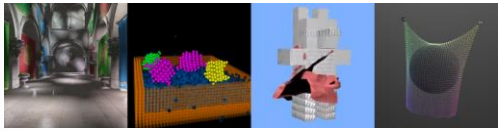


OpenGL and WebGL

Patrick Cozzi
University of Pennsylvania
CIS 565 - Fall 2016

OpenGL

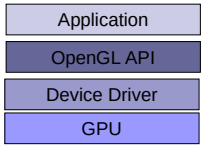
- Is a C-based API
- Is cross platform
- Is run by the *ARB*: Architecture Review Board
- Hides the device driver details
- OpenGL vs. Direct3D

OpenGL

- We are core profile
  - No fixed function vertex and fragment shading
  - No legacy API calls:
    - `glBegin()`
    - `glRotatef()`
    - `glTexEnvf()` ← Recall the fixed function light map
    - `AlphaFunc()` ← Why was the alpha test remove?
    - …

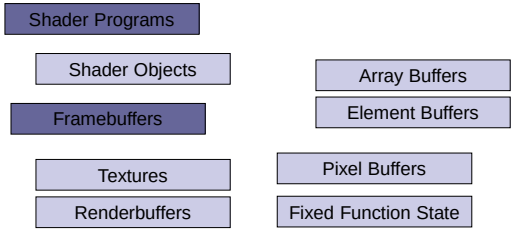
OpenGL

- Software stack:



OpenGL

- Major objects:



- We are not covering everything. Just surveying the most relevant parts for writing GLSL shaders

Shaders

- *Shader object*: an individual vertex, fragment, etc. shader
  - Are provided shader source code as a string
  - Are compiled
- Shader program: Multiple shader objects linked together

Shader Objects

- Compile a shader object:

```
const char *source = // ...
GLint sourceLength = // ...

GLuint v = glCreateShader(GL_VERTEX_SHADER);

glShaderSource(v, 1, &source, &sourceLength);

glCompileShader(v);

GLint compiled;
glGetShaderiv(v, GL_COMPILE_STATUS, &compiled);
// success:  compiled == GL_TRUE

// ...
glDeleteShader(v);
```

Shader Objects

- Compile a shader object:

```
const char *source = // ...
GLint sourceLength = // ...

GLuint v = glCreateShader(GL_VERTEX_SHADER);

glShaderSource(v, 1, &source, &sourceLength);

glCompileShader(v);

GLint compiled;
glGetShaderiv(v, GL_COMPILE_STATUS, &compiled);
// success:  compiled == GL_TRUE

// ...
glDeleteShader(v);
```

OpenGL functions start with `gl`. Why? How would you design this in C++?

`v` is an opaque object
- What is it under the hood?
- How would you design this in C++?

Shader Objects

- Compile a shader object:

```
const char *source = // ...
GLint sourceLength = // ...

GLuint v = glCreateShader(GL_VERTEX_SHADER);

glShaderSource(v, 1, &source, &sourceLength);

glCompileShader(v);

GLint compiled;
glGetShaderiv(v, GL_COMPILE_STATUS, &compiled);
// success:  compiled == GL_TRUE

// ...
glDeleteShader(v);
```

Provide the shader's source code

Where should the source come from?

Why can we pass more than one string?

Shader Objects

- Compile a shader object:

```
const char *source = // ...
GLint sourceLength = // ...

GLuint v = glCreateShader(GL_VERTEX_SHADER);

glShaderSource(v, 1, &source, &sourceLength);

glCompileShader(v);

GLint compiled;
glGetShaderiv(v, GL_COMPILE_STATUS, &compiled);
// success:  compiled == GL_TRUE

// ...
glDeleteShader(v);
```

Compile, but what does the driver really do?

Shader Objects

- Compile a shader object:

```
const char *source = // ...
GLint sourceLength = // ...

GLuint v = glCreateShader(GL_VERTEX_SHADER);

glShaderSource(v, 1, &source, &sourceLength);

glCompileShader(v);

GLint compiled;
glGetShaderiv(v, GL_COMPILE_STATUS, &compiled);
// success:  compiled == GL_TRUE

// ...
glDeleteShader(v);
```

Good developers check for error. Again, how would you design this in C++?

