**Kathmandu University**

**Department of Computer Science and Engineering**

**Dhulikhel, Kavre**



A Report on ‘**Lab Work 3’** [COMP 342]

**Submitted by:**

Chandan Kumar Mahato (31)

III-year, II semester

**Submitted to**:

Mr. Dhiraj Shrestha

Department of Computer Science and Engineering

**Submission Date**: April 18, 2023

***Qlab1***

1. ***Write a Program to implement mid-point Circle Drawing Algorithm***
2. ***Write a Program to implement mid-point Ellipse Drawing Algorithm***

***Solution:***

<https://flagwebgl.netlify.app>

I have used JavaScript as the programming language and WebGL as the graphics library. **JavaScript** is a popular programming language that is used for web development and has the capability of creating interactive graphics and animations on the web. **WebGL** is a graphics library that enables high-performance 3D graphics library that enables high-performance 3D graphics rendering in web browser using JavaScript. These technologies provide a powerful platform for creating engaging and interactive graphics, which is essential for my lab work.

The code snippets for setting graphics environment in my chosen graphics library and programming language and display system resolution are as follow:

**Source Code**

[**https://github.com/ChandankMahato/Graphics\_Lab\_6th\_Sem**](https://github.com/ChandankMahato/Graphics_Lab_6th_Sem)

**Circle Algorithm:**

*function* drawCircleMidPoint() {

*let* vertexData = [];

*let* radius = 100;

*let* xc = 300;

*let* yc = 300;

*let* x = 0;

*let* y = radius;

*let* p = 1 - radius;

  while (x <= y) {

    vertexData.push(normalise(xc + x, canvasWidth));

    vertexData.push(normalise(yc + y, canvasHeight));

    vertexData.push(0);

    vertexData.push(normalise(xc + y, canvasWidth));

    vertexData.push(normalise(yc + x, canvasHeight));

    vertexData.push(0);

    vertexData.push(normalise(xc - x, canvasWidth));

    vertexData.push(normalise(yc + y, canvasHeight));

    vertexData.push(0);

    vertexData.push(normalise(xc - y, canvasWidth));

    vertexData.push(normalise(yc + x, canvasHeight));

    vertexData.push(0);

    vertexData.push(normalise(xc + x, canvasWidth));

    vertexData.push(normalise(yc - y, canvasHeight));

    vertexData.push(0);

    vertexData.push(normalise(xc + y, canvasWidth));

    vertexData.push(normalise(yc - x, canvasHeight));

    vertexData.push(0);

    vertexData.push(normalise(xc - x, canvasWidth));

    vertexData.push(normalise(yc - y, canvasHeight));

    vertexData.push(0);

    vertexData.push(normalise(xc - y, canvasWidth));

    vertexData.push(normalise(yc - x, canvasHeight));

    vertexData.push(0);

    if (p < 0) {

      x++;

      p += 2 \* x + 1;

    } else {

      x++;

      y--;

      p += 2 \* x - 2 \* y + 1;

    }

  }

*let* fragCode = `void main() {gl\_FragColor = vec4(1, 0, 0, 1);}`; //red color

  DrawObject(gl.POINTS, 1, fragCode, vertexData, 0, vertexData.length);

}

**Ellipse Algorithm:**

*function* drawEllipseMidPoint() {

*let* vertexData = [];

*let* a = 200;

*let* b = 100;

*let* xc = 220;

*let* yc = 200;

*let* x = 0;

*let* y = b;

*let* p = b \* b - a \* a \* b + (a \* a) / 4;

*let* dx = 2 \* b \* b \* x;

*let* dy = 2 \* a \* a \* y;

  while (dx < dy) {

    vertexData.push(normalise(xc + x, canvasWidth));

    vertexData.push(normalise(yc + y, canvasHeight));

    vertexData.push(0);

    vertexData.push(normalise(xc - x, canvasWidth));

    vertexData.push(normalise(yc + y, canvasHeight));

    vertexData.push(0);

    vertexData.push(normalise(xc + x, canvasWidth));

    vertexData.push(normalise(yc - y, canvasHeight));

    vertexData.push(0);

    vertexData.push(normalise(xc - x, canvasWidth));

    vertexData.push(normalise(yc - y, canvasHeight));

    vertexData.push(0);

    if (p < 0) {

      x++;

      dx += 2 \* b \* b;

      p += dx + b \* b;

    } else {

      x++;

      y--;

      dx += 2 \* b \* b;

      dy -= 2 \* a \* a;

      p += dx - dy + b \* b;

    }

  }

  p = b \* b \* (x + 0.5) \* (x + 0.5) + a \* a \* (y - 1) \* (y - 1) - a \* a \* b \* b;

  while (y >= 0) {

    vertexData.push(normalise(xc + x, canvasWidth));

    vertexData.push(normalise(yc + y, canvasHeight));

    vertexData.push(0);

    vertexData.push(normalise(xc - x, canvasWidth));

    vertexData.push(normalise(yc + y, canvasHeight));

    vertexData.push(0);

    vertexData.push(normalise(xc + x, canvasWidth));

    vertexData.push(normalise(yc - y, canvasHeight));

    vertexData.push(0);

    vertexData.push(normalise(xc - x, canvasWidth));

    vertexData.push(normalise(yc - y, canvasHeight));

    vertexData.push(0);

    if (p > 0) {

      y--;

      dy -= 2 \* a \* a;

      p += a \* a - dy;

