

COMP 370 assignment #6: WebGL Texture Mapping

Thomas Williamson

id: 588206

2021/12/01

The goal of this project is to make a rotating 3d cube with texture mapping.

Startup



Rotate X

Rotate Y

Rotate Z

Toggle Rotation



Rotate X

Rotate Y

Rotate Z

Toggle Rotation

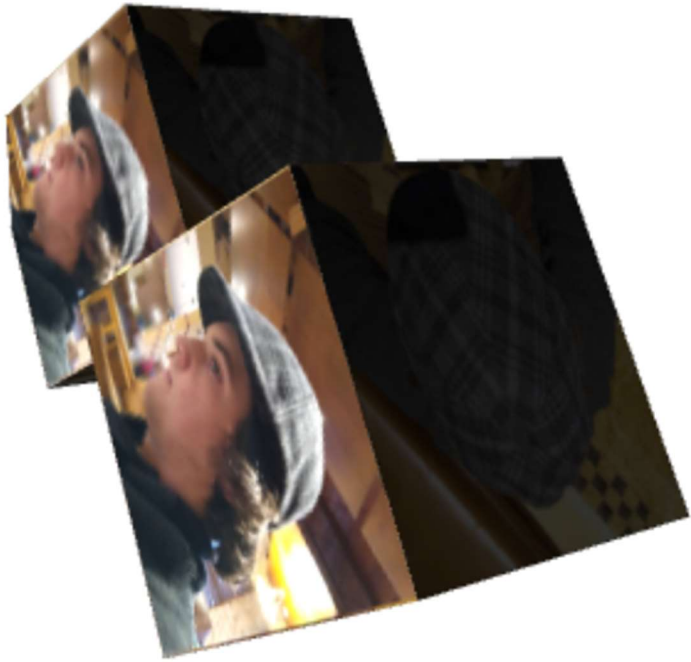


Rotate X

Rotate Y

Rotate Z

Toggle Rotation



Rotate X

Rotate Y

Rotate Z

Toggle Rotation

```

1  <!--
2  COMP 370 assignment #6: WebGL Texture Mapping
3  Thomas Williamson
4  id: 588206
5  2021/12/01
6  -->
7  <!DOCTYPE html>
8  <html>
9  <head>
10     <title>Texture Mapping</title>
11 </head>
12 <body>
13 <canvas id="gl-canvas" width="512" height="512"> </canvas>
14 <img id = "texImage" src = "Six_Photos_of_Your_Face.png" hidden><
img>
15 <div>
16     <button id = "ButtonX">Rotate X</button>
17     <button id = "ButtonY">Rotate Y</button>
18     <button id = "ButtonZ">Rotate Z</button>
19     <button id = "ButtonT">Toggle Rotatation</button>
20 </div>
21
22 <script id="vertex-shader" type="x-shader/x-vertex">
23 #version 300 es
24
25 in vec4 aPosition;
26 in vec2 aTexCoord;
27 in vec3 aNormal;
28 in vec4 aColor;
29
30 out vec4 vColor;
31 out vec2 vTexCoord;
32
33 uniform mat4 uModelViewMatrix;
34 uniform mat4 uProjectionMatrix;
35 uniform vec4 uAmbientProduct, uDiffuseProduct, uSpecularProduct;
36 uniform vec4 uLightPosition;
37 uniform float uShininess:

```

```

38
39 void main()
40 {
41     vec3 pos = aPosition.xyz;
42
43     //light position
44     vec3 light = uLightPosition.xyz;
45
46     vec3 L = normalize(light - pos);
47     vec3 E = normalize(-pos);
48     vec3 H = normalize(L + E);
49     vec4 NN = vec4(aNormal,0);
50
51     // Transform vertex normal into eye coordinates
52     vec3 N = normalize((uModelViewMatrix*NN).xyz);
53
54     // Compute terms in the illumination equation
55     vec4 ambient = uAmbientProduct;
56
57     float Kd = max(dot(L, N), 0.0);
58     vec4 diffuse = Kd * uDiffuseProduct;
59
60     float Ks = pow( max(dot(N, H), 0.0), uShininess );
61     vec4 specular = Ks * uSpecularProduct;
62
63     if( dot(L, N) < 0.0 ) {
64         specular = vec4(0.0, 0.0, 0.0, 1.0);
65     }
66     gl_Position = uProjectionMatrix * uModelViewMatrix *aPosition;
67     vTexCoord = aTexCoord;
68     vColor = ambient + diffuse + specular + aColor;
69     vColor.a = 1.0;
70 }
71 </script>
72
73 <script id="fragment-shader" type="x-shader/x-vertex">
74 #version 300 es
75

