WebGL
for OpenGL programmers

A hands-on approach

Morten Nobel-Jørgensen
DTU Compute

Interactive version:

http://mortennobel.github.io/WebGLForOpenGLProgrammers/

Agenda

http://ro.me

- Web programming 101
  - Html, CSS
  - JavaScript
- WebGL
  - Getting started
  - Security
- WebGL Libraries
  - Math
  - Debug
  - Game engines

Web programming 101

HTML, CSS and JavaScript

HTML: Hypertext Markup Language

- Markup language
- Easy to read and write
- HTML documents contains content and structure

```
<html>
  <body>
    <a href=”/home”>
      Link
    </a>
    Hello<br>World
  </body>
</html>
```

Link Hello

World

HTML: Hypertext Markup Language

- Tags:
  - Examples: <html>, <body>, <a>,
  - Start and end tag defines a scope e.g. <a>
  - Some tags don’t have end-tags e.g.

```
<html>
  <body>
    <a href=”/home”>
      Link
    </a>
    Hello<br>World
  </body>
</html>
```

Link Hello

World

HTML: Hypertext Markup Language

- Attributes
  - Add additional information to a tag
- Example:
  <img src=”a.png”>
  <a href=”a.html”>

```
<html>
  <body>
    <a href=”/home”>
      Link
    </a>
    Hello<br>World
  </body>
</html>
```

Link Hello

World

HTML: Hypertext Markup Language

- Text blocks
  - Usually contains visible text
- Other examples: style sheets or script code

```
<html>
  <body>
    <a href=”/home”>
      Link
    </a>
    Hello<br>World
  </body>
</html>
```

Link Hello

World

CSS: Stylesheet

- Changes visual appearance of html
- Styles can be applied at different levels
  - Document
  - Class
  - Element

```
<html>
  <style type="text/css">
    body {
     background-color:yellow;
    }
  </style>
  <body>
    <a href="/home"
          style="color:red;">
        Link
    </a>
  </body>
</html>
```

Link

JavaScript

- Makes websites dynamic
- Modifies web pages through the Document Object Model API (DOM API)

```
<html>
  <body>
  <script>
    var hw = 'Hello world';
    document.body.innerHTML = hw;
    </script>
  </body>
</html>
```

Hello world

JavaScript - C-like statements

- **If - statements**
  if-else statements

```
if (true){
  var hw = 'Hello world';
  document.write(hw);
} else {
  var b = 'Bye!';
  document.write(n);
}
```

Hello world

JavaScript - C-like statements

- If - statements
  if-else statements
- **for - loops**

```
for (var i=0;i<3;i++){
  var hw = 'Hello world '+i;
  document.write(hw);
}
```

Hello world 0Hello world 1Hello world 2

JavaScript - C-like statements

- If - statements
  if-else statements
- for - loops
- **while - loops**

```
var i=0;
while (i<3){
  var hw = 'Hello world '+i;
  document.write(hw);
  i++;
}
```

Hello world 0Hello world 1Hello world 2

JavaScript - C-like statements

- If - statements
  if-else statements
- for - loops
- while - loops
- **do-while loops**

```
var i=0;
do{
  var hw = 'Hello world '+i;
  document.write(hw);
  i++;
} while (i<3);
```

Hello world 0Hello world 1Hello world 2

JavaScript - Dynamic typing

- Types are associated with values not variables
- Variables can have values of different types

```
var i = "Hello world ";
document.write(i);
i = 0;
document.write(i);
```

Hello world 0

JavaScript - Types

- Boolean (true, false)
- Number (3.14)
  (Double-precision
  floating-point)
- String ('Test', "GL")
- Functions (function(){})
- Objects ({pi:3.14})
- Arrays ([3.14,'GL'])

JavaScript - Types

- Boolean (true, false)
- Number (3.14) (Double-precision floating-point)

Copy by value

- String ('Test', "GL")
- Functions (function(){})

Copy by reference
Immutable

- Objects ({pi:3.14})
- Arrays ([3.14,'GL'])

