Computer Graphics

David Bremner

Contents

1 WebGL

1.1 History

OpenGL

1990s Developed by Silicon Graphics, standard in "high end" graphics

2004 OpenGL 2.0 introduces C-like shader language GLSL

2007 OpenGL ES 2.0 (embedded systems) removes "fixed-function rendering", replaces with shaders

2010 WebGL (roughly) equivalent to OpenGL ES 2.0, callable from JavaScipt.

Portable by omission

OpenGL is device/window-system independent, so some "glue layer" is needed.

- (Free)GLUT for OpenGL on X11, Win32
- EGL for OpenGL ES.
- Canvas for WebGL

1.2 Coordinate Systems

Screen Coordinates

Figure 2_16 in ML2013

Right-handed coordinate systems

Figure 2_17 in ML2013

Canvas coordinates

Figure 2_18 in ML2013

1.3 Hello World

HTML Loader

```
<body onload="main()">
  <canvas id="webgl" width="400" height="400">
  Please use a browser that supports "canvas"
  </canvas>

  <script src="../lib/webgl-utils.js"></script>
  <script src="../lib/webgl-debug.js"></script>
  <script src="../lib/cuon-utils.js"></script>
  <script src="HelloCanvas.js"></script>
</body>
```

- Example code and convenience libraries used in the text can be downloaded from https://sites.google.com/site/webglbook/
- To open the JavaScript console in Chrome, use Shift-Control-J

- In Firefox use Shift-Control-K

ML:2.2

ch02/HelloCanvas [source]

```
function main() {
  // Retrieve <canvas> element
  var canvas = document.getElementById('webgl');

  // Get the rendering context for WebGL
  var gl = getWebGLContext(canvas);
  if (!gl) {
    console.log('Failed to get context for WebGL');
    return;
  }

  // Set clear color
  gl.clearColor(0.0, 0.0, 0.0, 1.0);

  // Clear <canvas>
  gl.Clear(gl.COLOR_BUFFER_BIT);
}
```

1.4 Hello Shaders

ML:2.3

WebGL rendering

Figure 2_10 in ML2013

Vertex Shader

```
void main() {
  gl_Position = vec4(0.0, 0.0, 0.0, 1.0);// (x,y,z,w)
  gl_Pointsize = 10.0;      // point size
}
```

Fragment Shader

```
void main() {
  gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0); //(r,g,b,α)
}
```

Using shaders I: Code in strings

```
var VSHADER_SOURCE =
  'void main() {\n' +
  '  gl_Position = vec4(0.0, 0.0, 0.0, 1.0);\n' +
  '  gl_PointSize = 10.0;\n' +
  '}\n';

var FSHADER_SOURCE =
  'void main() {\n' +
  '  gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);\n' +
  '}\n';
```

- Embed as scripts in html files
- Use JS to read from files (on remote server)

Using Shaders II: initialize

ch02/HelloPoint1 [source]

```
function main() {
  ⋮
  // Initialize shaders
  if (!initShaders(gl, VSHADER_SOURCE,
                           FSHADER_SOURCE)) {
    console.log('Failed to intialize shaders.');
    return;
  }
  ⋮
  // Draw a point
  gl.drawArrays(gl.POINTS, 0, 1);
}
```

Figure 2_14 in ML2013

1.5 Attribute Variables

ML:2.4

Controlling the pipeline with variables

Figure 2_20 in ML2013

Input to shaders

ch02/HelloPoint2 [source]

```
var VSHADER_SOURCE =
  'attribute vec4 a_Position;\n' + // attribute var
⋮
function main() {
  ⋮
  // Get the storage location of a_Position
  var a_Position = gl.getAttribLocation(gl.program, 'a_Position');
  if (a_Position < 0) {
    console.log('Failed to get storage location');
    return;
  }
  // Pass vertex position to attribute variable
  gl.vertexAttrib3f(a_Position, 0.0, 0.0, 0.0);
  ⋮
}
```

Communicating with the vertex shader

Figure 2_22 in ML2013

1.6 Event handlers

Register event handler

ch02/ClickedPoints [source]

```
function main() {
⋮
  // function to be called on a mouse press
  canvas.onmousedown =
    function(ev){
      click(ev, gl, canvas, a_Position);
    };

  gl.clearColor(0.0, 0.0, 0.0, 1.0);
  gl.clear(gl.COLOR_BUFFER_BIT);
}
```

Canvas and Browser Coordinates ML:2.5

Figure 2_26 in ML2013

Canvas and WebGL Coordinates

Figure 2_27 in ML2013

Handling events I: convert coordinates

ch02/ClickedPoints [source]

```
// The array for the position of a mouse press
var g_points = [];
function click(ev, gl, canvas, a_Position) {
  var x = ev.clientX; // pointer x
  var y = ev.clientY; // pointer y
  var rect = ev.target.getBoundingClientRect() ;

  // convert to WebGL coordinates
  x = ((x - rect.left) - canvas.width/2)/(canvas.width/2);
  y = (canvas.height/2 - (y - rect.top))/(canvas.height/2);

  // Store the coordinates to g_points array
  g_points.push(x); g_points.push(y);
⋮
}
```

Handling events II: drawing

ch02/ClickedPoints [source]

```
function click(ev, gl, canvas, a_Position) {
  ⋮
  // Store the coordinates to g_points array
  g_points.push(x); g_points.push(y);

  gl.clear(gl.COLOR_BUFFER_BIT);
  var len = g_points.length;
  for(var i = 0; i < len; i += 2) {
    // Pass the position to shader
    gl.vertexAttrib3f(a_Position, g_points[i], g_points[i+1],
```

```
                                        0.0);
    // Draw
    gl.drawArrays(gl.POINTS, 0, 1);
  }
}
```

