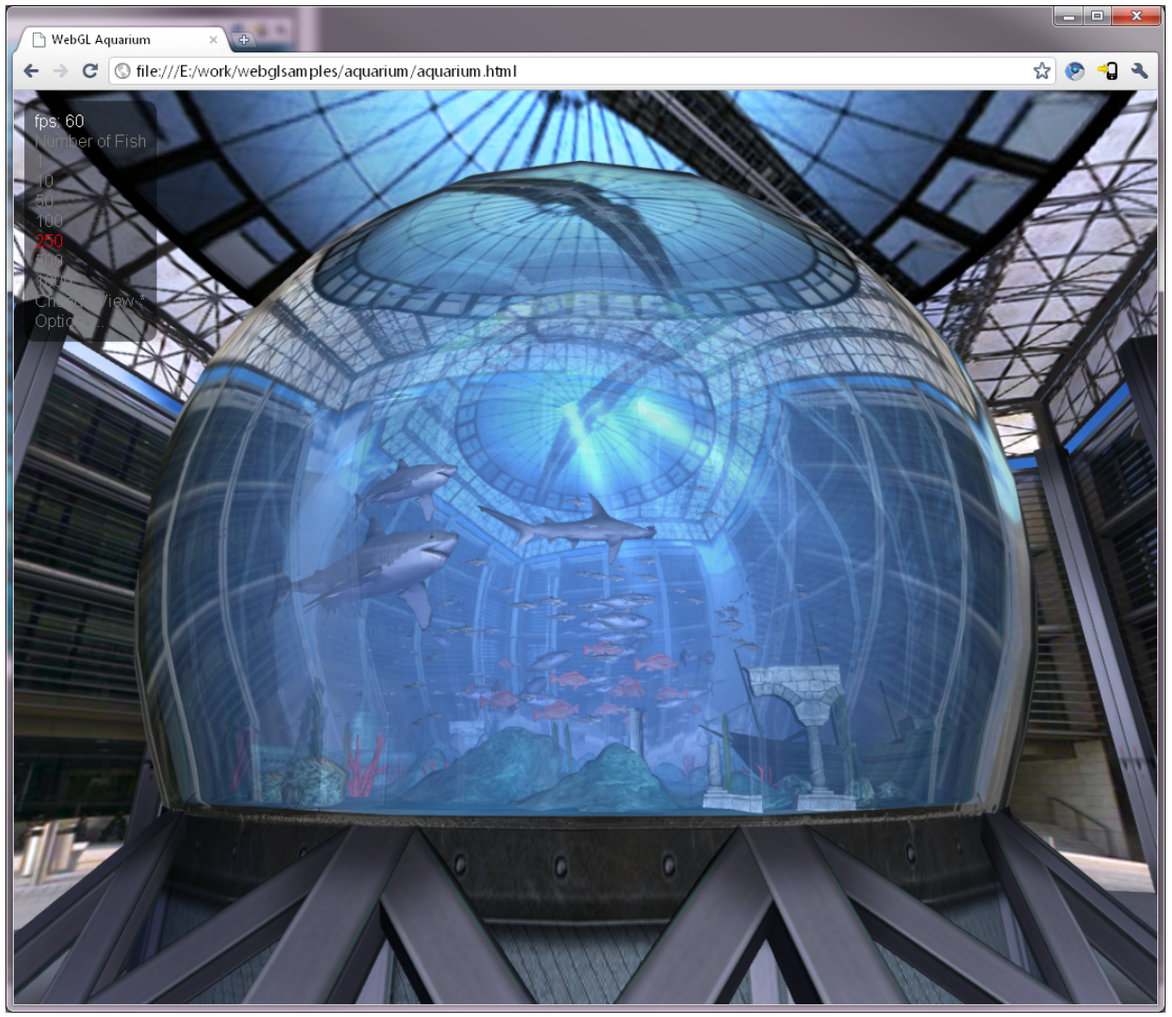
Set gl.pixelStorei(
gl.UNPACK_COLORSPACE_CONVERSION_WEBGL,
false);



WebGL Aquarium: Environment Mapping and Refraction

Cubemaps loaded as .jpgs

canvas makes them easy to manipulate



WebGL Aquarium: Reflection Maps

Use .PNG files.

```
gl.pixelStorei(
      gl.UNPACK_PREMULTIPLY_ALPHA_WEBGL, false);
```

WebGL Aquarium: GPU Particles

Bubbles are GPU based particles.

WebGL Aquarium: Etc...

Time based simulation

1000 fish at 60hz ...?

Low-Poly Meshes

(small fish 60 tris?, large 250 tris)

<10 meg textures

JQueryUI

WebGL Aquarium: Liquid Galaxy

8 monitors, 8 computers

WebGL Aquarium: Liquid Galaxy

First attempt:

- Python based server
  (python -m SimpleHTTPServer)
- XMLHttpRequest (XHR)
- Main machine sends settings to server
  through XHR
- All machines poll the server for settings.

worked but slow...

WebGL Aquarium: Liquid Galaxy

Second attempt ... WebSockets!

WebGL Aquarium: WebSockets

```
socket = new WebSocket(urlToServer);
```

WebGL Aquarium: WebSockets

```
socket = new WebSocket(urlToServer);

socket.onmessage = function(event) {
    console.log("received: " + event.data);
}
```

WebGL Aquarium: WebSockets

```
socket = new WebSocket(urlToServer);

socket.onmessage = function(event) {
   console.log("received: " + event.data);
}
```

```
socket.send("hello world");
```

WebGL Aquarium: WebSockets

```
socket = new WebSocket(urlToServer);

socket.onmessage = function(event) {
    var obj = JSON.parse(event.data);
}
```

```
socket.send(JSON.stringify(obj));
```

WebGL Aquarium: WebSockets

```
socket = new WebSocket(urlToServer);

socket.onmessage = function(event) {
    var obj = JSON.parse(event.data);
}
```

```
socket.send(JSON.stringify(obj));
```

```
var obj = {
  hitpoints: 999, name: "gman",
  position: [1.1, 2.2, 3.3],
};
```

WebGL Aquarium: Liquid Galaxy 2.0

Second attempt:

node.js
Websockets
socket.io
main machine sends settings to server
server broadcasts settings to all machines

FAST!!! fyi: WebSockets is in the dog house :-(

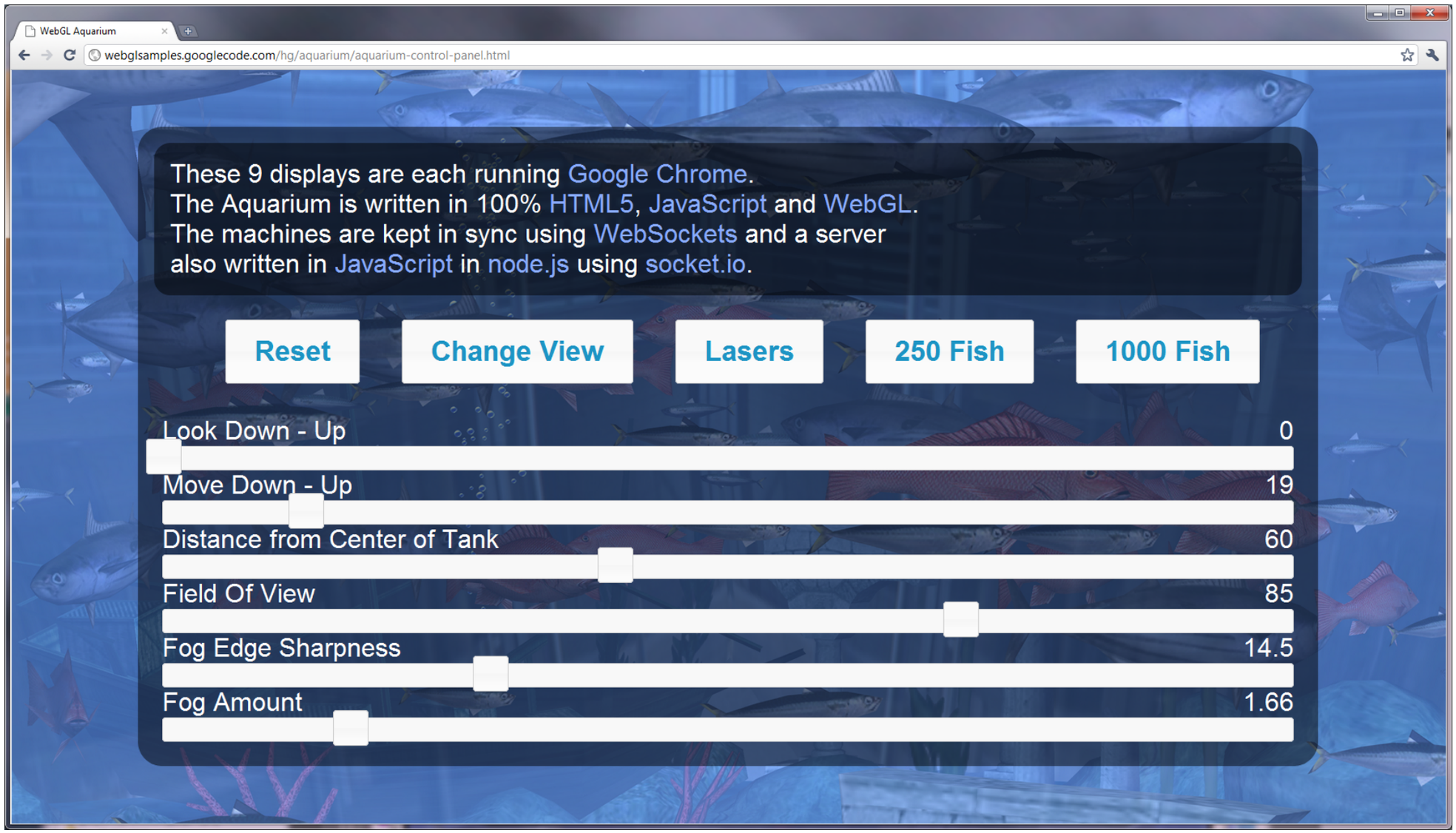
Clock Sync:

- ntpdate = good
- doing it myself = better

WebGL Aquarium: Liquid Galaxy

http://www.youtube.com/watch?v=64TcBiqmVko

WebGL Aquarium: LG Control Panel

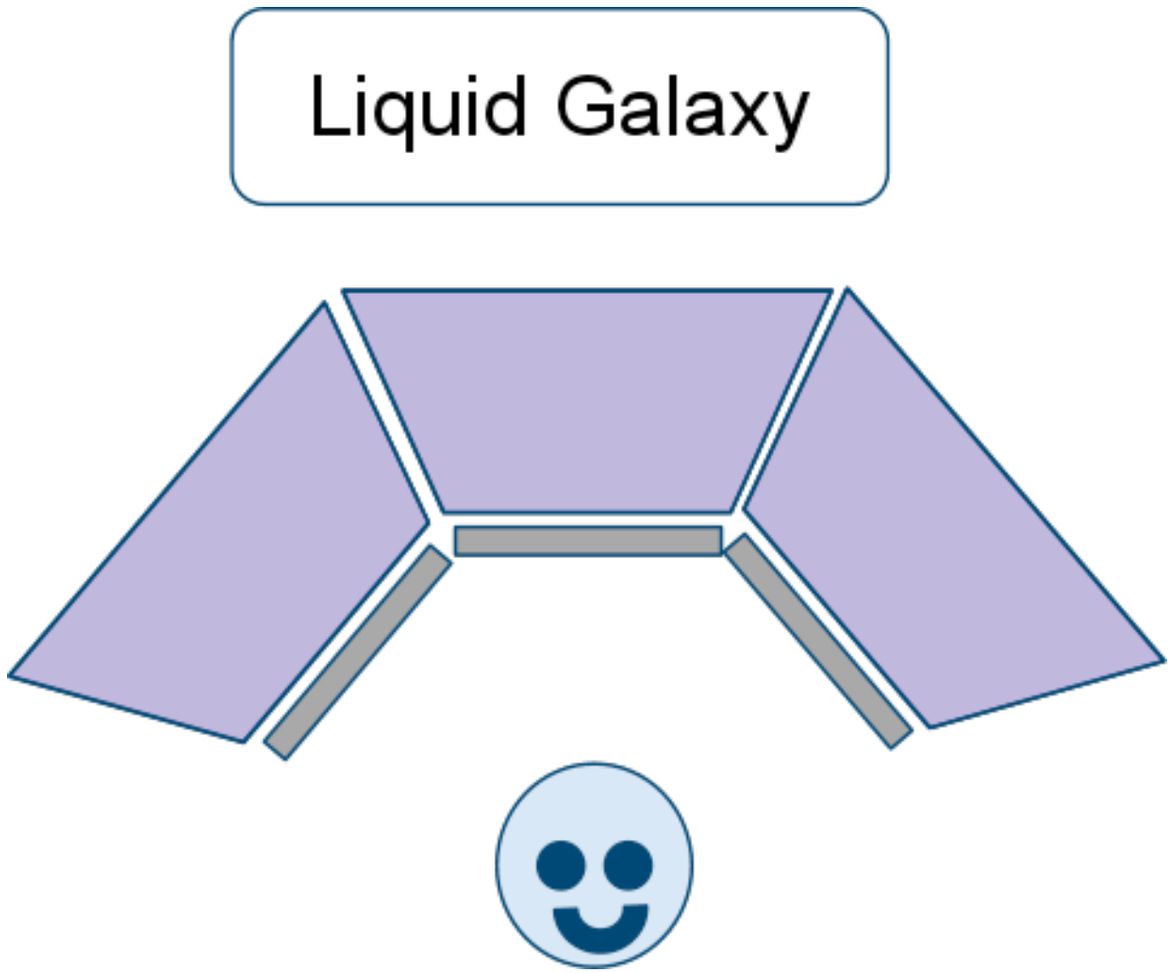


Nearly free!

