

Kathmandu University
School of Engineering
Department of Computer Science and Engineering (DoCSE)
Dhulikhel, Kavre



Computer Graphics - Lab 1
A flag of Nepal using the WebGL

Submitted By:

Mission Shrestha (54)

CE 3rd Year 2nd Semester

Submitted to:

Mr. Dhiraj Shrestha,

DoCSE

Date of Submission: 2023/03/29

Mention the name of Programming language and Graphics Library you are using this semester for performing your Computer Graphics Lab and Project.

: This semester, I will be using JavaScript as the programming language and WebGL as the graphics library for my Computer Graphics Lab and Project.

Write the code snippets for setting graphics environment in your chosen graphics library and display the resolution of your display system through functions/classes provided by your graphics library

The code is written using WebGL for rendering graphics. The following code snippets show how the graphics environment is set in WebGL:

- 1) Create a canvas element in the HTML document

```
<!-- 54 Mission Shrestha -->

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Flag of Nepal</title>
    <link rel="stylesheet" href="Style/style.css" />
  </head>
  <body>

    <!-- creates a canvas element in the HTML document with an id of "glcanvas" -->
    <canvas id="glcanvas" width="640" height="480"></canvas>

    <div id="resolution-container" class="resolution"></div>

    <div class="end-despriction">
      Graphical Representation of Flag of Nepal.
    </div>
    <div class="end-despriction">Done by: Name: Mission Shrestha</div>
    <script src="JS/script.js"></script>
  </body>
</html>
```

2) Creating a WebGL context for rendering:

```
// we select the canvas element from the HTML document
const canvas = document.querySelector("#glcanvas");

// create a WebGL context for rendering
const gl = canvas.getContext("webgl");

// If WebGL is not available, alert the user and stop execution
if (!gl) {
  alert(
    "Unable to initialize WebGL. Your browser or machine may not support it."
  );
}
```

3) Creating a buffer to hold vertex data:

```
const buffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, buffer);
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(vertexData), gl.STATIC_DRAW);
```

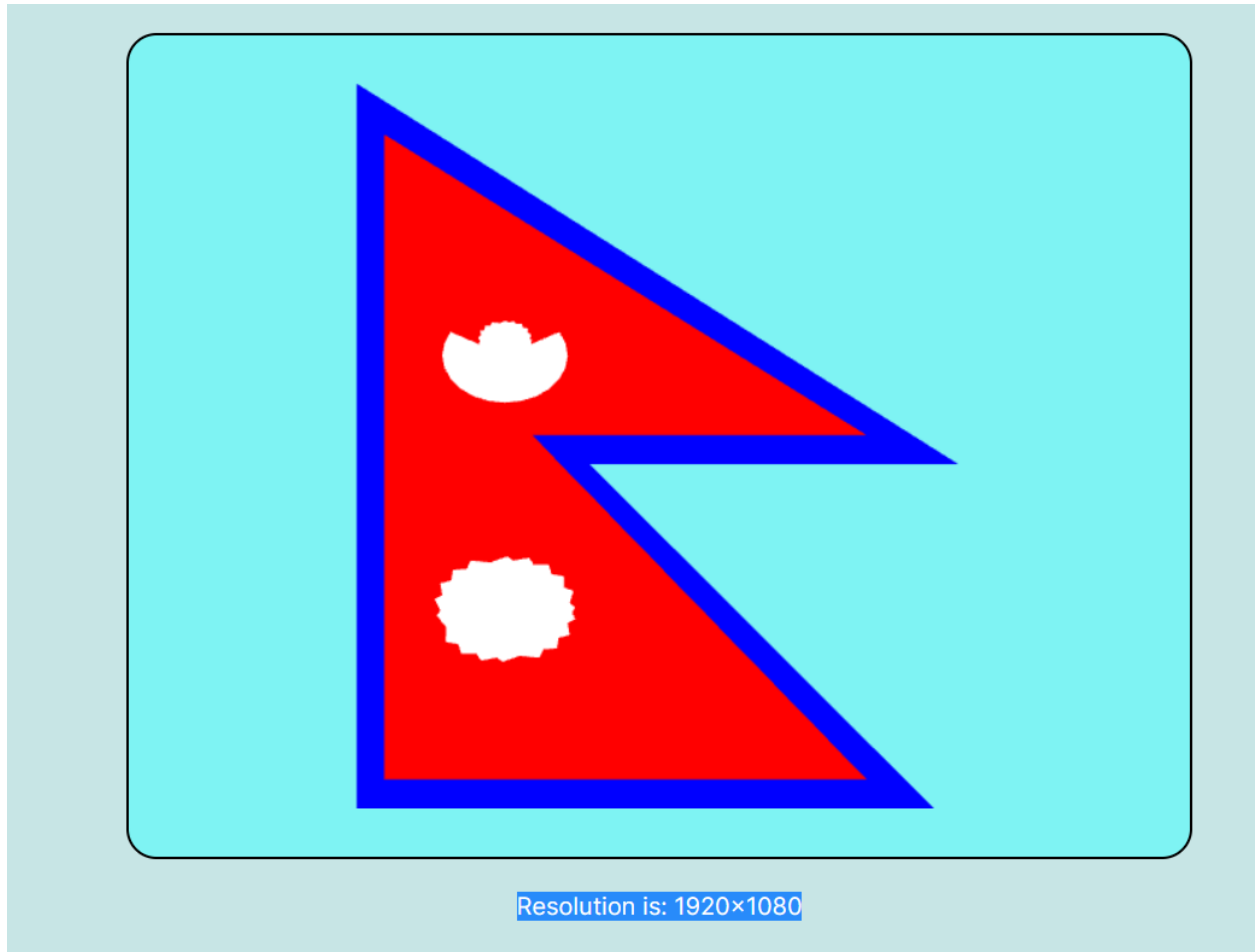
4) To display the resolution of the display system, the following code snippet is used:

```
const resolutionContainer = document.querySelector("#resolution-container");
resolutionContainer.style.margin = "5px";
resolutionContainer.style.color = "black";

// Calculate and display the screen resolution

resolutionContainer.innerHTML =
  "Resolution is: " +
  (window.screen.width * window.devicePixelRatio).toFixed() +
  "x" +
  (window.screen.height * window.devicePixelRatio).toFixed();
```