Calling `glGet*` has performance implications. Why?

Shader Objects

- Compile a shader object:

```
const char *source = // ...
GLint sourceLength = // ...

GLuint v = glCreateShader(GL_VERTEX_SHADER);

glShaderSource(v, 1, &source, &sourceLength);

glCompileShader(v);

GLint compiled;
glGetShaderiv(v, GL_COMPILE_STATUS, &compiled);
// success:  compiled == GL_TRUE

// ...
glDeleteShader(v);
```

Good developers also cleanup resources

Shader Programs

- Link a shader program:

```
GLuint v = glCreateShader(GL_VERTEX_SHADER);
GLuint f = glCreateShader(GL_FRAGMENT_SHADER);
// ...

GLuint p = glCreateProgram();
glAttachShader(p, v);
glAttachShader(p, f);

glLinkProgram(p);

GLint linked;
glGetShaderiv(p, GL_LINK_STATUS, &linked);
// success:  linked == GL_TRUE

// ...
glDeleteProgram(v);
```

Shader Programs

- Link a shader program:

```
GLuint v = glCreateShader(GL_VERTEX_SHADER);
GLuint f = glCreateShader(GL_FRAGMENT_SHADER);
// ...

GLuint p = glCreateProgram();
glAttachShader(p, v);
glAttachShader(p, f);

glLinkProgram(p);

GLint linked;
glGetShaderiv(p, GL_LINK_STATUS, &linked);
// success:  linked == GL_TRUE

// ...
glDeleteProgram(v);
```

A program needs a vertex and fragment shader

Shader Programs

- Link a shader program:

```
GLuint v = glCreateShader(GL_VERTEX_SHADER);
GLuint f = glCreateShader(GL_FRAGMENT_SHADER);
// ...

GLuint p = glCreateProgram();
glAttachShader(p, v);
glAttachShader(p, f);

glLinkProgram(p);

GLint linked;
glGetShaderiv(p, GL_LINK_STATUS, &linked);
// success:  linked == GL_TRUE

// ...
glDeleteProgram(v);
```

Shader Programs

- Link a shader program:

```
GLuint v = glCreateShader(GL_VERTEX_SHADER);
GLuint f = glCreateShader(GL_FRAGMENT_SHADER);
// ...

GLuint p = glCreateProgram();
glAttachShader(p, v);
glAttachShader(p, f);

glLinkProgram(p);

GLint linked;
glGetShaderiv(p, GL_LINK_STATUS, &linked);
// success:  linked == GL_TRUE

// ...
glDeleteProgram(v);
```

Be a good developer again

Using Shader Programs

```
GLuint p = glCreateProgram();
// ...

glUseProgram(p);
glDraw*(); // * because there are lots of draw functions
```

Part of the current state
- How do you draw different objects with different shaders?
- What is the cost of using multiple shaders?
- How do we reduce the cost?
  - Hint: write more CPU code – really.

Uniforms

```
GLuint p = glCreateProgram();
// ...
glLinkProgram(p);

GLuint m = glGetUniformLocation(p, "u_modelViewMatrix");
GLuint l = glGetUniformLocation(p, "u_lightMap");

glUseProgram(p);
mat4 matrix = // ...
glUniformMatrix4fv(m, 1, GL_FALSE, &matrix[0][0]);
glUniform1i(l, 0);
```

Uniforms

Each *active* uniform has an integer index location.

```
GLuint p = glCreateProgram();
// ...
glLinkProgram(p);

GLuint m = glGetUniformLocation(p, "u_modelViewMatrix");
GLuint l = glGetUniformLocation(p, "u_lightMap");

glUseProgram(p);
mat4 matrix = // ...
glUniformMatrix4fv(m, 1, GL_FALSE, &matrix[0][0]);
glUniform1i(l, 0);
```

Uniforms

```
GLuint p = glCreateProgram();
// ...
glLinkProgram(p);

GLuint m = glGetUniformLocation(p, "u_modelViewMatrix");
GLuint l = glGetUniformLocation(p, "u_lightMap");

glUseProgram(p);
mat4 matrix = // ...
glUniformMatrix4fv(m, 1, GL_FALSE, &matrix[0][0]);
glUniform1i(l, 0);
```