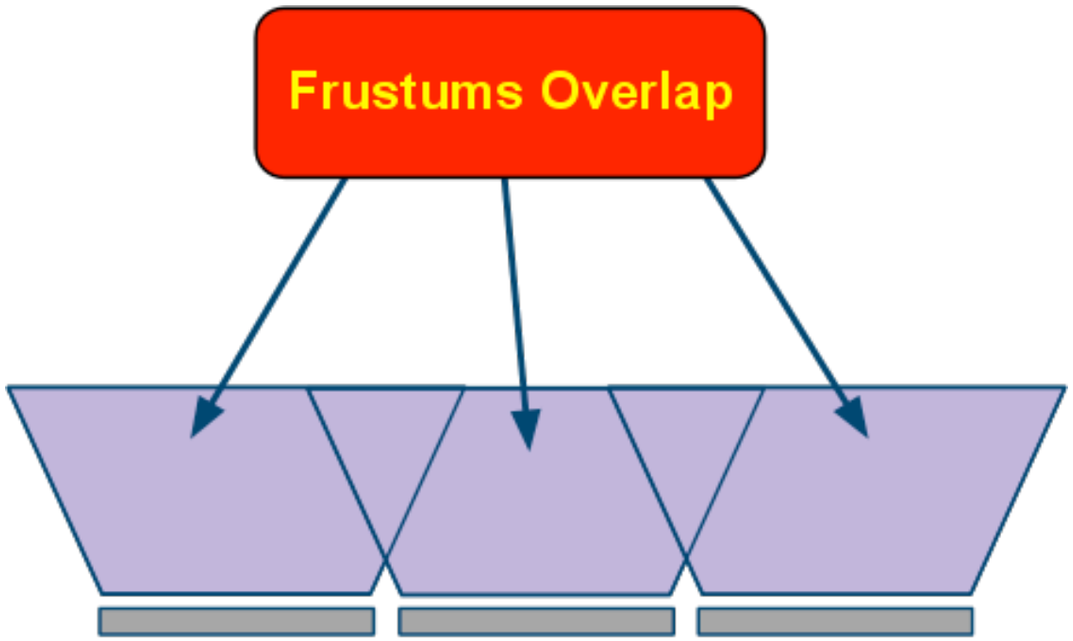
WebGL Aquarium: Virtual Monitor

Using multiple monitors to make 1 big monitor

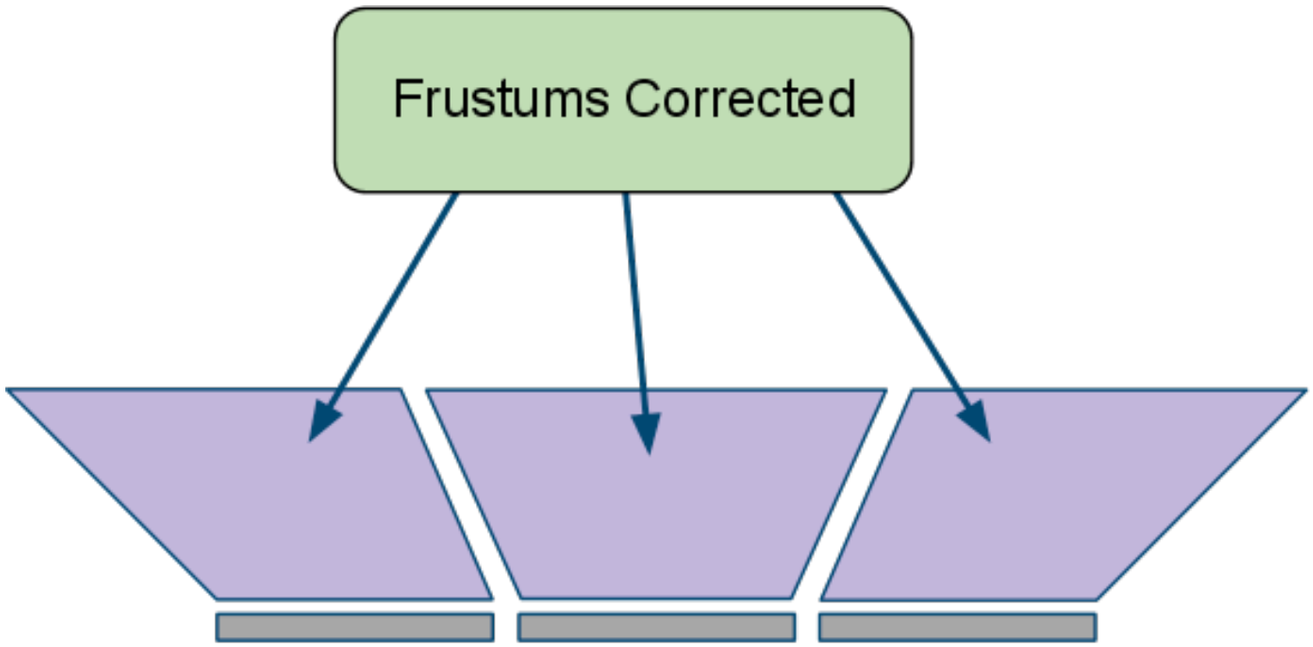
WebGL Aquarium: Virtual Monitor



WebGL Aquarium: Virtual Monitor

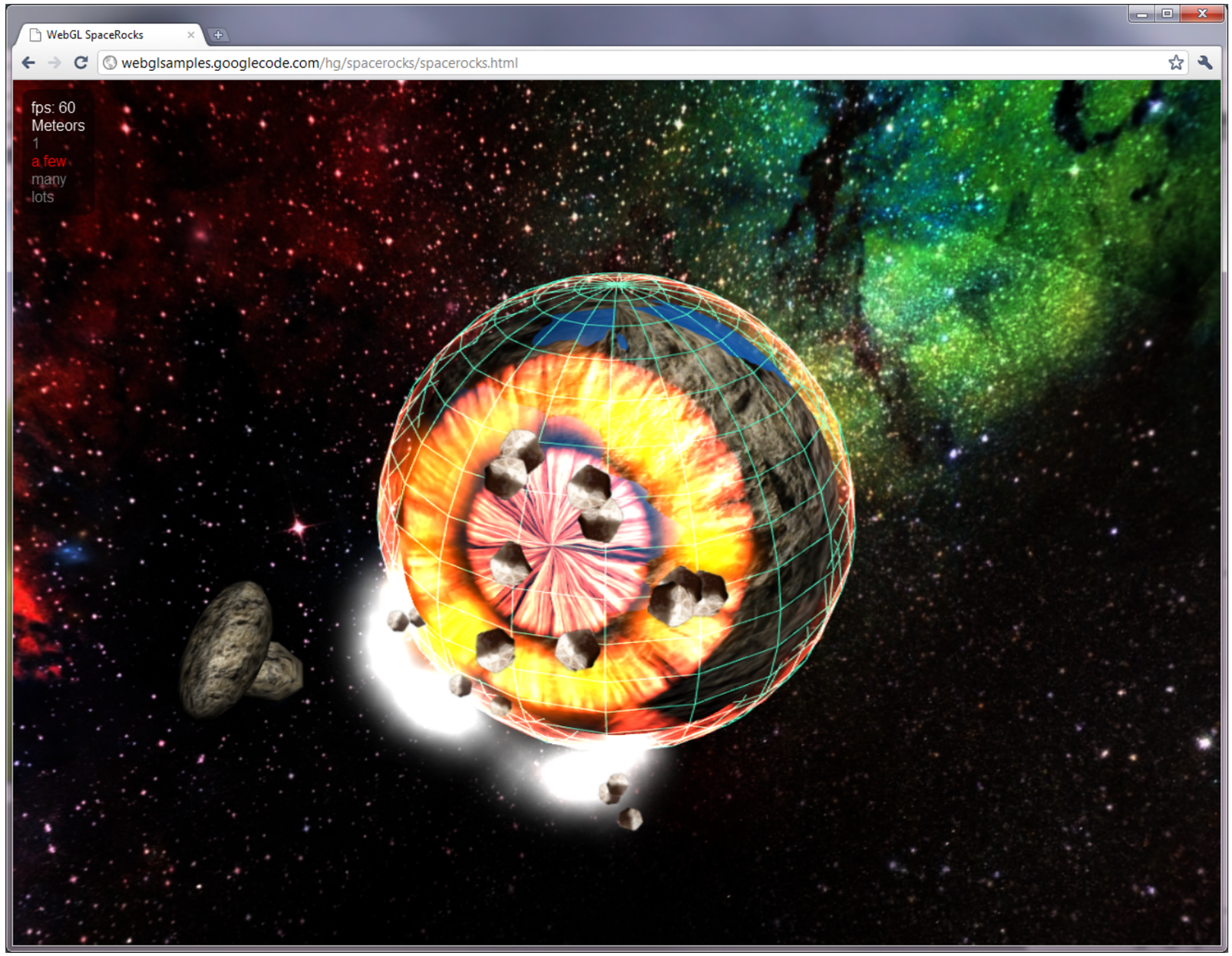


WebGL Aquarium: Virtual Monitor



see: glFrustum

WebGL Spacerocks



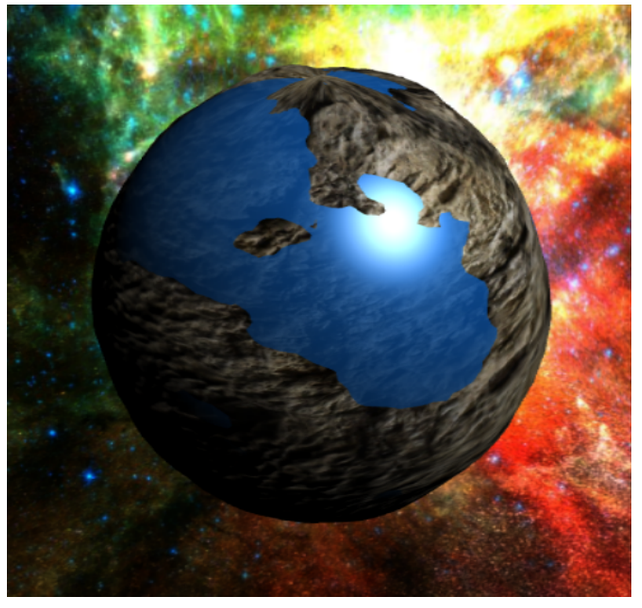
WebGL Spacerocks: Planet

Used Displacement map

Vertex Texture Fetch not available on all systems

WebGL allowed to return 0 for

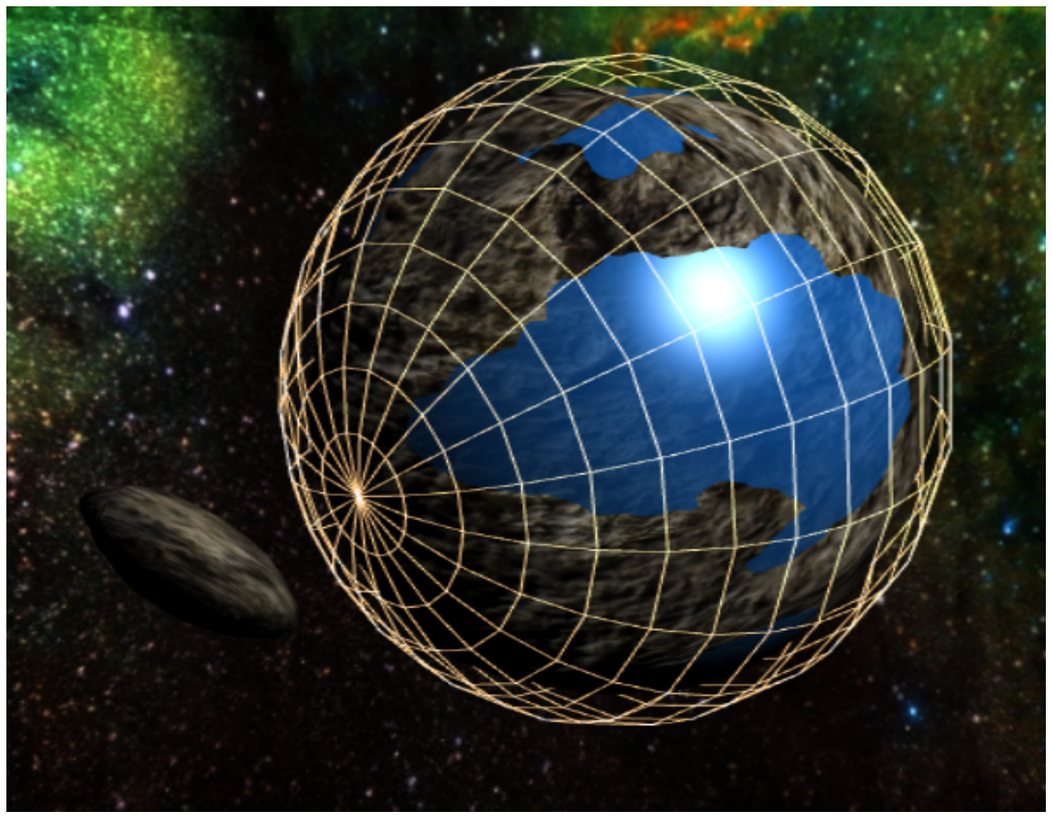
```
ctx.getParameter(
  ctx.MAX_VERTEX_TEXTURE_IMAGE_UNITS)
```

