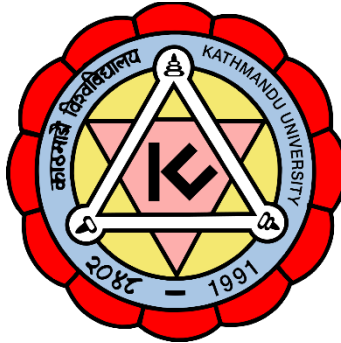


Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre



Computer Graphics(COMP 342) – Lab 3

Submitted To:

Dhiraj Shrestha Sir

Department of Computer Science and Engineering

Submitted By:

Mission Shrestha (54)

CE-2019 Year/Sem: 3rd Year/2nd Sem

Submission Date: 26th April, 2023

Implementation of mid- point Circle & Ellipse Drawing Algorithm.

1. Circle Algorithm

Source Code:

```
// This function takes three parameters: the radius of the circle, and the x and y
coordinates of the circle's center.
function midPointCircle(radius, xc, yc) {
    let tempVertices = [];

    // Initialize the current point on the circumference to (0, radius).
    let x = 0;
    let y = radius;

    // Calculate the initial value of p, which is used in the algorithm to
    determine the next point on the circumference.
    p = 5 / 4 - radius;

    tempVertices = [...tempVertices, ...otherVertices(x, y, xc, yc)];

    // Continue calculating points until the entire circumference has been
    calculated.
    while (x < y) {
        // If p is less than 0, the next point should be to the right of the
        current point.
        if (p < 0) {
            x = x + 1;
            y = y;
            p = p + 2 * x + 1;
        } else {
            // Otherwise, the next point should be to the lower-right of the
            current point.
            x = x + 1;
            y = y - 1;
            p = p + 2 * x + 1 - 2 * y;
        }

        tempVertices = [...tempVertices, ...otherVertices(x, y, xc, yc)];
    }

    return tempVertices;
}

// This is a helper function that calculates the remaining vertices of the circle
given the current point and the center.
function otherVertices(x, y, xc, yc) {
```

```
let tempVertices = [];  
  
// Calculate each vertex and store it in the array.  
tempVertices.push((x + xc) / 600);  
tempVertices.push((y + yc) / 600);  
tempVertices.push(0);  
  
tempVertices.push((y + xc) / 600);  
tempVertices.push((x + yc) / 600);  
tempVertices.push(0);  
  
tempVertices.push((xc - x) / 600);  
tempVertices.push((yc + y) / 600);  
tempVertices.push(0);  
  
tempVertices.push((xc + y) / 600);  
tempVertices.push((yc - x) / 600);  
tempVertices.push(0);  
  
tempVertices.push((xc - x) / 600);  
tempVertices.push((yc - y) / 600);  
tempVertices.push(0);  
  
tempVertices.push((xc - y) / 600);  
tempVertices.push((yc - x) / 600);  
tempVertices.push(0);  
  
tempVertices.push((xc - y) / 600);  
tempVertices.push((yc + x) / 600);  
tempVertices.push(0);  
  
tempVertices.push((xc + x) / 600);  
tempVertices.push((yc - y) / 600);  
tempVertices.push(0);  
  
return tempVertices;  
}
```

```

1 // This function takes three parameters: the radius of the circle, and the x and y coordinates of the circle's center.
2 function midPointCircle(radius, xc, yc) {
3     let tempVertices = [];
4
5     // Initialize the current point on the circumference to (0, radius).
6     let x = 0;
7     let y = radius;
8
9     // Calculate the initial value of p, which is used in the algorithm to determine the next point on the circumference.
10    p = 5 / 4 - radius;
11
12    tempVertices = [...tempVertices, ...otherVertices(x, y, xc, yc)];
13
14    // Continue calculating points until the entire circumference has been calculated.
15    while (x < y) {
16        // If p is less than 0, the next point should be to the right of the current point.
17        if (p < 0) {
18            x = x + 1;
19            y = y;
20            p = p + 2 * x + 1;
21        } else {
22            // Otherwise, the next point should be to the lower-right of the current point.
23            x = x + 1;
24            y = y - 1;
25            p = p + 2 * x + 1 - 2 * y;
26        }
27
28        tempVertices = [...tempVertices, ...otherVertices(x, y, xc, yc)];
29    }
30
31    return tempVertices;
32 }
33
34 // This is a helper function that calculates the remaining vertices of the circle given the current point and the center.
35 function otherVertices(x, y, xc, yc) {
36     let tempVertices = [];
37
38     // Calculate each vertex and store it in the array.
39     tempVertices.push((x + xc) / 600);
40     tempVertices.push((y + yc) / 600);
41     tempVertices.push(0);
42
43     tempVertices.push((y + xc) / 600);
44     tempVertices.push((x + yc) / 600);
45     tempVertices.push(0);
46
47     tempVertices.push((xc - x) / 600);
48     tempVertices.push((yc + y) / 600);
49     tempVertices.push(0);
50
51     tempVertices.push((xc + y) / 600);
52     tempVertices.push((yc - x) / 600);
53     tempVertices.push(0);
54
55     tempVertices.push((xc - x) / 600);
56     tempVertices.push((yc - y) / 600);
57     tempVertices.push(0);
58
59     tempVertices.push((xc - y) / 600);
60     tempVertices.push((yc - x) / 600);
61     tempVertices.push(0);
62
63     tempVertices.push((xc - y) / 600);
64     tempVertices.push((yc + x) / 600);
65     tempVertices.push(0);
66
67     tempVertices.push((xc + x) / 600);
68     tempVertices.push((yc - y) / 600);
69     tempVertices.push(0);
70
71    return tempVertices;
72 }
73