Display Resolution:



Get Familiar with the coordinate system and Draw a flag of Nepal using the chosen Graphics geometrical functions/ classes provided by your chosen graphics library and also color the flag accordingly.

Following contain the screenshot of the source code.



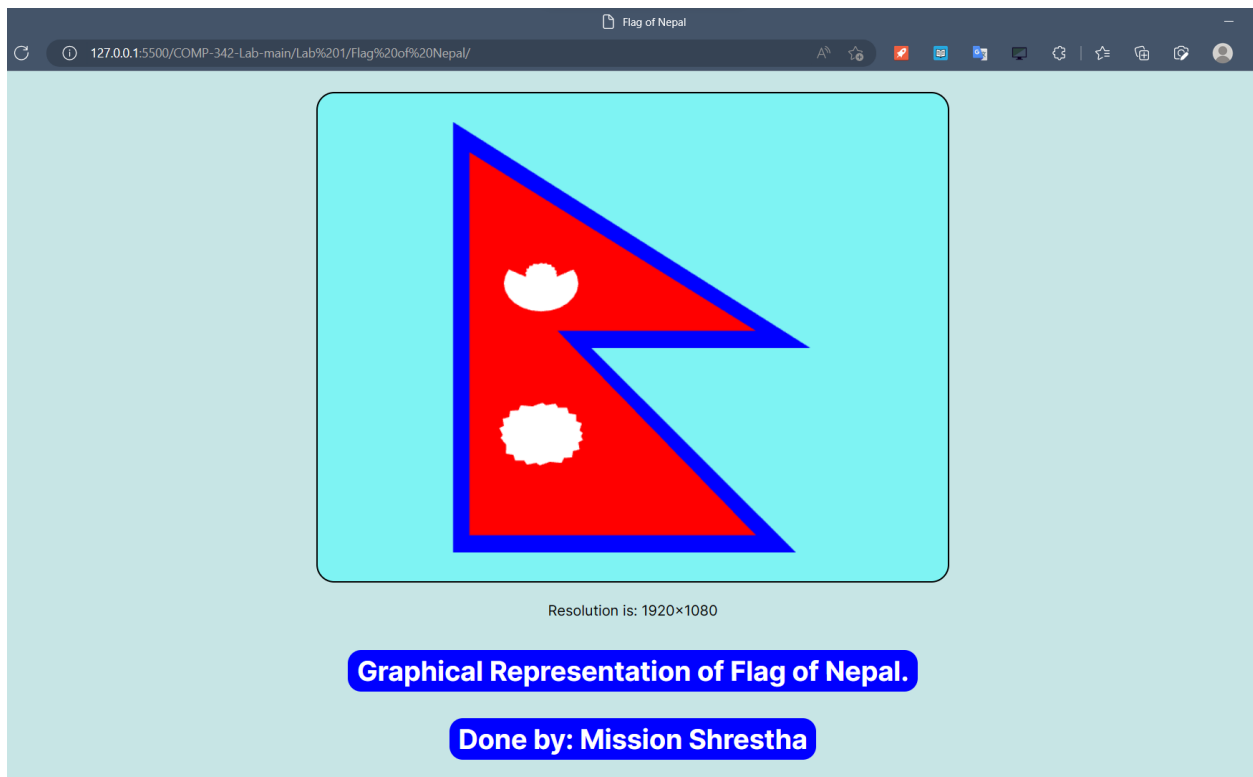
```
1
2 // 54 Mission Shrestha
3
4
5
6 // Creates a vertex with given radius, x-coordinate and y-coordinate
7
8 function createVertex(radius, xo, yo) {
9   let vertices = [];
10
11   // Loop over angles in degrees from 0 to 360 degrees with increments of 15 degrees
12
13   for (i = 0; i < 360; i += 15) {
14     vertices.push(...[xo, yo, 0]);
15     x = radius * Math.cos((Math.PI / 180) * i) + xo;
16     y = radius * Math.sin((Math.PI / 180) * i) + yo;
17     vertices.push(...[x, y, 0]);
18
19     // Calculate coordinates for next point on circumference
20     x = radius * Math.cos((Math.PI / 180) * (i + 15)) + xo;
21     y = radius * Math.sin((Math.PI / 180) * (i + 15)) + yo;
22     vertices.push(...[x, y, 0]);
23   }
24
25   return vertices;
26 }
27
28 // Creates half a vertex with given radius, x-coordinate and y-coordinate
29 function createHalfVertex(radius, xo, yo) {
30   let vertices = [];
31   for (i = 150; i < 390; i += 15) {
32     vertices.push(...[xo, yo, 0]);
33     x = radius * Math.cos((Math.PI / 180) * i) + xo;
34     y = radius * Math.sin((Math.PI / 180) * i) + yo;
35     vertices.push(...[x, y, 0]);
36     x = radius * Math.cos((Math.PI / 180) * (i + 15)) + xo;
37     y = radius * Math.sin((Math.PI / 180) * (i + 15)) + yo;
38     vertices.push(...[x, y, 0]);
39   }
40
41   return vertices;
42 }
43
44 // Creates data for a spike with given radius, x-coordinate and y-coordinate
45 function createSpikeData(radius, xo, yo) {
46   let vertices = [];
47   midWidth = radius / 8;
48   for (i = -20; i < 345; i += 20) {
49     x = radius * Math.cos((Math.PI / 180) * i) + xo;
50     y = radius * Math.sin((Math.PI / 180) * i) + yo;
51     vertices.push(...[x, y, 0]);
52     x = radius * Math.cos((Math.PI / 180) * (i + 30)) + xo;
53     y = radius * Math.sin((Math.PI / 180) * (i + 30)) + yo;
54     vertices.push(...[x, y, 0]);
55     if (i <= 90) {
56       x = radius * Math.cos((Math.PI / 180) * (i + 15)) + xo + midWidth;
57       y = radius * Math.sin((Math.PI / 180) * (i + 15)) + yo + midWidth;
58       vertices.push(...[x, y, 0]);
59     } else if (i <= 180) {
60       x = radius * Math.cos((Math.PI / 180) * (i + 15)) + xo - midWidth;
61       y = radius * Math.sin((Math.PI / 180) * (i + 15)) + yo + midWidth;
62       vertices.push(...[x, y, 0]);
63     } else if (i <= 270) {
64       x = radius * Math.cos((Math.PI / 180) * (i + 15)) + xo - midWidth;