    } else {

      x++;

      y--;

      dx += 2 \* b \* b;

      dy -= 2 \* a \* a;

      p += dx - dy + a \* a;

    }

  }

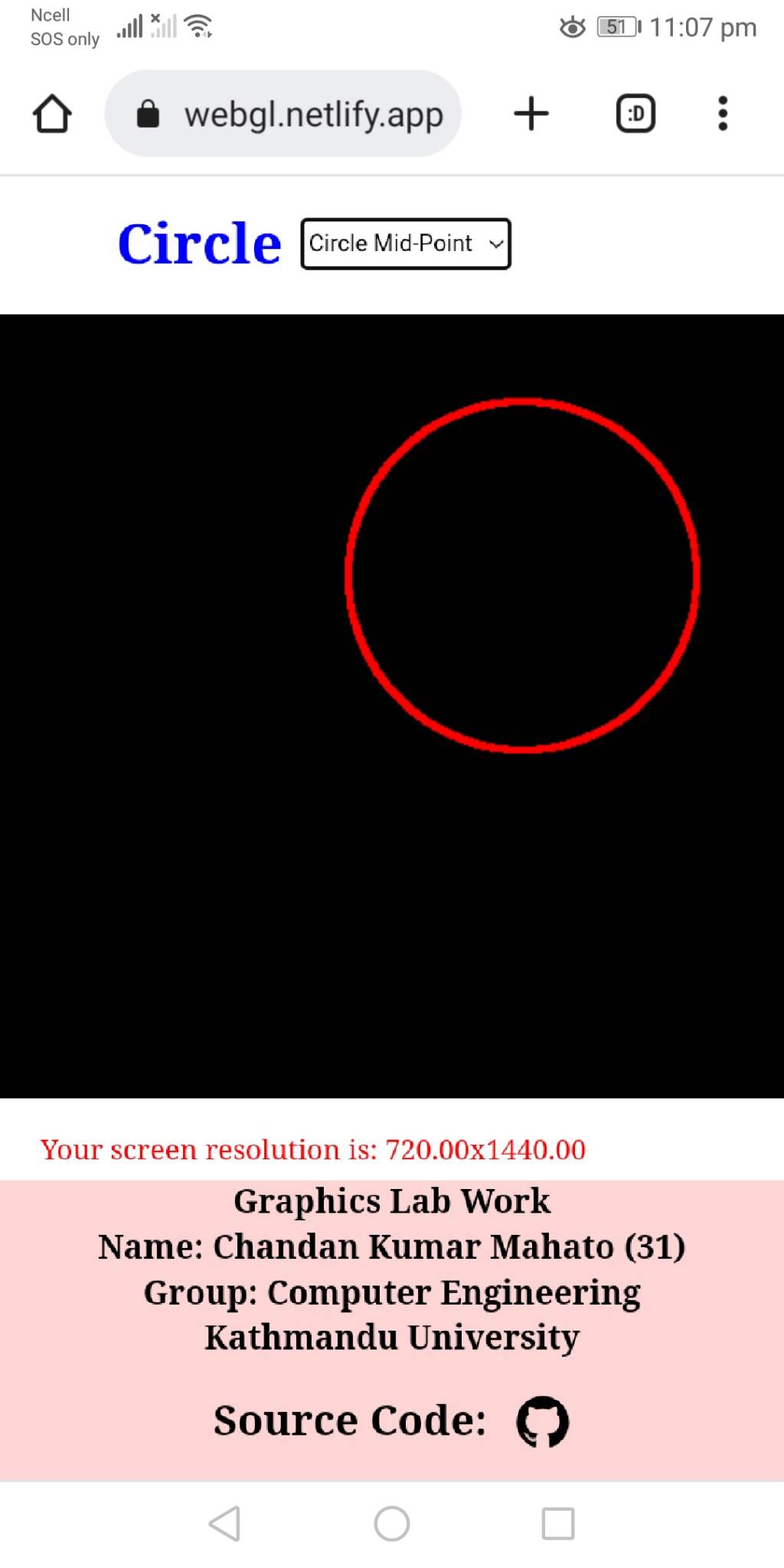
*let* fragCode = `void main() {gl\_FragColor = vec4(1, 0, 0, 1);}`; //red color

  DrawObject(gl.POINTS, 1, fragCode, vertexData, 0, vertexData.length);

}

**Output and Screenshots:**

****

****

**Fig 1: Circle Fig 2: Ellipse**

**Conclusion**

By getting familiar with the coordinate system and utilizing the graphics geometrical functions and classes provided by our chosen graphics library, I was able to successfully draw the circle using mid-point algorithm and Ellipse using mid-point algorithm.

This exercise helped me to develop a better understanding of the coordinate system and the use of graphics geometrical functions and classes to create visually appealing graphics.