```

2. Ellipse Algorithm

Source Code:

```
3.  // This function takes in the parameters rx (radius along x-axis), ry
    (radius along y-axis), and the center point xc,yc
4.  function midPointEllipse(rx, ry, xc, yc) {
5.      let tempVertices = [];
6.
7.      // Initialize variables
8.      let x = 0;
9.      let y = ry;
10.     let p = ry * ry - rx * rx * ry + (rx * rx) / 4;
11.     let dx = 2 * ry * ry * x;
12.     let dy = 2 * rx * rx * y;
13.
14.     tempVertices = [...tempVertices, ...otherVertice(x, y, xc, yc)];
15.
16.     // This loop calculates the vertices for the first half of the
    ellipse (Region-1)
17.     while (2 * ry * ry * x < 2 * rx * rx * y) {
18.         if (p < 0) {
19.             x = x + 1;
20.             y = y;
21.             p = p + 2 * ry * ry * x + ry * ry;
22.         } else {
23.             x = x + 1;
24.             y = y - 1;
25.             p = p + 2 * ry * ry * x - 2 * rx * rx * y + ry * ry;
26.         }
27.         tempVertices = [...tempVertices, ...otherVertice(x, y, xc,
    yc)];
28.     }
29.
30.     // Calculate the value of p for the second half of the ellipse
    (Region-2)
31.     p = ry * ry * (x + (1 / 2)) * (x + (1 / 2)) + rx * rx * (y - 1) *
    (y - 1) - rx * rx * ry * ry;
```

```

32.      // This loop calculates the vertices for the second half of the
    ellipse (Region-2)
33.      while (y > 0) {
34.          if (p > 0) {
35.              y = y - 1;
36.              p = p + rx * rx - 2 * rx * rx * y;
37.          } else {
38.              x = x + 1;
39.              y = y - 1;
40.              p = p + 2 * ry * ry * x - 2 * rx * rx * y + rx * rx;
41.          }
42.          tempVertices = [...tempVertices, ...otherVertex(x, y, xc,
yc)];
43.      }
44.
45.      return tempVertices;
46.  }
47.
48.  function otherVertex(x, y, xc, yc) {
49.      let tempVertices = [];
50.
51.      tempVertices.push((x + xc) / 600);
52.      tempVertices.push((y + yc) / 600);
53.      tempVertices.push(0);
54.
55.      tempVertices.push((xc - x) / 600);
56.      tempVertices.push((yc + y) / 600);
57.      tempVertices.push(0);
58.
59.      tempVertices.push((xc - x) / 600);
60.      tempVertices.push((yc - y) / 600);
61.      tempVertices.push(0);
62.
63.      tempVertices.push((xc + x) / 600);
64.      tempVertices.push((yc - y) / 600);
65.      tempVertices.push(0);
66.

