AS3 - CS 411

Homework Questions

WebGL

(a) Explain the difference between OpenGL, OpenGL ES, and WebGL.

OpenGL ES is a subset of OpenGL for embedded systems and is used on devices such as mobile phones. WebGL brings the functionality of OpenGL to HTML, and provides similar rendering capabilities as OpenGL ES.

(b) Explain the purpose of the color, depth, and stencil buffers.

The color buffer contains RGB and RGBa pixel data.

The depth buffer is where data is held to ensure that pixels aren't accidentally overridden ontop of eachother.

The stencil buffer allows us to mask or unmask pieces of the buffer. This can be helpful to reveal pixels to render.

(c) Explain the difference between the WebGL functions vertexAttrib1f and vertexAttrib4fv. What are the arguments of each function?

vertexAttrib[1234]f differs from vertexAttrib[1234]fv in that the one that ends with an f can be passed individual vertices, while the one that ends with a v can be passed an array of values.

vertexAttrib1f(index, v0)

vertexAttrib4fv(index, value

Having special versions of the same function makes for more efficient computation, as we don't waste time looking for how many parameters are passed.

(d) Explain the operations that happen between the vertex and fragment shader. What are fragments?

Between the vertex and fragment shader, geometric shape assembly and rasterization occurs. Fragments are pixels of some depth.

(e) List the kind of objects that can be drawn by webgl.

WebGL only draws points, lines and triangles, but nearly anything can be built off of these basics.

(f) Explain how to set a viewport using WebGL. What is the purpose of the viewport transformation?

GL viewport can be set like: gl.viewport(x, y, width, height);

Viewport transformations are important, because they allow us to see objects on the screen in an accurate way to that of the screen we are viewing. Viewing transformations are identical to normal transformations.

Shaders

(a) Explain the difference between attribute, uniform, and varying variables.

Attribute variables are pieces of data that are different for each vertex, while uniform variables are the same for each vertex, i.e. global variables for vertexes passed from the JS program. Varying variables are passed from the vertex shader, and must be float or related types.

(b) What predefined variable must be assigned a value in the vertex shader for rendering to occur? Explain the need for this assignment.

gl_Position is predefined, and must be assigned in the vertex shader or else nothing will be rendered.

(c) What predefined variable must be assigned a value in the fragment shader for rendering to occur? Explain the need for this assignment.

The variables gl_FragColor is predefined, and must be passed to the fragment shader, as they will be used throughout the lifetime of the fragment shader and without it nothing will render.

(d) Explain how vertex color can be passed from the vertex shader to the fragment shader.

Vertex color can be passed from the vertex shader to the fragment shader via a varying variable called u_FragColor and assigning it to the global variable gl_FragColor to put the variable into the pipeline.

(e) Write the commands needed to obtain a pointer to a uniform or attribute variable in the vertex or fragment shaders.

```
var u_fragColor = gl.getUniformLocation(gl.program, 'u_FragColor');
var a_PointSize = gl.getAttribLocation(gl.program, 'a_PointSize');
```

(f) Write the commands used to pass a specific value to an attribute or uniform variable.

Passing specific value to an attribute variable:

```
gl.vertexAttrib3f(a_position, 0.0, 0.0, 0.0);
```

Passing a specific value to a uniform variable:

```
gl.uniform4f(u_FragColor, rgba[0], rgba[1], rgba[2], rgba[3]);
```

(g) Explain the steps involved in creating a shader program at run time. What is the advantage in compiling the shaders at run time?

Normally, we use initShaders(gl, vshader, fshader) to create the shader program, but the steps normally include: creating shader objects, load and compile the shaders, create a shader program, attach the shaders to the shader program, link the program to the graphics card, using the program.

(h) What is the command that is used to define the type of objects that will be assembled?

gl.drawArrays(gl.POINTS, 0, n); Where the first parameter in gl.drawArrays informs gl what we are looking to draw.

(i) Write a basic vertex shader that receives a position attribute and a transformation matrix, and applies the transformation matrix to the position. Explain how the transformation matrix is prepared in the JavaScript program.

```
attribute vec4 a_Position; // position attribute
uniform mat4 u_TransformMatrix;  // transformation matrix
```

```
void main() {
	var gl_Position = a_Position + u_TransformMatrix;
}
```

Transformations are applied in the vertex shader.

Vertex buffer objects

(a) Explain the difference between ARRAY_BUFFER and ELEMENT_ARRAY_BUFFER. Which one is more efficient in storage and why?

ARRAY_BUFFER specifies that the buffer object contains vertex data, while ELEMENT_ARRAY_BUFFER specifies that the buffer object contains index values pointing to vertex data.

(b) List the steps needed to use a vertex buffer object.

In the javascript program, you need to create it, bind it, allocate storage and initialize it with data, assign it to an attribute variable, and enable the attribute variable as an array.

(c) Explain the arguments of the function vertexAttribPointer.

Location: pointer to the shader's attribute variable
Size: number of elements per vertex
Type: data type in the buffer object
Normalized: specifies whether fixed-point data values should be normalized.
Stride: specifies the byte offset between consecutive vertex attributes.
Offset: specifies the offset in bytes to the first component of the first vertex attribute in the array

(d) What is the command that has to be executed for an attribute array to take effect? What is the command to disable the attribute array?

gl.enableVertexAttribArray(a_Position);
gl.disableVertexAttribArray(a_Position);

(e) What is the command that is used to delete a vertex buffer object?

gl.deleteBuffer(buffer);

(f) Given a Float32Array array of interleaved 2D vertices and RGB colors, whereas each cell used FSIZE elements, write the vertexAttribPointer command that will be used to access the vertex coordinates and the vertexAttribPointer that will be used to access colors.

vertexAttribPointer(a_Color, 3, gl.FLOAT, false, FSIZE * 5, FSIZE * 2);

Challenges

This assignment was a lot of trial and error for me, as are many assignments. I think the hardest part was getting started, especially the fragment and vertex shaders. I had no idea what I was doing when I was coding them, and had to look up tons of examples and talk to friends

before I finally got it. I ignored adding new lines to the shader once the errors got ironed out. I found myself tripping over \n and the ‘’ as I went.

Actually making a triangle fan was pretty straightforward after I looked up an example. I wanted to make a 5 pointed star, but I ended up giving up when I realized I'd probably have to hardcode all the triangles. I then instead made a 10 sided polygon, that is kinda circular, just because the code was much easier. I couldn’t figure out how to close the circle, so I just added the final triangle manually, without much problem. Simply, just used Math.Random to get a random number to calculate out the RGB value.

I am really not sure how the rotation/translation buttons were supposed to work. I have them set up so that they only ever fire once, and you must refresh to do the transformation again. I also just applied the most bare minimum transformation, without much funny business in trying to make it look super nice. The directions did not specify what kind of effort I needed to put into the buttons, so I assumed that I could just transform the matrices and went ahead with that. In the future, I’d try troubleshooting the rotation and translation code.

Demo

Tested in Google Chrome Version 93.0.4577.82.