```



```
76 precision mediump float;
77 in vec4 vColor;
78 in vec2 vTexCoord;
79
80 out vec4 fColor;
81
82 uniform sampler2D uTextureMap;
83 uniform vec4 uDiffuseProduct;
84
85 void main()
86 ▼ {
87     vec4 T = texture(uTextureMap, vTexCoord);
88     fColor = T * vColor;
89 }
90 </script>
91
92 <script src="../../Common/initShaders.js"></script>
93 <script src="../../Common/MV.js"></script>
94 <script src="Assignment6.js"></script>
95 </body>
96 </html>
97
```

```

1  /*<!--
2  COMP 370 assignment #6: WebGL Texture Mapping
3  Thomas Williamson
4  id: 588206
5  2021/12/01
6  -->*/
7  "use strict";
8  /*global variable defined*/
9  var canvas;
10 var gl;
11 var numPositions = 36;
12 //var texSize = 64;
13 var program;
14 var positionsArray = [];
15 var vertices = [
16     vec3(-0.5, -0.5, 0.5),
17     vec3(-0.5, 0.5, 0.5),
18     vec3(0.5, 0.5, 0.5),
19     vec3(0.5, -0.5, 0.5),
20     vec3(-0.5, -0.5, -0.5),
21     vec3(-0.5, 0.5, -0.5),
22     vec3(0.5, 0.5, -0.5),
23     vec3(0.5, -0.5, -0.5)];
24 var texCoordsArray = new Float32Array([
25     // select the top left image
26     0, 0,
27     0, 0.5,
28     0.25, 0.5,
29     0, 0,
30     0.25, 0.5,
31     0.25, 0,
32     // select the top middle image
33     0.25, 0,
34     0.25, 0.5,
35     0.5, 0.5,
36     0.25, 0,
37     0.5, 0.5,

```

```

38     0.5 , 0 ,
39     // select to top right image
40     0.5 , 0.5,
41     0.75, 0.5,
42     0.75, 0 ,
43     0.5 , 0.5,
44     0.75, 0 ,
45     0.5, 0,
46
47     // select the bottom left image
48     0.25 , 0.5 ,
49     0 , 0.5,
50     0 , 1 ,
51     0.25 , 0.5 ,
52     0 , 1,
53     0.25 , 1 ,
54     // select the bottom middle image
55     0.5, 1 ,
56     0.5 , 0.5,
57     0.25, 0.5,
58     0.5 , 1 ,
59     0.25, 0.5,
60     0.25, 1 ,
61     // select the bottom right image
62
63     0.5 , 0.5,
64     0.5 , 1 ,
65     0.75, 1 ,
66     0.5 , 0.5,
67     0.75 , 1 ,
68     0.75, 0.5,
69     ]);
70     var normalsArray = [
71         //front
72         vec3(0.0, 0.0, 1.0),
73         vec3(0.0, 0.0, 1.0),
74         vec3(0.0, 0.0, 1.0),