Copy by reference
Mutable

JavaScript - Functions

- **Has parameter list**
- **Optionally returns a value**

```
function f(a,b){
    return a+b;
}

var res = f(1,2);
document.write(res)
```

3

JavaScript - Functions

- Has parameter list
- Optionally returns a value
- **Note that functions are also objects**

```
function f(a,b){
    return a+b;
}

var res = f(1,2);
document.write(res)
```

```
var txt = f('a','b');
document.write(txt);
```

3ab

JavaScript - Functions

- Has parameter list
- Optionally returns a value
- Note that functions are also objects
- **Functions often used as callbacks**

```
var image = new Image();
image.onload = function(){
    console.log('loaded');
};
image.src = "img.png";
```

loaded

JavaScript - Objects

- Objects are sets of key-value pair
  - Keys are strings
  - Values can be any JS type
- Objects can reconfigured anytime
- Prototypes instead of classes

```
var o = new Object();
// var o = {};
o.x = 12;
document.write(o.x);
o.y = function(){
    document.write("y");
};
o.y();
```

12y

JavaScript - Arrays

- List of elements
- Can be indexed using [] operator
- Has a length property
- Can be dense or sparse
- Dynamically grows

```
var a = new Array();
// var a = [];
o[0] = 12;
document.write(o[0]);
o[1] = 'y';
document.write(o[1]);
document.write(o.length);
```

12y2

JavaScript - TypedArrays

- Create an array of a specific (numeric) type
  - Float32Array
  - Uint16Array
- Fixed length
- Needed for WebGL!

```
var a = new Uint16Array(2);
o[0] = -1;
document.write(o[0]);
o[1] = 3.14;
document.write('<br>');
document.write(o[1]);
```

65535
3

WebGL

WebGL - API

- Access WebGL functionality through an object
  - Functions and enums"
  - New objects replaces handles
- Naming as OpenGL API but without the gl-prefix
- OpenGL ES 2.0

OpenGL

```
glClearColor(1,1,1,1);
glClear(GL_COLOR_BUFFER_BIT);
```

WebGL

```
gl.clearColor(1,1,1,1);
gl.clear(gl.COLOR_BUFFER_BIT);
```

WebGL - Creating a context

```
<canvas id="n" width="500"
height="100"/>
<script>
var canvas =
  document.getElementById("n");
var gl = canvas.getContext("webgl");
gl.clearColor(1, 0, 0, 1);
gl.clear(gl.COLOR_BUFFER_BIT);
</script>
```

- Canvas element
- Create webgl-context using JavaScript

WebGL - interactive rendering

- Use setInterval to schedule callbacks
  - Alternative requestAnimationFrame
- Date.now() to get time in milliseconds

```
function display(){
  var timeSec = Date.now()/1000;
  // ...
}
setInterval(display, 16);
```

WebGL - textures

- Usually loaded from Image objects
- Only trusted data
  - Same host
  - Forbid local files
  - Cross-origin resource sharing (CORS)

```
var image = new Image();
image.onload = function() {
  var t= gl.createTexture();
  gl.activeTexture(
        gl.TEXTURE0);
  gl.bindTexture(
        gl.TEXTURE_2D, t);
  gl.texImage2D(
        gl.TEXTURE_2D, 0,
        gl.RGBA, gl.RGBA,
        gl.UNSIGNED_BYTE,
        image);
  // set min/mag filter
};
image.src = "img.png";
```

WebGL - textures embedded

- Binary image data can be encoded as text
- Base64 encoding
- This allows us to store images in the html document
- No security problems

```
var image = new Image();
image.onload = function() {
    ///
};
image.src = "[...]";
```

WebGL - Shader code

- GLSL ES Shaders as you know it
- Can be stored as
  - JavaString string
  - Text inside tags
  - External text documents

```
<script type="vertexShader"
id="vertexShader">
    attribute vec4 p;
    void main(void) {
        gl_Position = p;
    }
</script>
<script>
    var vertexShaderSrc =
document.getElementById( 'vert
exShader').textContent;
</script>
```