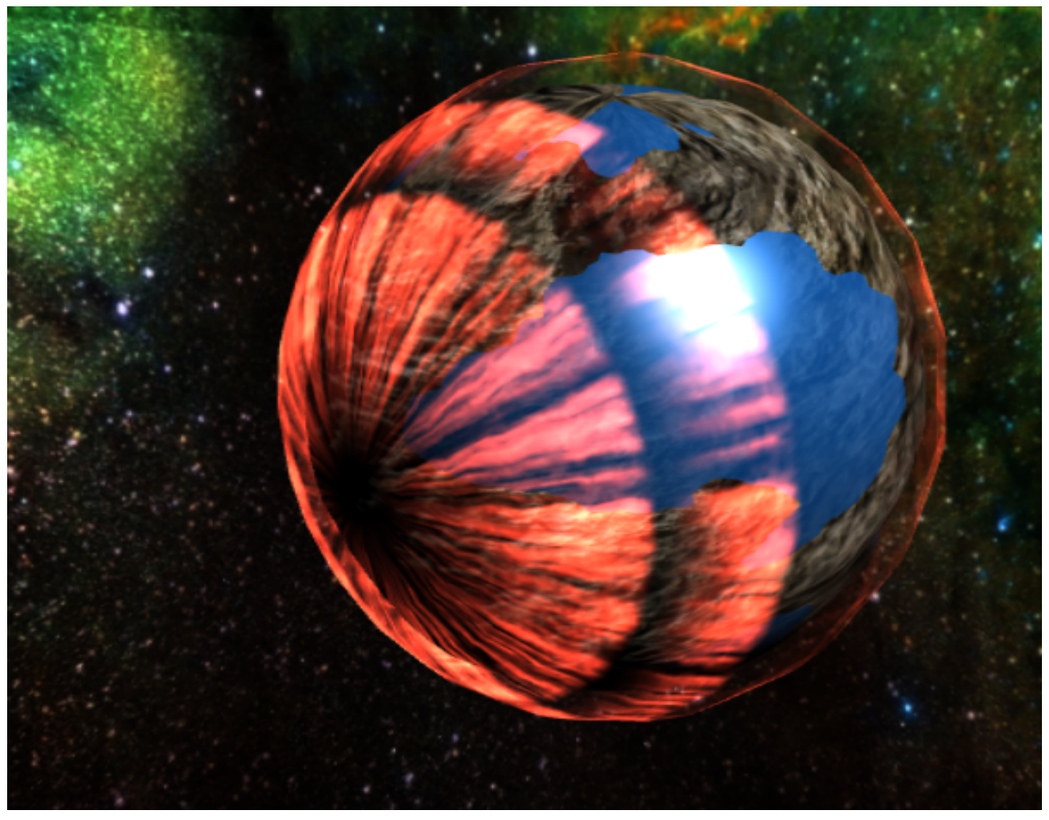


WebGL Spacerocks: Force Field

Texture scrolling for shield

20 spheres in queue, rotated to match meteor collision

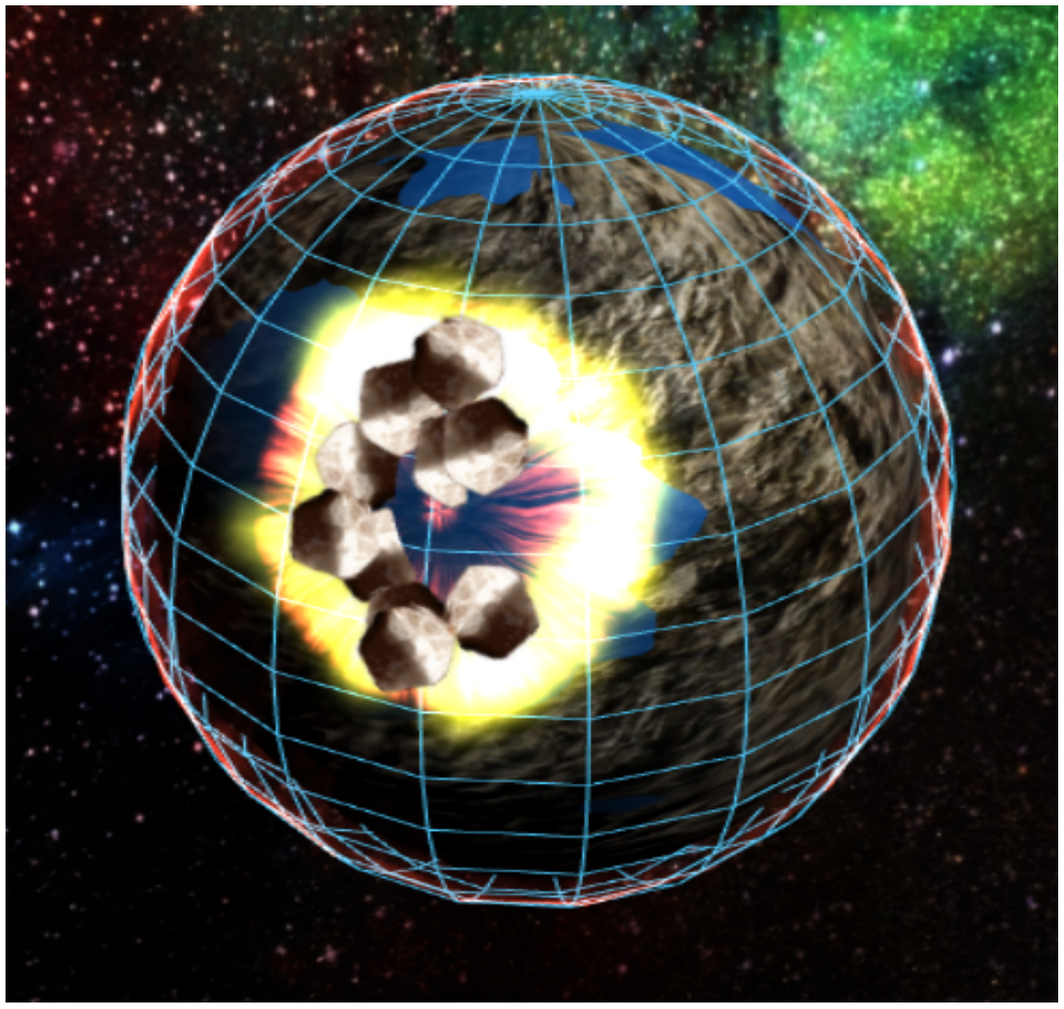




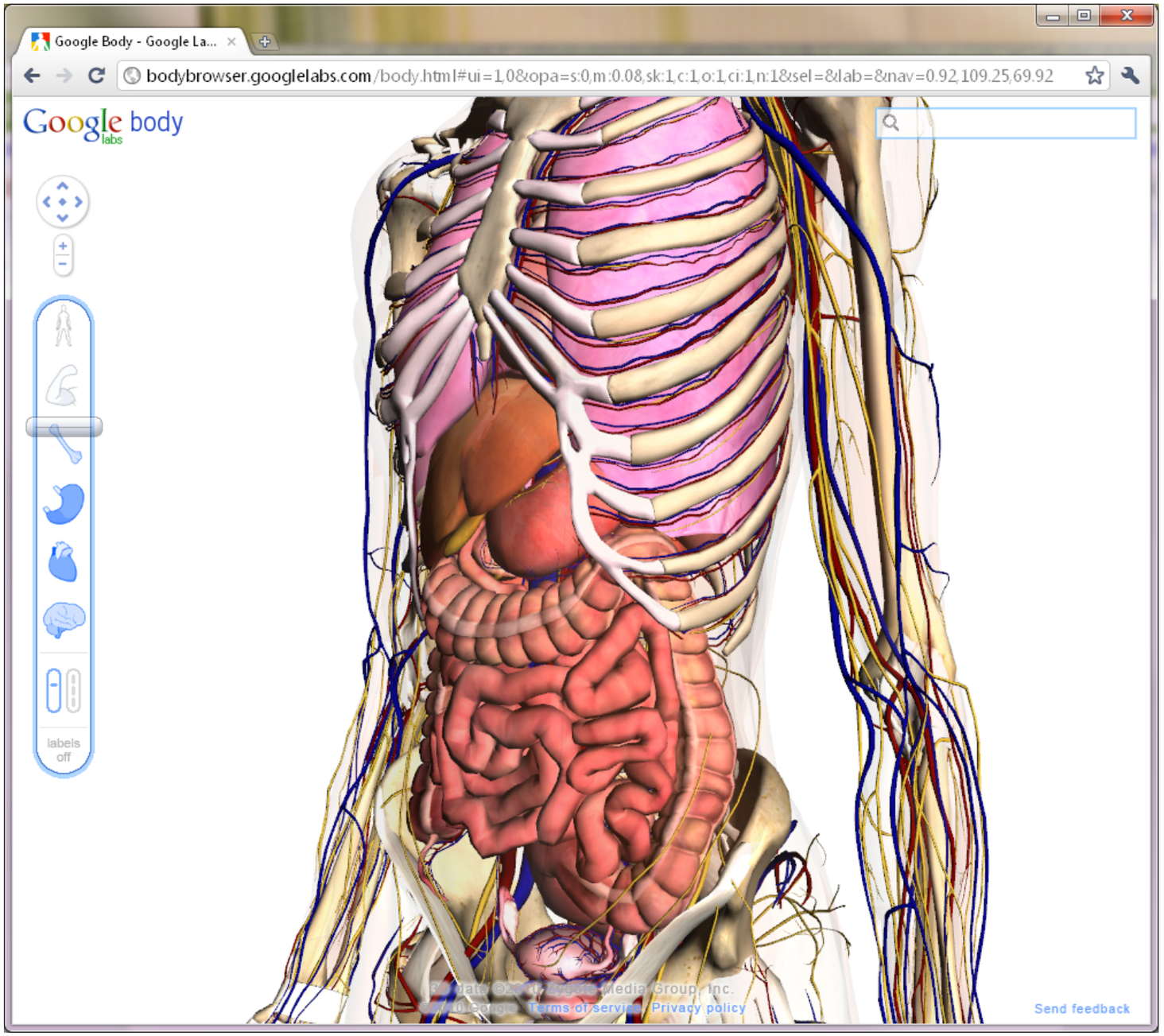
WebGL Spacerocks: Explosions

GPU based Particle system for explosions

- flames
- smoke
- debris



Google Body


Google Body - Google La...
bodybrowser.googlelabs.com/body.html#ui=1,0&opa=s:0,m:0.08,sk:1,c:1,o:1,ci:1,n:1&sel=&lab=&nav=0.92,109.25,69.92
Google labs body
labels off
Terms of service
Privacy policy
Send feedback

Google Body Details

- Highly compressed vertex data.
- 1.5 Million triangles at about 6 bytes per triangle.
- Plans to open source compressor