1.7 Uniform Variables

Uniform versus attribute variables ML:2.6

Figure 2_30 in ML2013

See also varying

Using Uniform Variables

ch02/ColoredPoints [source]

```
// Fragment shader program
var FSHADER_SOURCE =
  'precision mediump float;\n' +
  'uniform vec4 u_FragColor;\n' +  // uniform
  'void main() {\n' +
  '  gl_FragColor = u_FragColor;\n' +
  '}\n';n
⋮
  // Pass the position to shader
  gl.vertexAttrib3f(a_Position, xy[0], xy[1], 0.0);
  // Pass the color to shader
  gl.uniform4f(u_FragColor,rgba[0],rgba[1],rgba[2], rgba[3]);
  // Draw
  gl.drawArrays(gl.POINTS, 0, 1);
⋮
```

1.8 Buffers and drawing multiple points

WebGL Buffers ML:3.1

Figure 3_5 in ML2013

Drawing multiple points with gl.DrawArrays

ch03/MultiPoint [source]

```
function main() {

⋮
  // Write vertex positions to shader
  var n = initVertexBuffers(gl);
  if (n < 0) {
    console.log('Failed to set the positions of the vertices');
    return;
  }

⋮
  // Draw three points
  gl.drawArrays(gl.POINTS, 0, n);
}
```

Buffers I: creating

ch03/MultiPoint [source]

```
function initVertexBuffers(gl) {
  var vertices = new Float32Array(
    [ 0.0, 0.5, -0.5, -0.5, 0.5, -0.5 ]);
  var n = 3; // The number of vertices

  // Create a buffer object (1)
  var vertexBuffer = gl.createBuffer();
  if (!vertexBuffer) { return -1; }

  // Bind the buffer object to target (2)
  gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuffer);
  // Write date into the buffer object (3)
  gl.bufferData(gl.ARRAY_BUFFER, vertices,
        gl.STATIC_DRAW);
⋮
```

Buffers II: using
ch03/MultiPoint [source]

```
function initVertexBuffers(gl) {
:
  var a_Position = gl.getAttribLocation(gl.program,
        'a_Position');
  if (a_Position < 0) { return -1; }
  // Assign the buffer to a_Position variable (4)
  gl.vertexAttribPointer(a_Position, 2, gl.FLOAT,
                false, 0, 0);

  // Enable the assignment to a_Position variable
  gl.enableVertexAttribArray(a_Position); (5)

  return n;
}
```

Create Buffer Object

Figure 3_6 in ML2013

Bind Buffer Object to graphics context

Figure 3_7 in ML2013

Load Buffer Object

Figure 3_8 in ML2013

Get attribute location

Figure 3_9 in ML2013

Connect buffer with attribute

Figure 3_10 in ML2013

Streaming array into attribute

Figure 3_11 in ML2013

Glossary

attribute Declare glsl attribute (per vertex) variable. GLES2.10.4, GLSL4.3.3. 5

console.log Write to the browser JavaScript console. 3

function Define JavaScript anonymous function. 5

getWebGLContext Convenience function in cuon-utils.js. ML2.2. 3

gl.ARRAY_BUFFER WebGL buffer "target" used for vertex data, ML3, https://www.opengl.org/wiki/Vertex_Specification#Vertex_Buffer_Object. 8

gl.bindBuffer Connect WebGL buffer to "target", ML3, http://www.khronos.org/opengles/sdk/2.0/docs/man/xhtml/glBindBuffer.xml. 8

gl.bufferData Load WebGL buffer, ML3, https://www.khronos.org/registry/webgl/specs/1.0/#5.14.5, http://www.khronos.org/opengles/sdk/2.0/docs/man/xhtml/glBufferData.xml. 8

gl.Clear Clear canvas, ML2.2, https://www.khronos.org/registry/webgl/specs/1.0/#5.14.3, http://www.khronos.org/opengles/sdk/2.0/docs/man/xhtml/glClear.xml. 3

gl.clearColor Set Clear Colour, ML2.2, https://www.khronos.org/registry/webgl/specs/1.0/#5.14.3, http://www.khronos.org/opengles/sdk/2.0/docs/man/xhtml/glClearColor.xml. 3

gl.drawArrays Draw arrays of points. https://www.khronos.org/registry/webgl/specs/1.0/#5.14.11, ML7.5, A93. 4, 7, 8

gl_FragColor GLSL fragment shader color output. GLES3.8. 3

gl.getAttribLocation Get storage offset for vertex attribute. http://www.khronos.org/registry/webgl/specs/latest/#5.14.10 GLES2.10.4, ML2.4. 5

gl_Pointsize GLSL vertex shader point. GLES3.3. 3

gl_Position GLSL vertex shader vertex coordinate output. GLES2.12. 3

gl.vertexAttrib3f Get storage offset for vertex attribute. http://www.khronos.org/registry/webgl/specs/latest/#5.14.10 GLES2.10, ML2.4. 5

initShaders Utility function in cuon-utils.js. Compile shaders and link to graphics context. ML9.2. 4

onload JavaScript function to call when the HTML file is loaded. 2

onmousedown JavaScript function to call on mouse click (in canvas). 5

uniform Declare GLSL uniform (nonvarying global) variable. GLSL4.3.4. 7

varying per-vertex variables used to pass values from vertex shader to fragment shader. GLSL4.3.5. 7