65       y = radius * Math.sin((Math.PI / 180) * (i + 15)) + yo - midWidth;
66       vertices.push(...[x, y, 0]);
67     } else if (i <= 360) {
68       x = radius * Math.cos((Math.PI / 180) * (i + 15)) + xo + midWidth;
69       y = radius * Math.sin((Math.PI / 180) * (i + 15)) + yo - midWidth;
70       vertices.push(...[x, y, 0]);
71     }
72   }
73   return vertices;
74 }
75
76
```

```

1
2 // 3D Matrix library
3
4 // we want the canvas element from the HTML document
5 const canvas = document.querySelector("#glcanvas");
6
7 // create a WebGL context for rendering
8 const gl = canvas.getContext("webgl");
9
10 // If none is not available, alert the user and stop execution
11 if (!gl) {
12   alert(
13     "Unable to initialize WebGL. Your browser or machine may not support it."
14   );
15 }
16
17 // Define vertex data for various shapes
18
19 let squareVertexData = createVertex(1, 1, -0.5, -0.5);
20 let squareVertexData = createVertex(0.5, 0.5, 0.5);
21 let squareVertexData = createVertex(1, 1, -0.5, 0.5);
22 let squareVertexData = createVertex(0.5, 0.5, 0.5);
23 let squareVertexData = createVertex(0.5, 0.5, 0.5);
24
25 // Combining all vertex data into a single array
26
27 const vertexData = [
28   // square - blue border
29   -0.5, -0.5, 0,
30   -0.5, 0.5, 0,
31   0.5, 0.5, 0,
32   0.5, -0.5, 0,
33   // inner blue square
34   -0.4, -0.4, 0,
35   -0.4, 0.4, 0,
36   0.4, 0.4, 0,
37   0.4, -0.4, 0,
38   // inner red
39   -0.3, -0.3, 0,
40   -0.3, 0.3, 0,
41   0.3, 0.3, 0,
42   0.3, -0.3, 0,
43   // inner green
44   -0.2, -0.2, 0,
45   -0.2, 0.2, 0,
46   0.2, 0.2, 0,
47   0.2, -0.2, 0,
48   // blue
49   ...squareVertexData,
50   ...squareVertexData,
51   // blue
52   ...squareVertexData,
53   ...squareVertexData,
54   ...squareVertexData,
55   ...squareVertexData,
56 ];
57
58 const buffer = gl.createBuffer();
59 gl.bufferData(gl.ARRAY_BUFFER, buffer);
60 gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(vertexData), gl.STATIC_DRAW);
61
62 const vertexShader = gl.createShader(GL_VERTEX_SHADER);
63 gl.shaderSource(
64   vertexShader,
65   `
66     attribute vec3 position;
67     void main() {
68       gl_Position = vec3(position, 1);
69     }
70   `
71 );
72 gl.compileShader(vertexShader);
73
74 function drawTriangle(fragmentShaderGLSL, start, end) {
75   const fragmentShader = gl.createShader(GL_FRAGMENT_SHADER);
76   gl.shaderSource(fragmentShader, fragmentShaderGLSL);
77   gl.compileShader(fragmentShader);
78
79   const program = gl.createProgram();
80   gl.attachShader(program, vertexShader);
81   gl.attachShader(program, fragmentShader);
82   gl.linkProgram(program);
83
84   const positionLocation = gl.getAttribLocation(program, "position");
85   gl.vertexAttribPointer(positionLocation, 3, gl.FLOAT, false, 0, 0);