WebGL - Geometry data

- Usually stored raw in JavaScript
  - JavaScript data objects is called JSON
- Could parse binary or text model formats
  - OBJ, FBX, Collada

```
<script>
var modelData =
   {"vertex":[0,0,0],
    "normal":[0,1,0],
    "uv":[0,1],
    "indices":[0,1]};
</script>
```

WebGL libraries

gl-Matrix 2.0

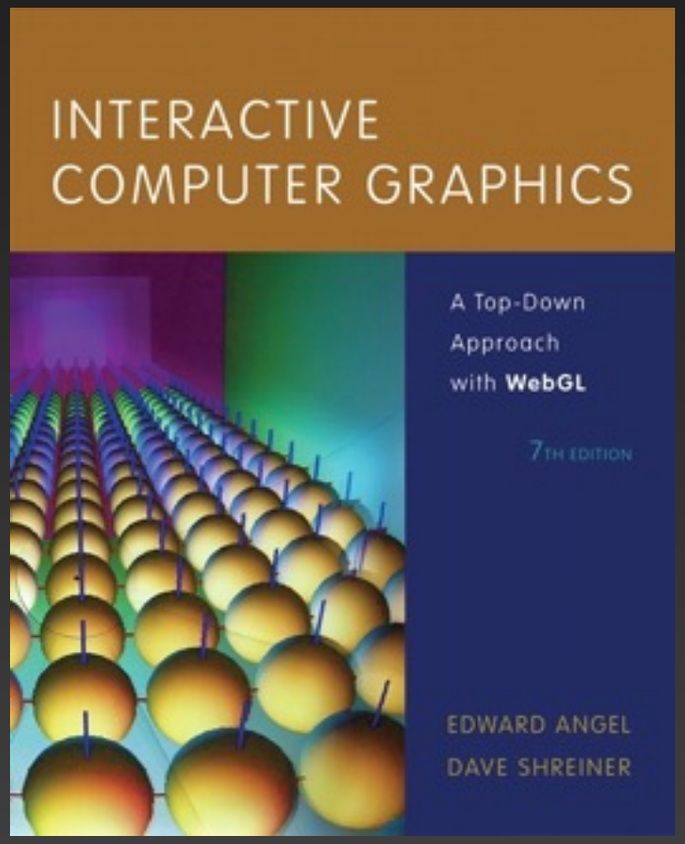
- Math library for WebGL
- A bit awkward to use:
  - Must specify both input and output parameters
- Documentation: http://glmatrix.net/docs/2.2.0/

```
<script>
var p = vec3.create();
var m = mat4.create();
mat4.identity(m);

var pr =
  mat4.perspective(m,
  3.14/2, 1, 0.1, 100);
</script>
```

Angel.js

- Simple Math library for WebGL
- From the book:




```
<script>
var projection =
  perspective(fovy,
  aspect, near, far);

var view =
  lookAt(vec3(0, 0, 3),
  vec3(0, 0, 0),
  vec3(0, 1, 0));

</script>
```

Debugging WebGL

- WebGLDebugContext checks for webgl errors after each gl call and write result to developer console
- Another useful tool is WebGL Inspector (a Chrome Extension)

```
<script src="https://
www.khronos.org/registry/
webgl/sdk/debug/webgl-
debug.js"></script>

<script>
var canvas =
 document.getElementById("n");
var gl =
canvas.getContext("webgl");
gl = WebGLDebugUtils.
    makeDebugContext(gl);
</script>
```

WebGL Abstractions

- Motivations:
  1. A lot of boilerplate code is needed to do simple things.