```

```
75     vec3(0.0, 0.0, 1.0),
76     vec3(0.0, 0.0, 1.0),
77     vec3(0.0, 0.0, 1.0),
78
79     //right
80     vec3(1.0, 0.0, 0.0),
81     vec3(1.0, 0.0, 0.0),
82     vec3(1.0, 0.0, 0.0),
83     vec3(1.0, 0.0, 0.0),
84     vec3(1.0, 0.0, 0.0),
85     vec3(1.0, 0.0, 0.0),
86     // Bottom
87
88     vec3(0.0, -1, 0.0),
89     vec3(0.0, -1, 0.0),
90     vec3(0.0, -1, 0.0),
91     vec3(0.0, -1, 0.0),
92     vec3(0.0, -1, 0.0),
93     vec3(0.0, -1, 0.0),
94     //top
95     vec3(0.0, 1, 0.0),
96     vec3(0.0, 1, 0.0),
97     vec3(0.0, 1, 0.0),
98     vec3(0.0, 1, 0.0),
99     vec3(0.0, 1, 0.0),
100    vec3(0.0, 1, 0.0),
101    //back
102
103    vec3(0.0, 0.0, -1.0),
104    vec3(0.0, 0.0, -1.0),
105    vec3(0.0, 0.0, -1.0),
106    vec3(0.0, 0.0, -1.0),
107    vec3(0.0, 0.0, -1.0),
108    vec3(0.0, 0.0, -1.0),
109
110    //left
111
```

```

112     vec3(-1.0, 0.0, 0.0),
113     vec3(-1.0, 0.0, 0.0),
114     vec3(-1.0, 0.0, 0.0),
115     vec3(-1.0, 0.0, 0.0),
116     vec3(-1.0, 0.0, 0.0),
117     vec3(-1.0, 0.0, 0.0),
118 ];
119 var texture;
120 var xAxis = 0;
121 var yAxis = 1;
122 var zAxis = 2;
123 var axis = xAxis;
124 var theta = vec3(0.0, 0.0, 0.0);
125 var flag = false;
126 var modelViewMatrixLoc;
127 var projectionMatrixLoc;
128 var textureLocation;
129 var viewMatrix;
130 var time = 0;
131 var projectionMatrix;
132 var modelViewMatrix;
133 ▼ var cubes = [
134     translate(0, 0, 0),
135     translate(1, 0, 1)
136 ];
137 var lightPosition = vec4(0.0, 1.0, 2.0, 0.0);
138 var lightAmbient = vec4(0.2, 0.2, 0.2, 1.0);
139 var lightDiffuse = vec4(1.0, 1.0, 1.0, 1.0);
140 var lightSpecular = vec4(1.0, 1.0, 1.0, 1.0);
141 var materialAmbient = vec4(1.0, 1.0, 1.0, 1.0 );
142 var materialDiffuse = vec4(1.0, 1.0, 1.0, 1.0);
143 var materialSpecular = vec4(1.0, 1.0, 1.0, 1.0 );
144 var materialShininess = 100;
145
146
147 ▼ function quad(a, b, c, d) {
148     positionsArray.push(vertices[a]);

```

```

149
150     positionsArray.push(vertices[b]);
151
152     positionsArray.push(vertices[c]);
153
154
155     positionsArray.push(vertices[a]);
156
157     positionsArray.push(vertices[c]);
158
159     positionsArray.push(vertices[d]);
160 }
161
162
163 function colorCube()
164 {
165     quad(1, 0, 3, 2);
166     quad(2, 3, 7, 6);
167     quad(3, 0, 4, 7);
168     quad(6, 5, 1, 2);
169     quad(4, 5, 6, 7);
170     quad(5, 4, 0, 1);
171 }
172
173 //Execute a JavaScript immediately after a page has been loaded
174 window.onload = function init(){
175
176     //Initialize the canvas by document.getElementById method
177     canvas = document.getElementById("gl-canvas");
178     gl = canvas.getContext('webgl2');
179     if (!gl){
180         alert("WebGL 2.0 isn't available");
181     }
182     //set the viewport and canvas background color
183     gl.viewport(0, 0, canvas.width, canvas.height);
184     gl.clearColor(1.0, 1.0, 1.0, 1);
185

```



```

186     gl.enable(gl.DEPTH_TEST);
187
188     //Load shaders and initialize attribute buffers
189     program = initShaders(gl,"vertex-shader", "fragment-shader");
190     gl.useProgram(program);
191
192     colorCube()
193     //Create buffer for normals
194     var nBuffer = gl.createBuffer();
195     gl.bindBuffer(gl.ARRAY_BUFFER, nBuffer);
196     gl.bufferData(gl.ARRAY_BUFFER, flatten(normalsArray), gl.STATIC_DRAW);
197     var normalLoc = gl.getAttribLocation(program, "aNormal");
198     gl.vertexAttribPointer(normalLoc, 3, gl.FLOAT, false, 0, 0);
199     gl.enableVertexAttribArray(normalLoc);
200
201     //Create buffer for vertex
202     var vBuffer = gl.createBuffer();
203     gl.bindBuffer(gl.ARRAY_BUFFER, vBuffer);
204     gl.bufferData(gl.ARRAY_BUFFER, flatten(positionsArray), gl.STATIC_DRAW);
205     var positionLoc = gl.getAttribLocation(program, "aPosition");
206     gl.vertexAttribPointer(positionLoc, 3, gl.FLOAT, false, 0, 0);
207     gl.enableVertexAttribArray(positionLoc);
208
209     //Create buffer for texture coordinate
210     var tBuffer = gl.createBuffer();
211     gl.bindBuffer(gl.ARRAY_BUFFER, tBuffer);
212     gl.bufferData(gl.ARRAY_BUFFER, texCoordsArray, gl.STATIC_DRAW);
213     var texCoordLoc = gl.getAttribLocation(program, "aTexCoord");
214     gl.vertexAttribPointer(texCoordLoc, 2, gl.FLOAT, false, 0, 0);
215     gl.enableVertexAttribArray(texCoordLoc);
216
217     //load an image
218     var image = document.getElementById("texImage");
219     configureTexture(image);
220     modelViewMatrixLoc = gl.getUniformLocation(program, "uModelViewMatrix");
221     projectionMatrixLoc = gl.getUniformLocation(program, "uProjectionMatrix");
222