`mat4` is part of the C++ GLM library

GLM: http://www.g-truc.net/project-0016.html#menu

Uniforms

```
GLuint p = glCreateProgram();
// ...
glLinkProgram(p);

GLuint m = glGetUniformLocation(p, "u_modelViewMatrix");
GLuint l = glGetUniformLocation(p, "u_lightMap");

glUseProgram(p);
mat4 matrix = // ...
glUniformMatrix4fv(m, 1, GL_FALSE, &matrix[0][0]);
glUniform1i(l, 0);
```

`glUniform*` for all sorts of datatypes

Uniforms can be changed as often as needed, but are constant during a draw call

Not transposing the matrix

Uniforms

```
GLuint p = glCreateProgram();
// ...
glLinkProgram(p);

GLuint m = glGetUniformLocation(p, "u_modelViewMatrix");
GLuint l = glGetUniformLocation(p, "u_lightMap");

glUseProgram(p);
mat4 matrix = // ...
glUniformMatrix4fv(m, 1, GL_FALSE, &matrix[0][0]);
glUniform1i(l, 0);
```

Why not `glUniform*(p, …)`?

WebGL

- The web has text, images, and video
  - What is the next media-type?
- We want to support
  - Windows, Linux, Mac
  - Desktop and mobile

Bring 3D to the Masses

- Put it in on a webpage
  - Does not require a plugin or install
  - Does not require administrator rights
- Make it run on most GPUs

WebGL

- OpenGL ES 2.0 for JavaScript
  - Seriously, JavaScript

Image from http://www.khronos.org/assets/uploads/developers/library/2011-siggraph-mobile/Khronos-and-the-Mobile-

WebGL

- Includes
  - Vertex shaders
  - Fragment shaders
  - Vertex buffers
  - Textures
  - Framebuffers
  - Render states
  - …
- Does not include
  - Geometry shaders
  - Tessellation shaders
  - Vertex Array Objects
  - Multiple render targets
  - Floating-point textures
  - Compressed textures
  - FS depth writes
  - …

See http://www.khronos.org/registry/webgl/specs/latest/

WebGL

- If you know *OpenGL*, you already know *WebGL*
- If you know *C++*, the real learning curve is *JavaScript*

WebGL

- Creating a context is easy:

```
// HTML:
<canvas id="glCanvas" width="1024"
  height="768"></canvas>

// JavaScript:
var gl =
  document.getElementById("glCanvas")
  .getContext("experimental-webgl");
```

WebGL

- The rest is similar to desktop OpenGL:

```
// ...
gl.bindBuffer(/* ... */);
gl.vertexAttribPointer(/* ... */);
gl.useProgram(/* ... */);
gl.drawArrays(/* ... */);
```

Checkout http://learningwebgl.com/

WebGL

- Create an animation loop:

```
(function tick(){
  // ... GL calls to draw scene
  window.requestAnimationFrame(tick);
})();
```

You want this to work cross-browser. See http://paulirish.com/2011/requestanimationframe-for-smart-animating/

WebGL Performance

- Performance can be very good. Why?

WebGL Performance

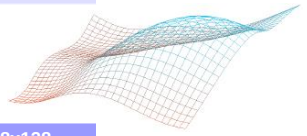
- Performance can be very good. Why?
  - The GPU is still doing the rendering
  - Batch!
    - Draw multiple objects with one draw call
    - Sort by texture
    - Push work into shaders
    - Push work into web workers

See http://www.youtube.com/watch?v=rfQ8rKGTVlg

WebGL Performance (out dated)

	32x32	64x64	128x128
C++	1.9 ms	6.25 ms	58.82 ms
Chrome 18	27.77 ms	111.11 ms	454.54 ms
x slowdown	14.62	17.78	7.73

CPU-intensive



	32x32	64x64	128x128
C++	3.33 ms	9.43 ms	37.03 ms
Chrome 18	12.82 ms	22.72 ms	41.66 ms
x slowdown	3.85	2.41	1.13

GPU-intensive (256 draws per frame)

33

WebGL and other APIs

- Take advantage of other web APIs:
  - HTML5 <video>
  - 2D <canvas>
  - CSS transforms
  - Composite UI elements
  - Web workers
  - Typed Arrays