  2. WebGL is more low level than any other JavaScript API

- KickJS: A small game engine I have written
- Three.JS: Widely used WebGL engine

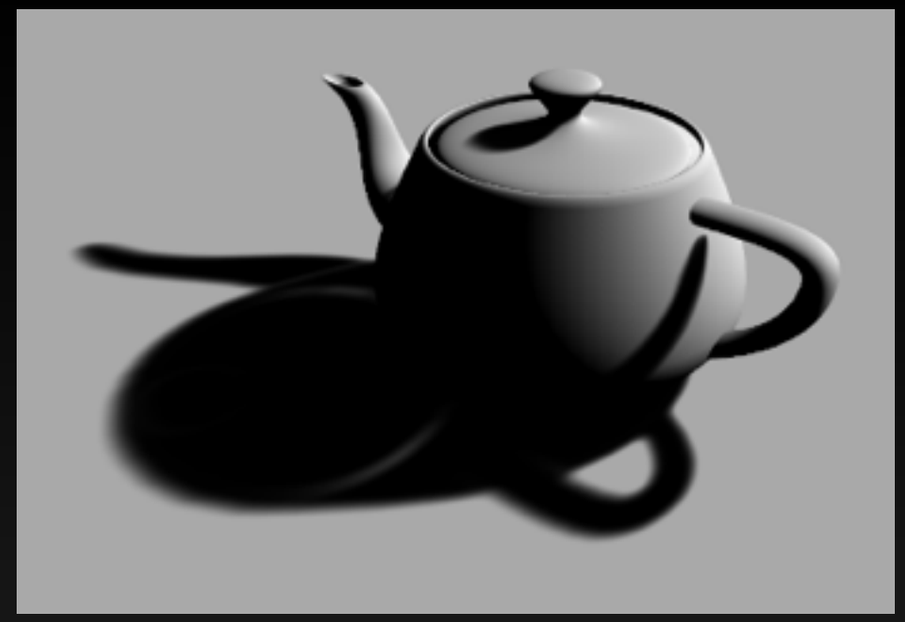
WebGL / OpenGL ES challenges and solutions

Lessons learned from creating WebGL

Picking

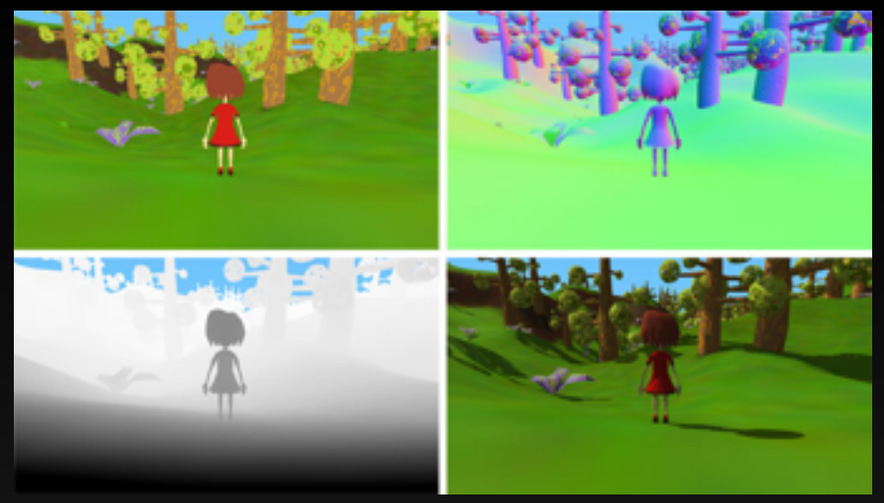
- Problem:
  - Want to find object in Windows space
- Solution:
  - Render to offscreen buffer using ids.
  - Copy pixels data back to read id.
  - Ids has to be packed into 4 bytes (using some “bit-shifting” tricks)

Shadow maps



- Problem: Cannot bind a depth buffer to a FBO
- Solution:
  - Use WebGL extension (on supported devices)
  - Or store depth value in color-buffer and do all the math yourself
  - (Problem: interpolation will not work correct)

Deferred rendering



http://www.neuroproductions.be/

- Problem:
  - Cannot render to multiple render targets (MRT)
- Solution:
  - Use WebGL extension (on supported devices)
  - Make multiple render passes

Useful references

- http://www.khronos.org/files/webgl/webgl-reference-card-1_0.pdf
- https://hacks.mozilla.org/2013/04/the-concepts-of-webgl/
- http://www.kickjs.org/tool/shader_editor/shader_editor.html
- http://www.kickjs.org
- http://threejs.org/