```

```

223 //set the perspective projection
224 var fieldOfView = 50; //Change the value
225 var aspect = canvas.width/canvas.height;
226 var zNear = 1; //Change the value
227 var zFar = 5; //Change the value
228 projectionMatrix = perspective(fieldOfView, aspect, zNear, zFar);
229
230 //set the model-view matrix
231 var cameraPosition = vec3(2, 2, 4);
232 var up = vec3(0.0, 1.0, 0.0);
233 var target = vec3(0.0, 0.0, 0.0);
234 modelViewMatrix = lookAt(cameraPosition, target, up);
235
236 //set event to the buttons
237 document.getElementById("ButtonX").onclick = function(){axisX = 1;};
238 document.getElementById("ButtonY").onclick = function(){axisY = 1;};
239 document.getElementById("ButtonZ").onclick = function(){axisZ = 1;};
240 document.getElementById("ButtonT").onclick = function(){flag = 1;};
241 render();
242 }
243
244 //function for setting the texture
245 function configureTexture(image){
246     texture = gl.createTexture();
247     gl.bindTexture(gl.TEXTURE_2D, texture);
248     gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, 1, 1, 0, gl.RGBA, gl.UNSIGNED_BYTE,
249         new Uint8Array([0, 0, 255, 255]));
250     gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, image);
251     //generate the Mipmap
252     gl.generateMipmap(gl.TEXTURE_2D);
253     gl.bindTexture(gl.TEXTURE_2D, texture);
254     gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);
255     gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LINEAR);
256 }
257
258 //render function
259 function render(){

```



```

260 gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
261 if(flag){
262     theta[axis] += 0.3; //Change the value
263 }
264 //rotating the light
265 lightPosition[0] = 5.5 * Math.sin(0.02 * time);
266 lightPosition[2] = 5.5 * Math.cos(0.02 * time);
267 time += .5; //Change the value
268
269 //generate two cubes, one is closer to the viewer and the other
270 for(var index = 0; index < cubes.length; index++){
271     gl.uniform4fv( gl.getUniformLocation(program, "uLightPosit
272     viewMatrix = mult(modelViewMatrix, cubes[index]);
273     viewMatrix = mult(viewMatrix, rotate(theta[xAxis], vec3(1,
274     viewMatrix = mult(viewMatrix, rotate(theta[yAxis], vec3(0,
275     viewMatrix = mult(viewMatrix, rotate(theta[zAxis], vec3(0,
276     gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(vie
277     gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(pr
278
279     var diffuseProduct = mult(lightDiffuse, materialDiffuse);
280     gl.uniform4fv(gl.getUniformLocation(program, "uDiffuseProd
281
282     var ambientProduct = mult(lightAmbient, materialAmbient);
283     gl.uniform4fv(gl.getUniformLocation(program, "uAmbientProd
284
285     var specularProduct = mult(lightSpecular, materialSpecular
286     gl.uniform4fv(gl.getUniformLocation(program, "uSpecularPro
287
288     gl.uniform1f(gl.getUniformLocation(program, "uShininess"),
289     gl.uniform1i(textureLocation, index);
290     gl.drawArrays(gl.TRIANGLES, 0, numPositions);
291 }
292 requestAnimationFrame(render);
293 }
294
295
296

```