86
87   gl.useProgram(program);
88
89   gl.drawArrays(gl.TRIANGLES, start, end);
90 }
91
92 // inner blue border
93 drawTriangle(
94   `
95     void main() {
96       gl_FragColor = vec4(0, 0, 1, 1);
97     }
98   `,
99   // start position of vertexData array
100   // end position of vertexData array
101 );
102
103 // inner green border
104 drawTriangle(
105   `
106     void main() {
107       gl_FragColor = vec4(0, 0, 1, 1);
108     }
109   `,
110   // start position of vertexData array
111   // end position of vertexData array
112 );
113
114 // inner red
115 drawTriangle(
116   `
117     void main() {
118       gl_FragColor = vec4(1, 0, 0, 1);
119     }
120   `,
121   // start position of vertexData array
122   // end position of vertexData array
123 );
124
125 // inner blue
126 drawTriangle(
127   `
128     void main() {
129       gl_FragColor = vec4(0, 0, 1, 1);
130     }
131   `,
132   // start position of vertexData array
133   // end position of vertexData array
134 );
135
136 // inner green
137 drawTriangle(
138   `
139     void main() {
140       gl_FragColor = vec4(0, 1, 0, 1);
141     }
142   `,
143   // start position of vertexData array
144   // end position of vertexData array
145 );
146
147 // inner red
148 drawTriangle(
149   `
150     void main() {
151       gl_FragColor = vec4(1, 0, 0, 1);
152     }
153   `,
154   // start position of vertexData array
155   // end position of vertexData array
156 );
157
158 // inner blue
159 drawTriangle(
160   `
161     void main() {
162       gl_FragColor = vec4(0, 0, 1, 1);
163     }
164   `,
165   // start position of vertexData array
166   // end position of vertexData array
167 );
168
169 // inner green
170 drawTriangle(
171   `
172     void main() {
173       gl_FragColor = vec4(0, 1, 0, 1);
174     }
175   `,
176   // start position of vertexData array
177   // end position of vertexData array
178 );
179
180 const resolutionContainer = document.querySelector("#resolution-container");
181 resolutionContainer.style.margin = "5px";
182 resolutionContainer.style.color = "red";
183
184 // Calculate and display the screen resolution
185
186 resolutionContainer.innerHTML =
187   "Resolution is: " +
188   (window.screen.width * window.devicePixelRatio).toFixed(2) +
189   "x" +
190   (window.screen.height * window.devicePixelRatio).toFixed(2);

```


Output:



Conclusion:

By using the geometrical functions and classes offered by WebGL, I was able to achieve a rough rendering of the flag of Nepal. Though this lab presented several challenges and confusion throughout its execution, it ultimately granted me a much deeper comprehension of WebGL's underlying mechanics as a result of completing it successfully.