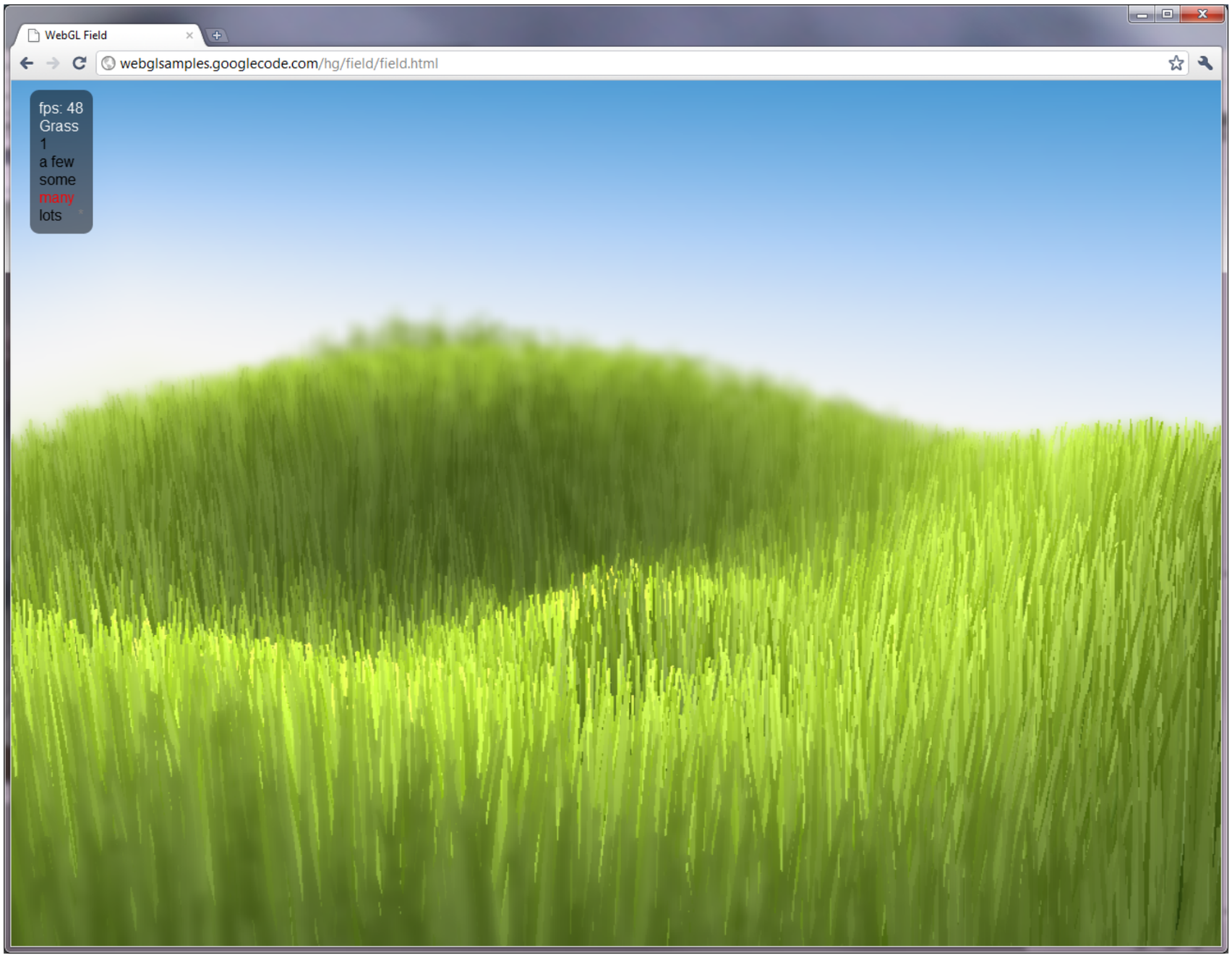
http://bodybrowser.googlelabs.com

Google Body Compression

- Quantize 14bit positions, 8bit normals, 10bit uvs
- Sort 20, 2, -6, 25 -> 2, -6, 20, 25
- Delta 2, -6, 20, 25 -> 2, -8, +28, +10
- ZigZag 2, -8, +58,+40 -> 2, 16, 57, 19
- UTF-8 most deltas are small so 1 byte each
- gzip browser un-gzips for you.

Linear-Speed Vertex Cache Optimization
by Tom Forsyth
http://goo.gl/H8fmv (sorting algorithm)

WebGL Field

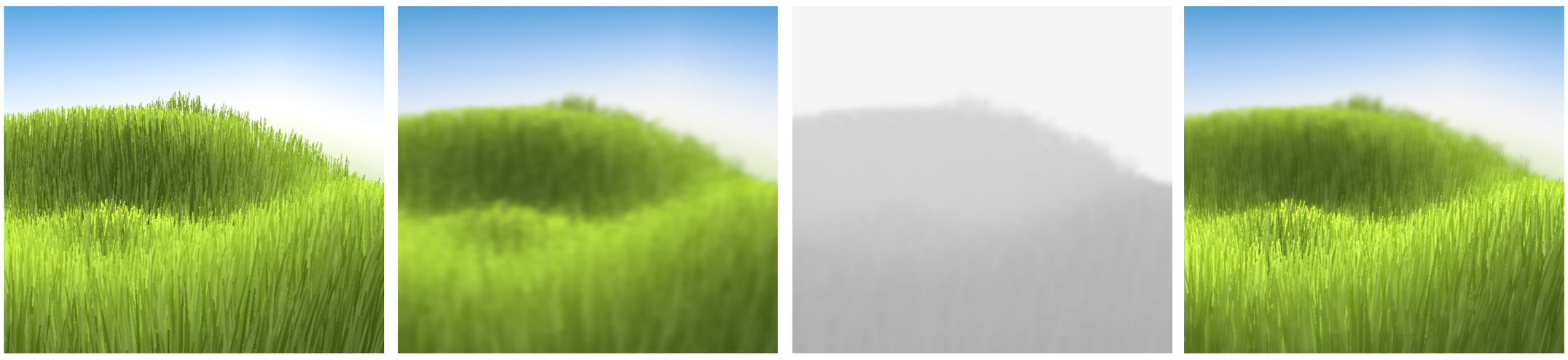

WebGL Field
webglsamples.googlecode.com/hg/field/field.html
fps: 48
Grass
1
a few
some
many
lots

WebGL Field: Depth of Field

Render to texture with depth in alpha

Make a blurred copy by rendering to another texture.

Render to backbuffer blending between original



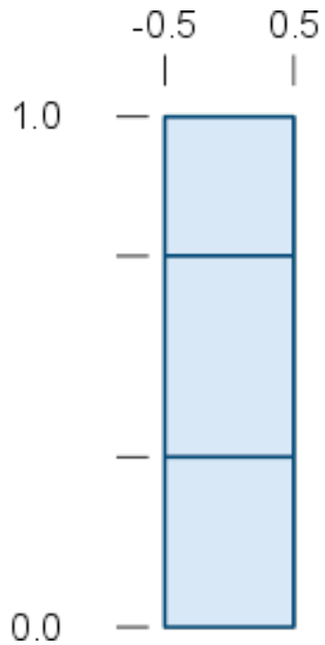
original blurred depth blended

WebGL Field: Grass Blades

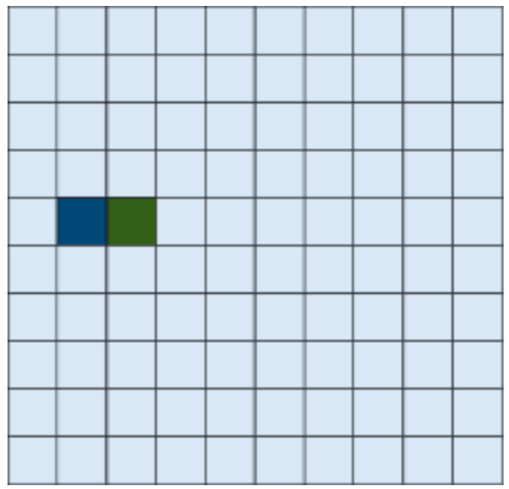
100 blades per patch.

Each blade has an id and 3 random numbers

WebGL Aquarium: Grass Blades



1 blade



10x10 blades

X	Y	Z	BladeX	BladeY
-0.5	1.0	0	1	4
+0.5	1.0	0	1	4
-0.5	0.66	0	1	4
+0.5	0.66	0	1	4
-0.5	0.33	0	1	4
+0.5	0.33	0	1	4

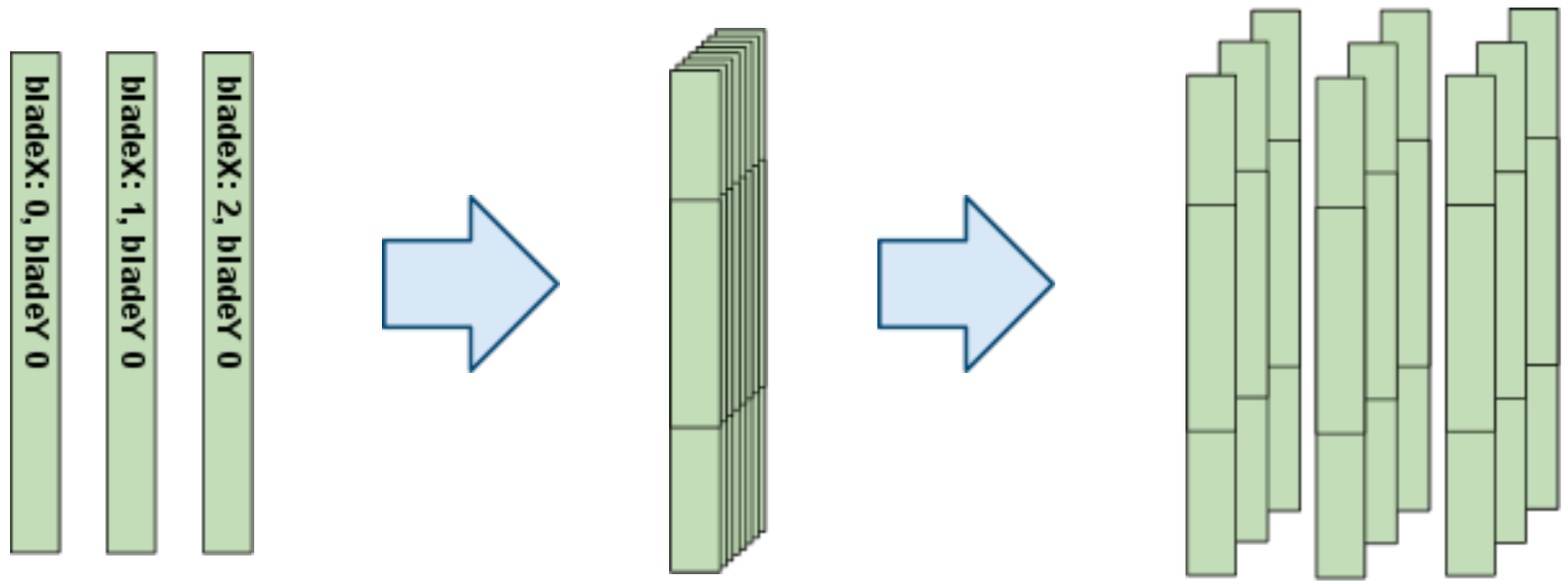
blade 1. 4

X	Y	Z	BladeX	BladeY
-0.5	1.0	0	2	4
+0.5	1.0	0	2	4
-0.5	0.66	0	2	4
+0.5	0.66	0	2	4
-0.5	0.33	0	2	4
+0.5	0.33	0	2	4

blade 2. 4

WebGL Field: Grass Blades

```
x = position.x + bladeId.x * bladeSpacing;
y = position.y + bladeId.y * bladeSpacing;
```