```

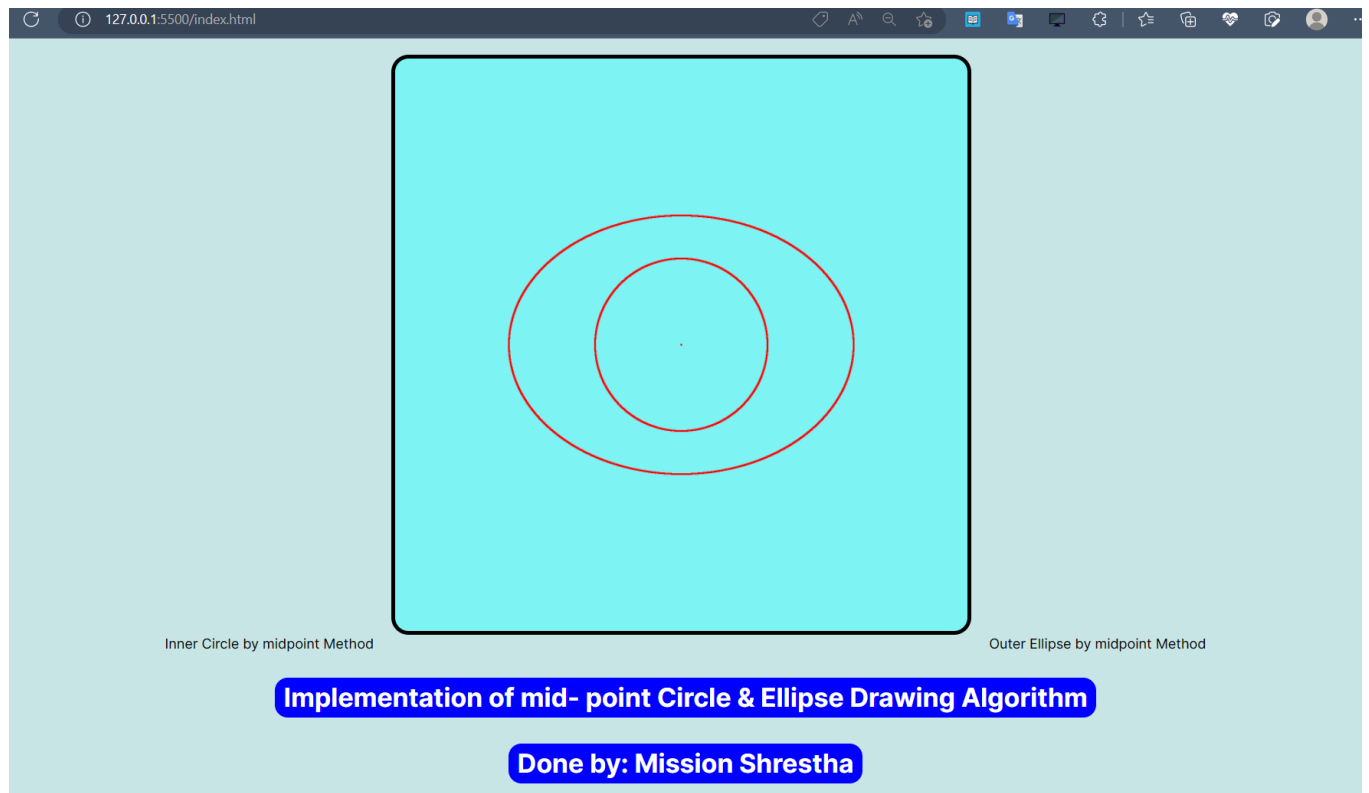
```
67.     return tempVertices;  
68. }  
69.
```

```

1 // This function takes in the parameters rx (radius along x-axis), ry (radius along y-axis), and the center point xc,yc
2 function midPointEllipse(rx, ry, xc, yc) {
3     let tempVertices = [];
4
5     // Initialize variables
6     let x = 0;
7     let y = ry;
8     let p = ry * ry - rx * rx * ry + (rx * rx) / 4;
9     let dx = 2 * ry * ry * x;
10    let dy = 2 * rx * rx * y;
11
12
13    tempVertices = [...tempVertices, ...otherVertex(x, y, xc, yc)];
14
15
16    // This loop calculates the vertices for the first half of the ellipse (Region-1)
17    while (2 * ry * ry * x < 2 * rx * rx * y) {
18        if (p < 0) {
19            x = x + 1;
20            y = y;
21            p = p + 2 * ry * ry * x + ry * ry;
22        } else {
23            x = x + 1;
24            y = y - 1;
25            p = p + 2 * ry * ry * x - 2 * rx * rx * y + ry * ry;
26        }
27        tempVertices = [...tempVertices, ...otherVertex(x, y, xc, yc)];
28    }
29
30    // Calculate the value of p for the second half of the ellipse (Region-2)
31    p = ry * ry * (x + (1 / 2)) * (x + (1 / 2)) + rx * rx * (y - 1) * (y - 1) - rx * rx * ry * ry;
32    // This loop calculates the vertices for the second half of the ellipse (Region-2)
33    while (y > 0) {
34        if (p > 0) {
35            y = y - 1;
36            p = p + rx * rx - 2 * rx * rx * y;
37        } else {
38            x = x + 1;
39            y = y - 1;
40            p = p + 2 * ry * ry * x - 2 * rx * rx * y + rx * rx;
41        }
42        tempVertices = [...tempVertices, ...otherVertex(x, y, xc, yc)];
43    }
44
45    return tempVertices;
46 }
47
48 function otherVertex(x, y, xc, yc) {
49     let tempVertices = [];
50
51     tempVertices.push((x + xc) / 600);
52     tempVertices.push((y + yc) / 600);
53     tempVertices.push(0);
54
55     tempVertices.push((xc - x) / 600);
56     tempVertices.push((yc + y) / 600);
57     tempVertices.push(0);
58
59     tempVertices.push((xc - x) / 600);
60     tempVertices.push((yc - y) / 600);
61     tempVertices.push(0);
62
63     tempVertices.push((xc + x) / 600);
64     tempVertices.push((yc - y) / 600);
65     tempVertices.push(0);
66
67     return tempVertices;
68 }
69

```


Output:



Conclusion:

Hence, midpoint algorithm was used to draw circle & ellipse.