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

```html
<!-- 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);
```

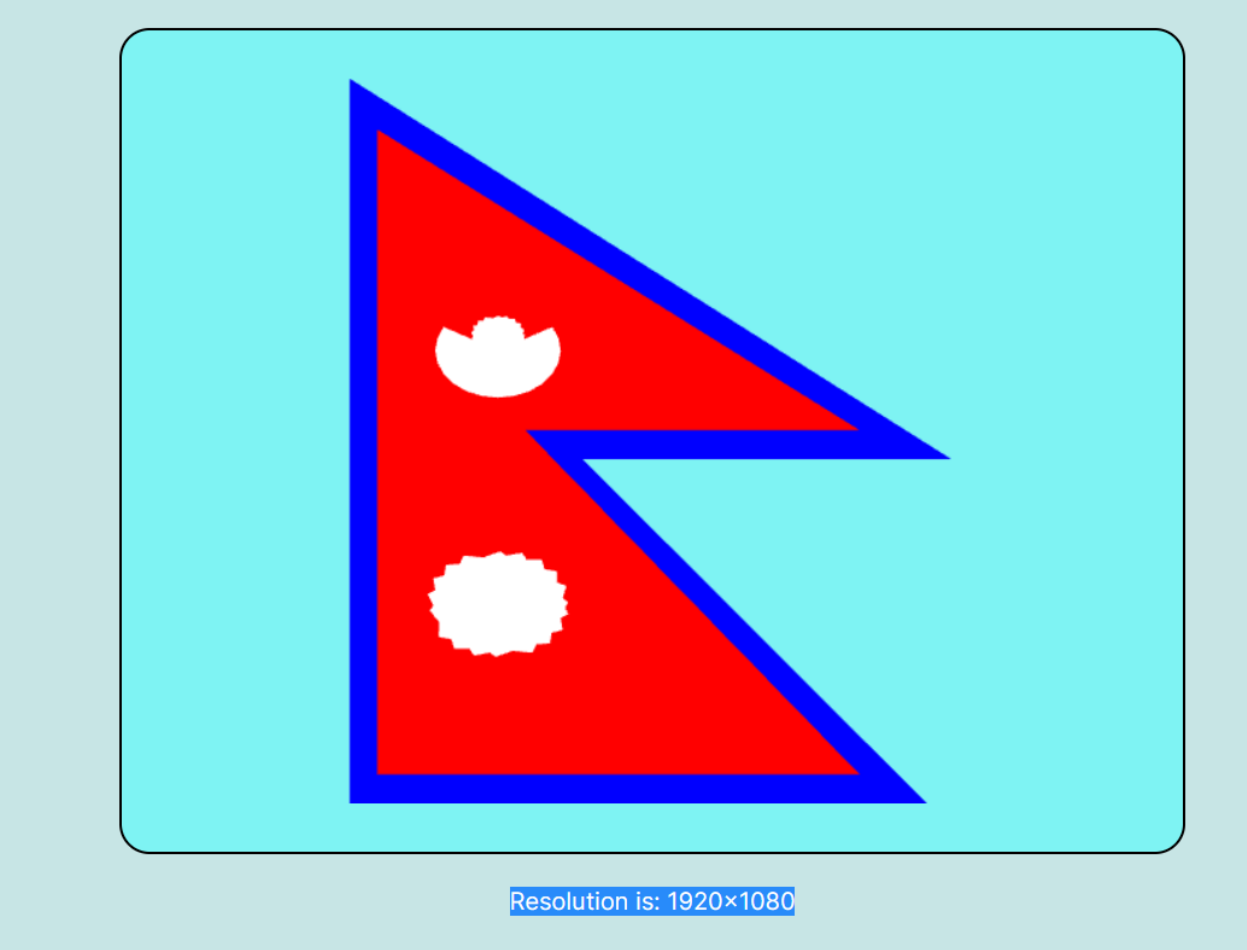
4) To display the resolution of the display system, the following code snippet is used:

```
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.

```

// 54 Mission Shrestha



// Creates a vertex with given radius, x-coordinate and y-coordinate

function createVertex(radius, xo, yo) {
  let vertices = [];

  // Loop over angles in degrees from 0 to 360 degrees with increments of 15 degrees

  for (i = 0; i < 360; i += 15) {
    vertices.push(...[xo, yo, 0]);
    x = radius * Math.cos((Math.PI / 180) * i) + xo;
    y = radius * Math.sin((Math.PI / 180) * i) + yo;
    vertices.push(...[x, y, 0]);

    // Calculate coordinates for next point on circumference
    x = radius * Math.cos((Math.PI / 180) * (i + 15)) + xo;
    y = radius * Math.sin((Math.PI / 180) * (i + 15)) + yo;
    vertices.push(...[x, y, 0]);
  }

  return vertices;
}

// Creates half a vertex with given radius, x-coordinate and y-coordinate
function createHalfVertex(radius, xo, yo) {
  let vertices = [];
  for (i = 150; i < 390; i += 15) {
    vertices.push(...[xo, yo, 0]);
    x = radius * Math.cos((Math.PI / 180) * i) + xo;
    y = radius * Math.sin((Math.PI / 180) * i) + yo;
    vertices.push(...[x, y, 0]);
    x = radius * Math.cos((Math.PI / 180) * (i + 15)) + xo;
    y = radius * Math.sin((Math.PI / 180) * (i + 15)) + yo;
    vertices.push(...[x, y, 0]);
  }