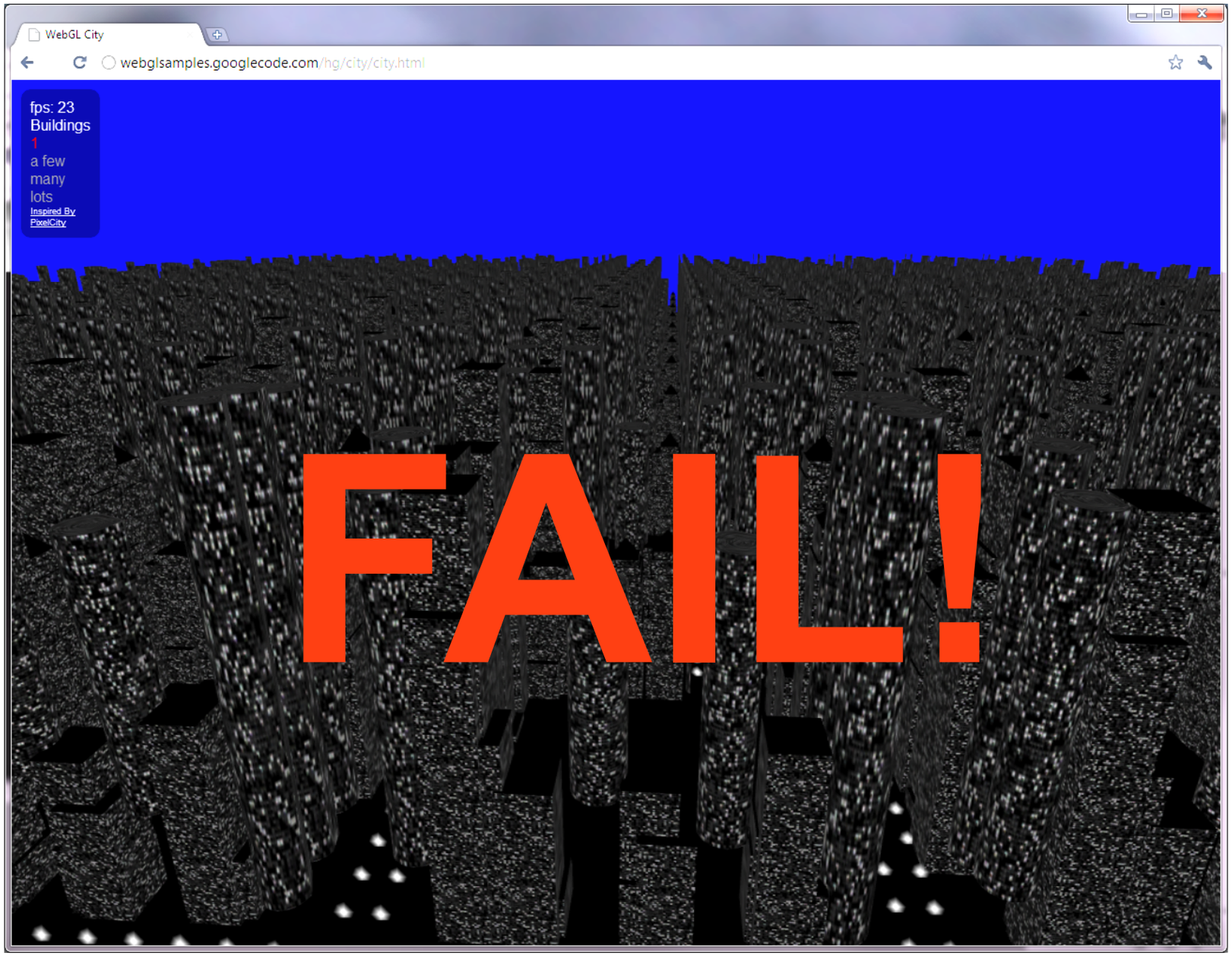
WebGL City

Inspired by Pixel City

http://www.youtube.com/watch?v=-d2-PtK4F6Y

http://flowingdata.com/2009/05/14/pixel-city-computer-generated-city/

WebGL City


WebGL City
webglsamples.googlecode.com/hg/city/city.html
fps: 23
Buildings
1
a few
many
lots
Inspired By
PixelCity
FAIL!

WebGL Libraries

ThreeDLibrary (TDL)

http://threedlibrary.googlecode.com

- A very low-level wrapper for WebGL
- Not an official library, just my own personal code
- math (from O3D)
- mesh primitives (cube, sphere, plane)
- generation of tangents and binormals
- easy async texture creation (from IMG, from CANVAS, from COLOR)
- Also used for Google Body

ThreeDLibrary (TDL) : Setup

```
var program = tdl.programs.loadProgram(
    vertexShaderSource, fragmentShaderSource);
```

ThreeDLibrary (TDL) : Setup

```
var program = tdl.programs.loadProgram(
    vertexShaderSource, fragmentShaderSource);

var textures = {
  diffuse: tdl.texture.loadTexture("myimg.jpg"),
  nmap: tdl.texture.loadTexture("normals.png")
};
```

ThreeDLibrary (TDL) : Setup

```
var program = tdl.programs.loadProgram(
    vertexShaderSource, fragmentShaderSource);

var textures = {
  diffuse: tdl.texture.loadTexture("myimg.jpg"),
  nmap: tdl.texture.loadTexture("normals.png")
};

var arrays = tdl.primitives.createSphere(
    1, 10, 10);
```

ThreeDLibrary (TDL) : Setup

```
var program = tdl.programs.loadProgram(
    vertexShaderSource, fragmentShaderSource);

var textures = {
  diffuse: tdl.texture.loadTexture("myimg.jpg"),
  nmap: tdl.texture.loadTexture("normals.png")
};

var arrays = tdl.primitives.createSphere(
    1, 10, 10);

var model = new tdl.models.Model(
    program, arrays, textures);
```

ThreeDLibrary (TDL) : Rendering

```
model.drawPrep({
    uniformThatIsTheSameForEachInstance1: value,
    uniformThatIsTheSameForEachInstance2: value
});
```

ThreeDLibrary (TDL) : Rendering

```
model.drawPrep({
    uniformThatIsTheSameForEachInstance1: value,
    uniformThatIsTheSameForEachInstance2: value
});

for (var i = 0; i < numToDraw; ++i) {
  ...
  model.draw({
    uniformThatIsDifferentForEachInstance1: value,
    uniformThatIsDifferentForEachInstance2: value
  });
}
```

X3DOM

HTML like inline declarative scene graph

```
<X3D ...>
    <Scene>
        <Viewpoint position='0 0 10' />
        <Shape>
            <Appearance>
                <Material diffuseColor='0.6 0.8 0.9' />
            </Appearance>
            <Box DEF='box'/>
        </Shape>
    </Scene>
</X3D>
```

X3DOM

- http://x3dom.org

SceneJS

- http://scenejs.org

  - Uses JSON for declarative scene graph
  - Inspired by JQuery

```
SceneJS.withNode("yaw").set("angle", yaw);
SceneJS.withNode("pitch").set("angle", pitch);

SceneJS.withNode("theScene").render();
```

Others

- GLGE http://www.glge.org/
- CopperLicht http://www.ambiera.com/copperlicht/
- SpiderGL http://spidergl.org/
- PhiloGL http://senchalabs.github.com/philogl/
- three.js https://github.com/mrdoob/three.js/

WebGL Tips

WebGL Tips: Check Features Exist

- WebGL's minimum required features.
  - 8 texture samplers
  - 0 vertex texture fetches!!! <===-------
  - 8 vertex attributes
  - 8 varyings
  - 128 uniforms
  - max draw buffer size
  - max texture size
  - max renderbuffer size
  - memory

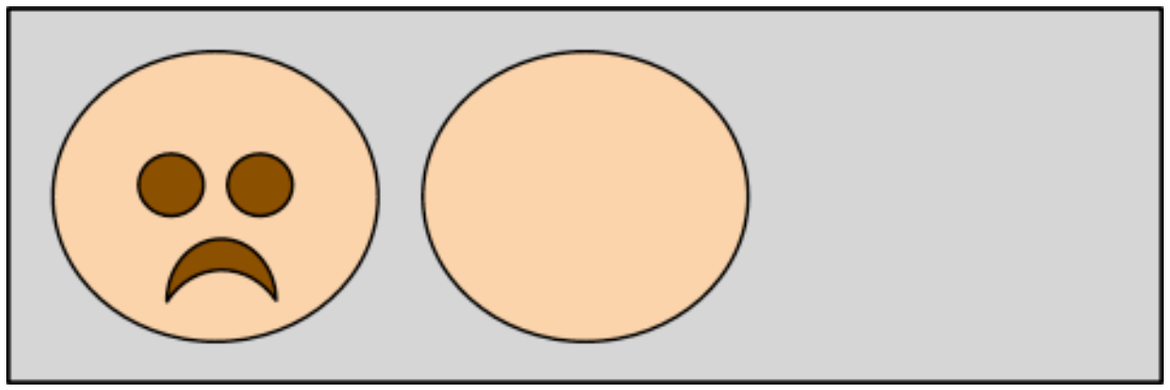
WebGL Tips: Batch up draw calls.

BAD

WebGL Tips: Batch up draw calls.