34

HTML5 on Mobile

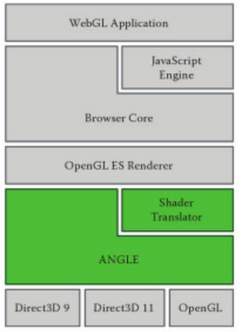
- Touch events
- Geolocation
- Device orientation and motion

- The future of HTML5 and WebGL on mobile is *very promising*

35

ANGLE

- *ANGLE* – *A*lmost *N*ative *G*raphics *L*ayer *E*ngine



36

Image from WebGL Insights

Tools Demos

- WebGL Report
- Chrome debugger
- Chrome profiler
- Firefox shader editor
- Firefox canvas inspector
- Web Tracing Framework

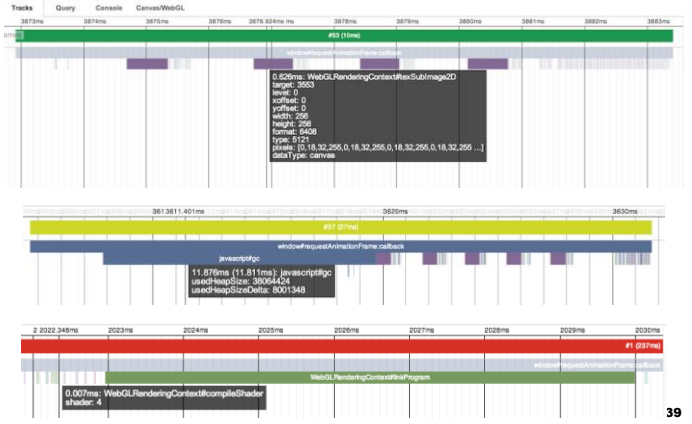
37

Performance Bottlenecks

- Garbage collector (browser CPU overhead)
- Shader compile and link (driver CPU overhead): compileShader, linkProgram, getProgramParameter, and friends. When is the performance hit?
- Texture/buffer upload (driver CPU overhead): texImage2D, texSubImage2D, bufferData, bufferSubData, and friends
- readPixels (stall CPU and starve GPU)
- getParameter and other get* functions (stall CPU for inter-process communication)
- drawElements/drawArrays - lack of view frustum culling and batching, i.e., doing a lot of calls to draw meshes that are not visible
- uniform* - lack of batching

38

Performance Bottleneck



39

*C*ross-*O*rigin *R*esource *S*haring

- Images can't always be used as texture sources. Why?

40

Cross-Origin Resource Sharing

- Same domain is OK:

```
var img = new Image();
img.onload = function() {
  gl.texImage2D(/* ... */, img);
};
img.src = "image.png";
```

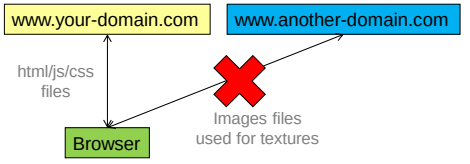

Cross-Origin Resource Sharing

- Another domain requires *CORS* if supported:

```
var img = new Image();
img.onload = function() {
  gl.texImage2D(/* ... */, img);
};
img.crossOrigin = "anonymous";
img.src =
"http://another-domain.com/image.png";
```

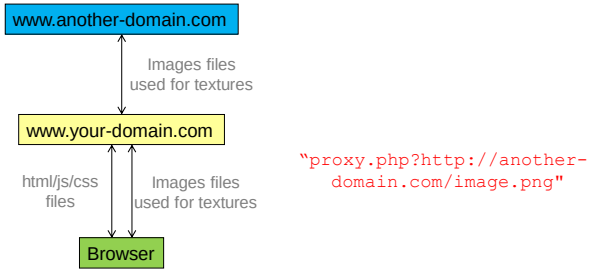

Cross-Origin Resource Sharing

- Not all servers support CORS:



Cross-Origin Resource Sharing

- Use a proxy server:



See http://resources.esri.com/help/9.3/arcgisserver/apis/javascript/arcgis/help/jshelp/ags_proxy.htm

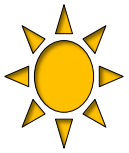
Denial of Service Attacks

- Long draw calls
  - Complicated shaders
  - Big vertex buffers
- Solutions
  - Kill long draw calls
  - Forbid further rendering