  return vertices;
}

// Creates data for a spike with given radius, x-coordinate and y-coordinate
function createSpikeData(radius, xo, yo) {
  let vertices = [];
  midWidth = radius / 8;
  for (i = -20; i < 345; i += 20) {
    x = radius * Math.cos((Math.PI / 180) * i) + xo;
    y = radius * Math.sin((Math.PI / 180) * i) + yo;
    vertices.push(...[x, y, 0]);
    x = radius * Math.cos((Math.PI / 180) * (i + 30)) + xo;
    y = radius * Math.sin((Math.PI / 180) * (i + 30)) + yo;
    vertices.push(...[x, y, 0]);
    if (i <= 90) {
      x = radius * Math.cos((Math.PI / 180) * (i + 15)) + xo + midWidth;
      y = radius * Math.sin((Math.PI / 180) * (i + 15)) + yo + midWidth;
      vertices.push(...[x, y, 0]);
    } else if (i <= 180) {
      x = radius * Math.cos((Math.PI / 180) * (i + 15)) + xo - midWidth;
      y = radius * Math.sin((Math.PI / 180) * (i + 15)) + yo + midWidth;
      vertices.push(...[x, y, 0]);
    } else if (i <= 270) {
      x = radius * Math.cos((Math.PI / 180) * (i + 15)) + xo - midWidth;
      y = radius * Math.sin((Math.PI / 180) * (i + 15)) + yo - midWidth;
      vertices.push(...[x, y, 0]);
    } else if (i <= 360) {
      x = radius * Math.cos((Math.PI / 180) * (i + 15)) + xo + midWidth;
      y = radius * Math.sin((Math.PI / 180) * (i + 15)) + yo - midWidth;
      vertices.push(...[x, y, 0]);
    }
  }
  return vertices;
}



```

```

// 04 Mission Shreetha

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

// Initialize 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."
  );
}

// Define vertex data for various shapes

let sunVertexData = createVertex(0.11, -0.32, -0.03);
let moonVertexData = createVertex(0.03, -0.57, 0.77);
let moonLowerVertexData = createHalfVertex(0.13, -0.32, 0.23);
let sunSpikeData = createSpikeData(0.15, 0.52, -0.03);
let moonSpikeData = createSpikeData(0.03, -0.32, 0.27);

// Combining all vertex data into a single array

const vertexData = [
  // upper blue border
  -0.627, -0.08, 0,
  -0.627, 1, 0,
  0.57, -0.85, 0,

  // lower blue border
  -0.627, -1, 0,
  -0.627, 0.38, 0,
  0.57, -1, 0,

  // upper red
  -0.57, 0.83, 0,
  -0.57, 0.98, 0,
  0.43, 0.83, 0,

  // lower red
  -0.57, -0.92, 0,
  -0.57, 0.98, 0,
  0.43, -0.92, 0,

  // Sun
  ...sunVertexData,
  ...sunSpikeData,

  // Moon
  ...moonVertexData,
  ...moonSpikeData,
  ...moonLowerVertexData


];

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

const vertexShader = gl.createShader(gl.VERTEX_SHADER);
gl.shaderSource(
  vertexShader,
  `
    attribute vec3 position;
    void main() {
      gl_Position = vec4(position, 1);
    }
`
);


gl.compileShader(vertexShader);

function drawTriangle(fragmentShaderGLSL, start, end) {
  const fragmentShader = gl.createShader(gl.FRAGMENT_SHADER);
  gl.shaderSource(fragmentShader, fragmentShaderGLSL);
  gl.compileShader(fragmentShader);

  const program = gl.createProgram();
  gl.attachShader(program, vertexShader);
  gl.attachShader(program, fragmentShader);
  gl.linkProgram(program);

  const positionLocation = gl.getAttribLocation(program, "position");
  gl.enableVertexAttribArray(positionLocation);

  gl.vertexAttribPointer(positionLocation, 3, gl.FLOAT, false, 0, 0);

  gl.useProgram(program);

  gl.drawArrays(gl.TRIANGLES, start, end);
}

// upper blue border
drawTriangle(
  `void main(){
      gl_FragColor = vec4(0, 0, 1, 1);
    }`,
  0, // start postion of vertexData array
  3 // end position of vertexData Array
);

// lower blue border
drawTriangle(
  `void main(){
      gl_FragColor = vec4(0, 0, 1, 1);
    }`,
  3, // start postion of vertexData array
  6 // end position of vertexData Array
);

// upper red
drawTriangle(
  `void main(){
      gl_FragColor = vec4(1, 0, 0, 1);
    }`,
  6, // start postion of vertexData array
  9 // end position of vertexData Array
);

// lower red
drawTriangle(
  `void main(){
      gl_FragColor = vec4(1, 0, 0, 1);
    }`,
  9, // start postion of vertexData array
  12 // end position of vertexData Array
);

// sun
drawTriangle(
  `void main() {
      gl_FragColor = vec4(1, 1, 1, 1);}
    `,
  12,
  81
);

// sun spike
drawTriangle(
  `void main() {
      gl_FragColor = vec4(1, 1, 1, 1);}
    `,
  81,
  117
);

// moon
drawTriangle(
  `void main() {
      gl_FragColor = vec4(1, 1, 1, 1);}
    `,
  117,
  189
);

// moon spike
drawTriangle(
  `void main() {
      gl_FragColor = vec4(1, 1, 1, 1);}
    `,
  189,
  225
);


drawTriangle(
  `void main() {
      gl_FragColor = vec4(1, 1, 1, 1);}
    `,
  225,
  264
);


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


// Calculate and display the screen resolution

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

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.