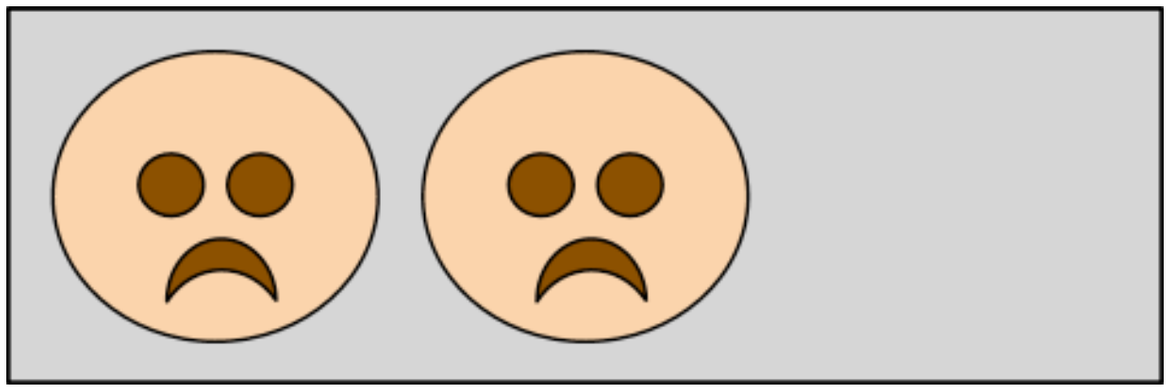
BAD

WebGL Tips: Batch up draw calls.



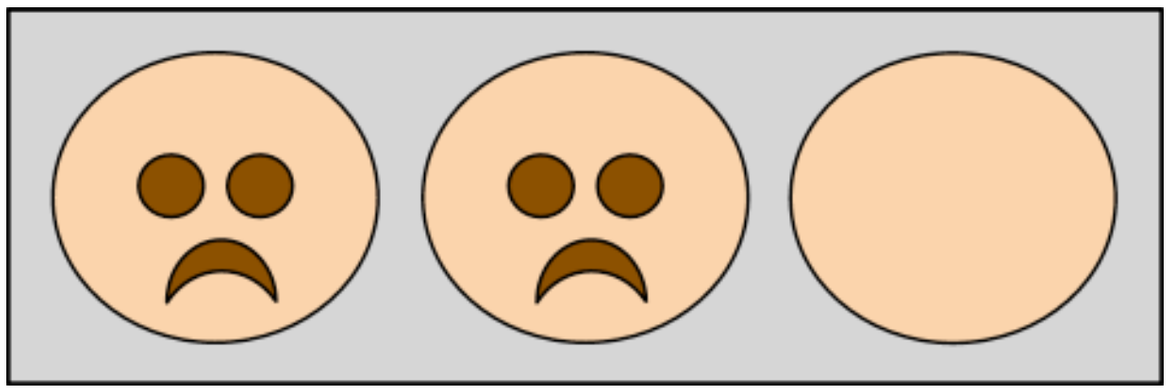
BAD

WebGL Tips: Batch up draw calls.



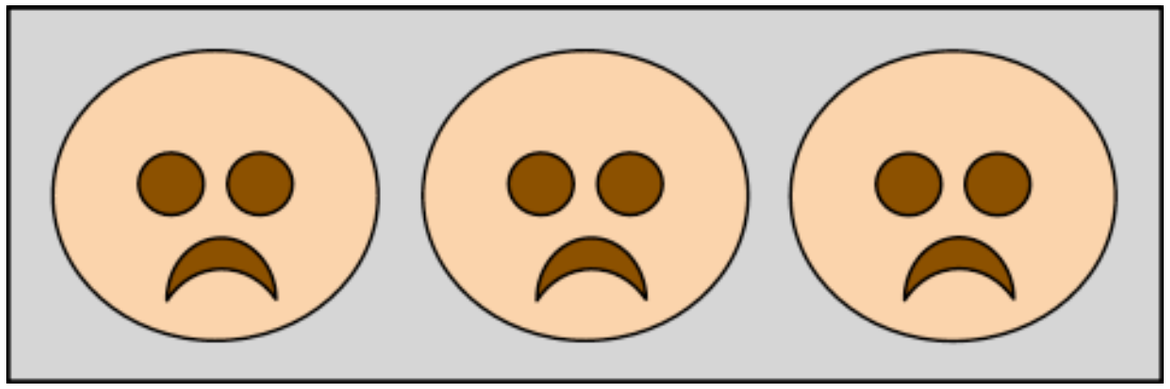
BAD

WebGL Tips: Batch up draw calls.



BAD

WebGL Tips: Batch up draw calls.

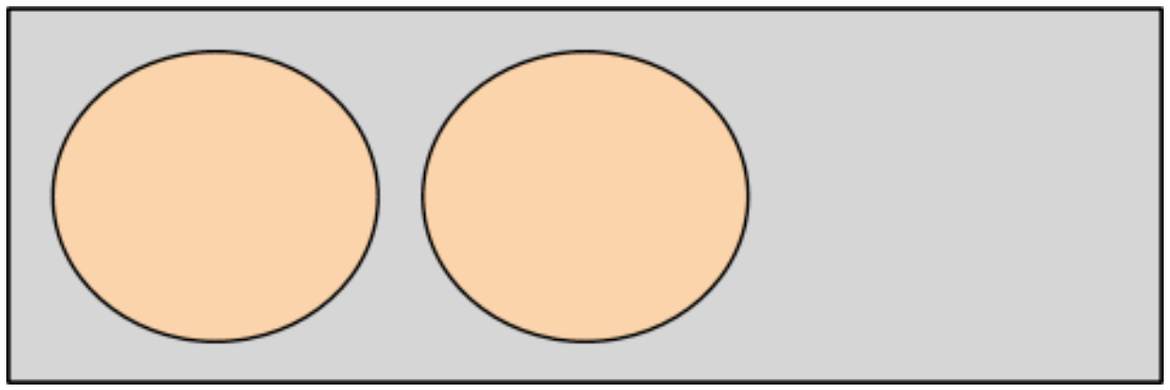


BAD

WebGL Tips: Batch up draw calls.

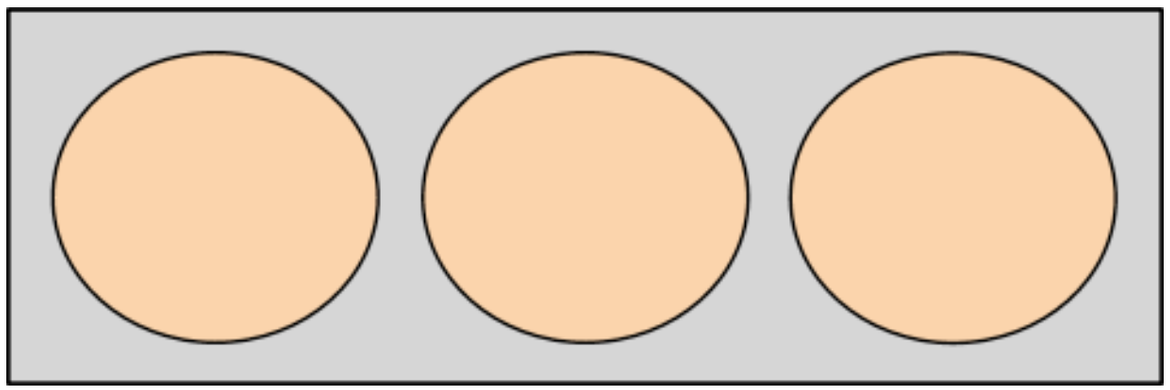
GOOD!

WebGL Tips: Batch up draw calls.



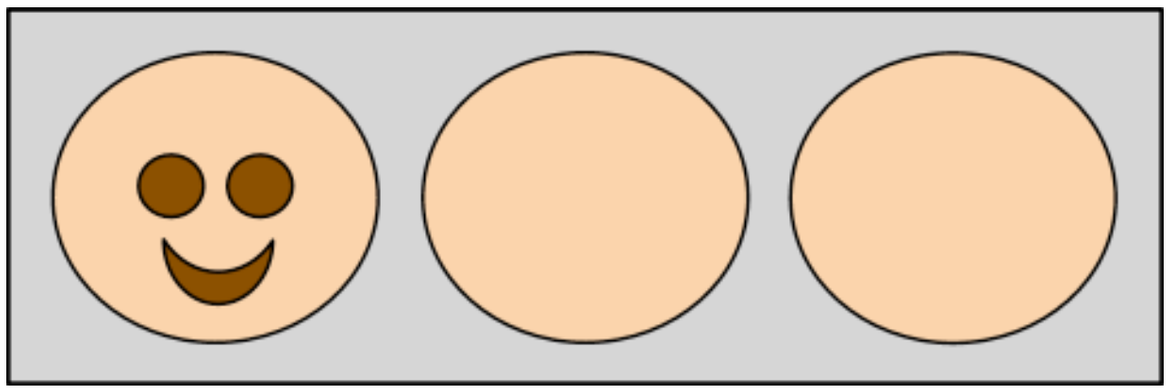
GOOD!

WebGL Tips: Batch up draw calls.



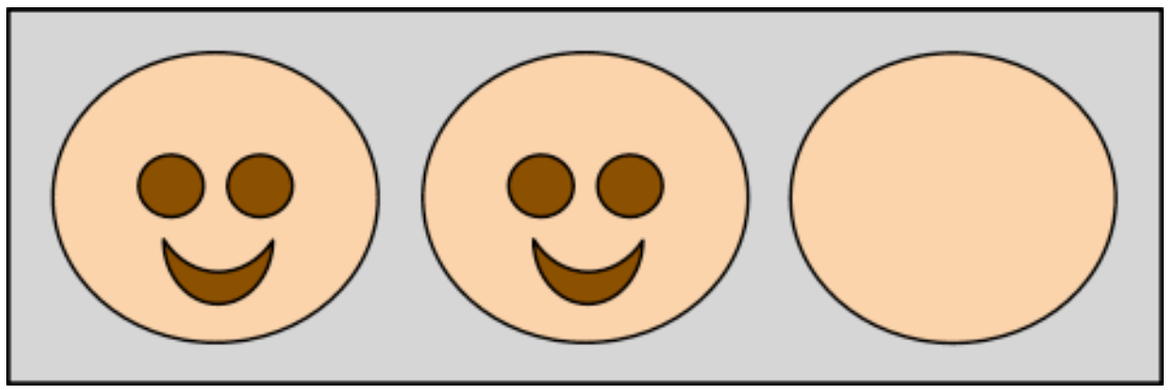
GOOD!