Lots of WebGL security info: http://learningwebgl.com/blog/?p=3890

WebGL Libraries

- Three.js: https://github.com/mrdoob/three.js/
- Cesium: http://cesium.agi.com/
- Many more: http://www.khronos.org/webgl/wiki/User_Contributions



The Joys of JavaScript

Skip the next 30 slides if you already know JavaScript

JavaScript is weakly typed…

JavaScript Type System

- `short`, `int`, `float`, `double`. Who needs them?

```
var n = 1;
```


JavaScript Type System

- JavaScript has numbers, strings, and booleans:

```
var n = 1;
var s = "WebGL";
var b = true;
```


JavaScript Type System

- This compiles:

```
var n = 1;
var s = "WebGL";
var b = true;

var sum = n + s + b;
```


JavaScript is a functional language…

JavaScript Functions

- Looks familiar:

```
function add(x, y) {
  return x + y;
}

var sum = add(1, 2);
```

- Functions are first-class objects, so…

JavaScript Functions

- Functions are objects:

```
var add = function(x, y) {
 return x + y;
};

var sum = add(1, 2);
```


JavaScript Functions

- Pass functions to functions:

```
var add = function // ...

function execute(op, x, y) {
  return op(x, y);
}

var sum = execute(add, 1, 2);
```


JavaScript Anonymous Functions

- Why name functions?

```
function execute(op, x, y) // ...

var sum = execute(function(x, y) {
 return x + y;
}, 1, 2);
```

JavaScript Closures

- Why limit scope?

```
var z = 3;

var sum = execute(function(x, y) {
 return x + y + z;
}, 1, 2);
```

57

JavaScript is a dynamic language…

58

JavaScript Object Literals

- Who needs `struct`? Create objects on the fly:

```
var position = {
  x : 1.0,
  y : 2.0
};
```

59

JavaScript Object Literals

- Why not add fields on the fly too?

```
var position = {
  x : 1.0,
  y : 2.0
};
position.z = 3.0;
```

60

JavaScript Object Literals

- Who needs `class`?

61

JavaScript Object Literals

- Who needs `class`? Create functions too:

```
var position = {
  x : 1.0,
  y : 2.0,
  min : function() {
    return Math.min(this.x, this.y);
  }
};
```

62

JavaScript Object Literals

- Why not change `min()`?

```
position.z = 3.0;
position.min = function() {
  return Math.min(this.x, this.y,
    this.z);
};
```

63

JavaScript Object Literals

- Useful for passing to functions. Why?

64

JavaScript Object Literals

- Useful for passing to functions. Why?
- What do these arguments mean?

```
pick(322, 40, 5, 4);
```


JavaScript Object Literals

- Useful for passing to functions. Why?
- What do these arguments mean?

```
pick({
  x : 322,
  y : 40,
  width : 5,
  height : 4
});
```


JavaScript does object-oriented…

JavaScript Constructor Functions

```
function Vector(x, y) {
  this.x = x;
  this.y = y;
}

var v = new Vector(1, 2);
```

JavaScript Constructor Functions

- Objects can have functions:

```
function Vector(x, y) {
  this.x = x;
  this.y = y;
  this.min = function() {
    return Math.min(this.x, this.y);
  };
}
```


JavaScript Constructor Functions

- Objects have prototypes:

```
function Vector(x, y) {
  this.x = x;
  this.y = y;
}

Vector.prototype.min = function() {
  return Math.min(this.x, this.y);
};
```


JavaScript Polymorphism

- No need for virtual functions

```
function draw(model) {
  model.setRenderState();
  model.render();
}
```


JavaScript Polymorphism

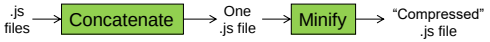
- No need for virtual functions

```
var level = {
  setRenderState : function() // ...
  render : function() // ...
};

draw(level); // Just works
```

JavaScript Build Pipeline

- Different than C++
- *Goal*: fast downloads
- Common:



- Alternative: fine-grain modules
- How do you deploy shaders?

See http://www.julienlecomte.net/blog/2007/09/16/

JavaScript Advice

- Use JSHint
- Have excellent test coverage
- Use the Chrome and Firefox debuggers