WebGL Tips: Batch up draw calls.



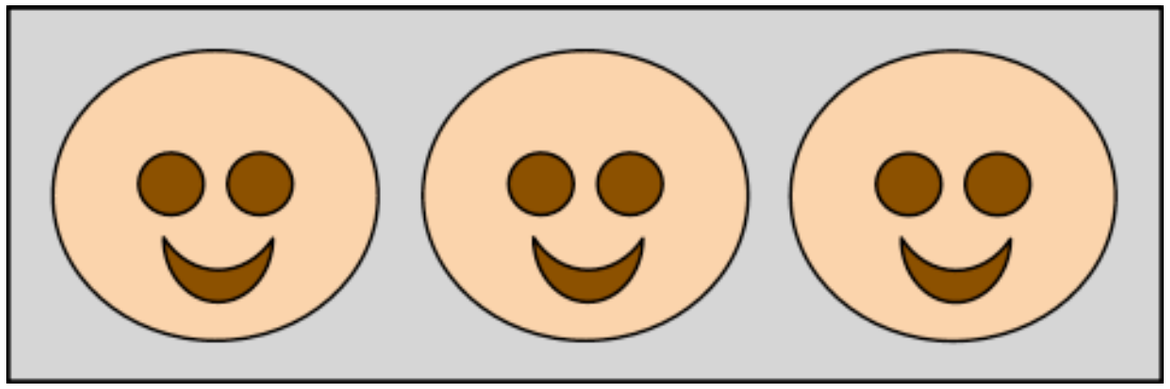
GOOD!

WebGL Tips: Batch up draw calls.



GOOD!

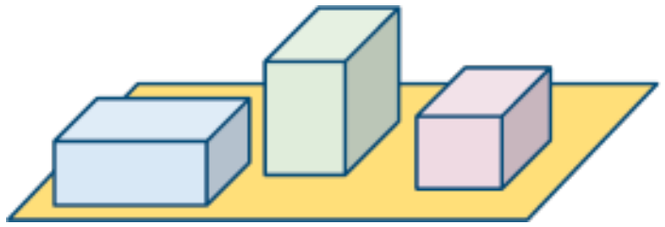
WebGL Tips: Batch up draw calls.

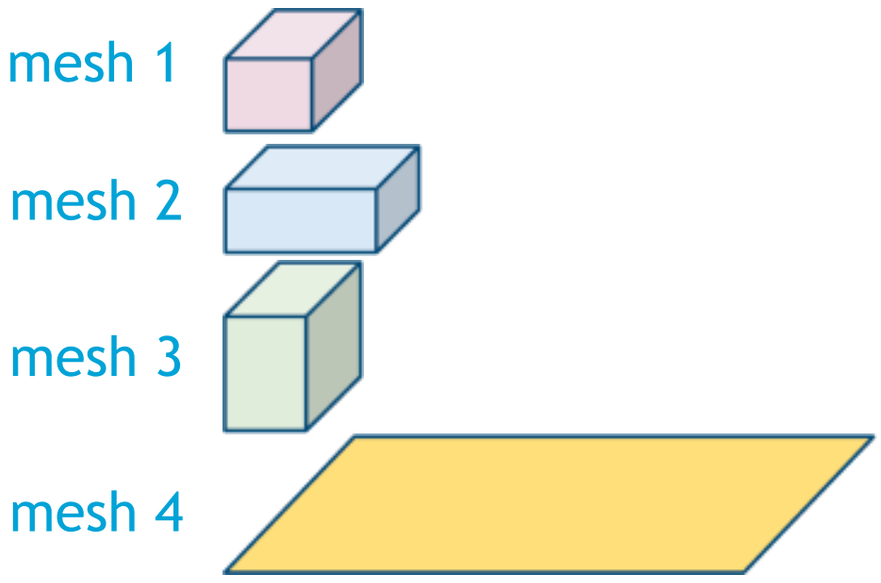


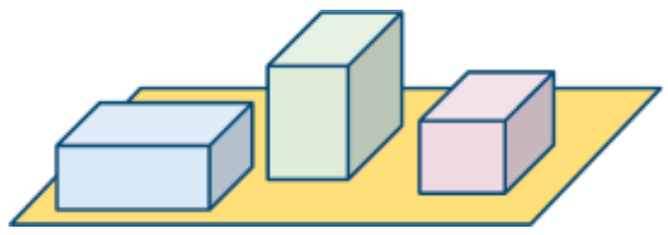
GOOD!

WebGL Tips: Bunch up geometry

Desired Scene







mesh 1

4 meshes = 4 draw calls (slow)

1 mesh = 1 draw call (fast)

WebGL Tips: XHR of TypeArrays

Loading Binary Data

```
var request = new XMLHttpRequest();
request.open('GET', url, true);
request.responseType = "arraybuffer";
request.onreadystatechange = myOnLoadFunc;
request.send(null);

...

function myOnLoadFunc()  {
    ...
    var arrayBuffer = request.response;
```

WebGL Tips: JavaScript Optimization

Learn JavaScript optimization

```
len = array.len;
for (i = 0; i < len; ++i)
```

is faster than

```
for(i = 0; i < array.len; ++i)
```

Also

```
myfunc()
```

is faster than

```
mylib.myobj.myfunc()
```

etc ....
In JavaScript, almost everything is a hash lookup except local vars.

WebGL Tips: Never gl.GetXXX!!!

Avoid gl.getXXX during rendering at all costs

- gl.getError
- gl.getUniformLocation
- gl.checkFramebufferStatus

don’t be a noob

WebGL Tips: Debug Context

Use a debug context:
(http://goo.gl/ZrfWY)

```
gl = canvas.getContext("webgl");
gl = WebGLDebugUtils.makeDebugContext(gl);
```

Calls gl.getError after every command.

WebGL Tips: WebGL Inspector

It's PIX for WebGL!

- capture a frame
- play back WebGL calls
- inspect buffers
- inspect textures

Shows the power of JavaScript!

WebGL Tips: Use HTML5 and Layers

Remember you are in a browser.

- Fast compositing of layers
- easy HUDs
- easy UIs
- easy texture generation with canvas
- easy image manipulation with canvas
- easy asynchronous downloading
- fast iteration
- easy to change code and data on the fly
- easy screen capture (cavnas.toDataURL)

WebGL MUSTS

do it or I will hunt you down

WebGL MUSTS: Check for WebGL

```
if (!window.WebGLRenderingContext) {
  // get a new browser!
  window.location = "http://get.webgl.org";
  return;
}
gl = canvas.getContext("webgl");
if (!gl) {
  // WebGL initialization failed
  window.location = "http://get.webgl.org/troubleshooting";
  return;
}
```

WebGLUtils: http://goo.gl/xcgir

MUST: use requestAnimationFrame

Do NOT use setInterval or setTimeout!!!

Use requestAnimationFrame

```
function render() {
    // draw stuff
    window.requestAnimationFrame(
        render, canvasElement);
}
render();
```

WebGLUtils Wrapper: http://goo.gl/xcgir

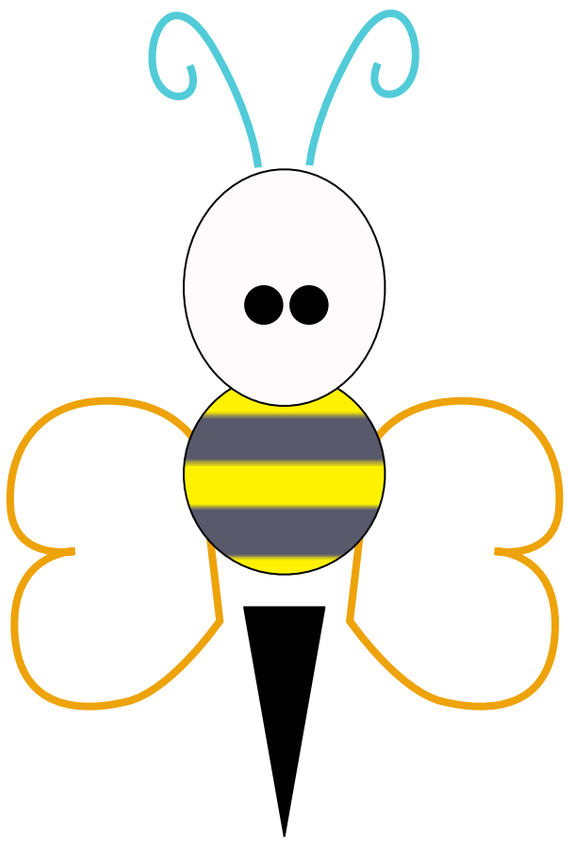
WebGL MUSTS: Stop Text Selection

Get rid of i-beam cursor and selection

```
function returnFalse() {
    return false;
}
canvas.onselectstart = returnFalse;
canvas.onmousedown = returnFalse;
```

WebGL: Reference

- "JavaScript: The Good Parts"
  by Douglas Crockford
- learningwebgl.com
- planet-webgl.org
- jquery.org
- jqueryui.com



©cREaTivE

WebGL

http://webglsamples.googlecode.com

http://webglsamples.googlecode.com/hg/gdc/2011

Gregg Tavares
gman@google.com
twitter: @greggman