DEERWALK INSTITUTE OF TECHNOLOGY

Tribhuvan University

Faculties of Computer Science

Bachelors of Science in Computer Science and Information Technology (BSc. CSIT)

Course: Computer Graphics (CSC209)

Year/Semester: II/III

A Lab report on:

Implementation of Mid-Point Ellipse Drawing Algorithm

Submitted by:
Name: Arun Mainali
Roll: 1307

Submitted to:
Binod Sitaula
Department of Computer Science

❖ LAB 4

OBJECTIVE:

Implementation of Midpoint Ellipse Drawing Algorithm

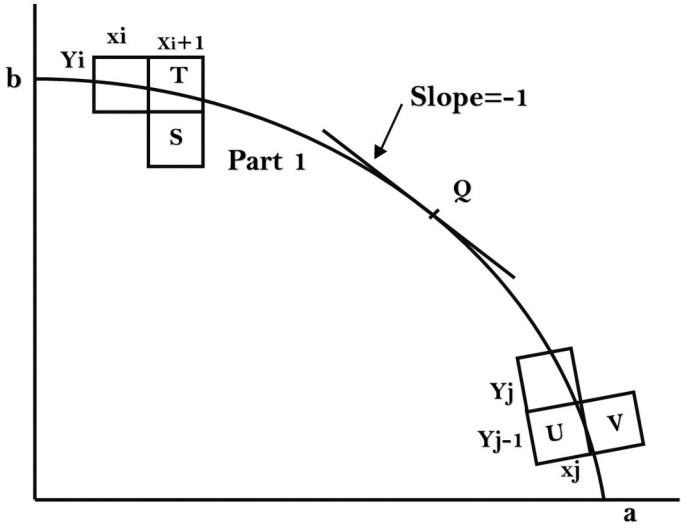
THEORY:

Midpoint Ellipse Drawing Algorithm:

This is an incremental method for scan converting an ellipse that is centered at the origin in standard position i.e., with the major and minor axis parallel to coordinate system axis. It is very similar to the midpoint circle algorithm. Because of the four-way symmetry property we need to consider the entire elliptical curve in the first quadrant.

Let's first rewrite the ellipse equation and define the function f that can be used to decide if the midpoint between two candidate pixels is inside or outside the ellipse:

$$f(x, y) = b^2x^2 + a^2y^2 - a^2b^2 = \begin{bmatrix} < 0 (x,y) \text{inside} \\ o\ (x,y) \text{on} \\ > 0\ (x,y) \text{outside} \end{bmatrix}$$



Now divide the elliptical curve from (0, b) to (a, 0) into two parts at point Q where the slope of the curve is -1.

Slope of the curve is defined by the f(x, y) = 0 is $\frac{dy}{dx} = -\frac{fx}{fy}$ where fx & fy are partial derivatives of f(x, y) with respect to x & y.

We have $fx = 2b^2 x$, $fy = 2a^2 y$ & $\frac{dy}{dx} = -\frac{2b^2x}{2a^2y}$ Hence we can monitor the slope value during the scan conversion process to detect Q. Our starting point is (0, b)

Suppose that the coordinates of the last scan converted pixel upon entering step i are (x_i, y_i). We are to select either T $(x_{i+1}), y_i)$ or S (x_{i+1}, y_{i-1}) to be the next pixel. The midpoint of T & S is used to define the following decision parameter.

$pi = f(x_{i+1}), y_i - \frac{1}{2})$

$pi = b^2 (x_{i+1})^2 + a^2 (y_i - \frac{1}{2})^2 - a^2 b^2$

If pi<0, the midpoint is inside the curve and we choose pixel T.

If pi>0, the midpoint is outside or on the curve and we choose pixel S.

Decision parameter for the next step is:

$$p_{i+1}=f(x_{i+1}+1,y_{i+1}-\frac{1}{2})$$
$$= b^2 (x_{i+1}+1)^2+a^2 (y_{i+1}-\frac{1}{2})^2-a^2 b^2$$

Since x_{i+1}=xi+1,we have

$$p_{i+1}-pi=b^2[((x_{i+1}+1)^2+a^2 (y_{i+1}-\frac{1}{2})^2-(y_i -\frac{1}{2})^2]$$
$$p_{i+1}= pi+2b^2 x_{i+1}+b^2+a^2 [(y_{i+1}-\frac{1}{2})^2-(y_i -\frac{1}{2})^2]$$

If T is chosen pixel (pi<0), we have y_{i+1}=yi.

If S is chosen pixel (pi>0) we have y_{i+1}=yi-1. Thus we can express p_{i+1}in terms of pi and (x_{i+1},y_{i+1}): $p_{i+1}= pi+2b^2 x_{i+1}+b^2$ if pi<0 = $pi+2b^2 x_{i+1}+b^2-2a^2 y_{i+1}$ if pi>0

The initial value for the recursive expression can be obtained by the evaluating the original definition of pi with (0, b):

$$p1 = (b^2+a^2 (b-\frac{1}{2})^2-a^2 b^2$$
$$= b^2-a^2 b+a^2/4$$

Suppose the pixel $(x_j\ y_j)$ has just been scan converted upon entering step j. The next pixel is either U (x_j ,y_j-1) or V (x_j+1,y_j-1). The midpoint of the horizontal line connecting U & V is used to define the decision parameter:

$$q_j=f(x_j+\frac{1}{2},y_j-1)$$
$$q_j=b^2 (x_j+\frac{1}{2})^2+a^2 (y_j -1)^2-a^2 b^2$$

If q_j<0, the midpoint is inside the curve and we choose pixel V.

If q_j≥0, the midpoint is outside the curve and we choose pixel U.Decision parameter for the next step is:

$$q_{j+1}=f(x_{j+1}+\frac{1}{2},y_{j+1}-1)$$
$$= b^2 (x_{j+1}+\frac{1}{2})^2+ a^2 (y_{j+1}-1)^2- a^2 b^2$$

Since $y_{j+1}=y_j-1$,we have

$q_{j+1}-q_j=b^2\ [(x_{j+1}+\frac{1}{2})^2-(x_j+\frac{1}{2})^2\ ]+a^2\ (y_{j+1}-1)^2-(\ y_{j+1})^2\ ]$

$q_{j+1}=q_j+b^2\ [(x_{j+1}+\frac{1}{2})^2-(x_j+\frac{1}{2})^2]-2a^2\ y_{j+1}+a^2$

If V is chosen pixel (qj<0), we have $x_{j+1}=xj$.

If U is chosen pixel (pi>0) we have $x_{j+1}=xj$. Thus we can express

q_{j+1}in terms of q_j and (x_{j+1},y_{j+1}):

$q_{j+1}=q_j+2b^2\ x_{j+1}-2a^2\ y_{j+1}+a^2$ if qj < 0

$=q_j-2a^2\ y_{j+1}+a^2$ if qj>0

The initial value for the recursive expression is computed using the original definition of qj. And the coordinates of $(x_k\ y_k)$ of the last pixel choosen for the part 1 of the curve:

$q1 = f(x_k+\frac{1}{2},y_k-1)=b^2\ (x_k+\frac{1}{2})^2-a^2\ (y_k-1)^2-\ a^2\ b^2$

ALGORITHM:

Step 1: Take input radius along x axis and y axis and obtain center of ellipse.

Step 2: Initially, we assume ellipse to be centered at origin and the first point as : $(x,\ y_0)=(0,\ r_y)$.

Step 3: Obtain the initial decision parameter for region 1 as: $p1_0=r_y^2+1/4r_x^2-r_x\ ^2r_y$

Step 4: For every x_k position in region 1 : If $p1_k<0$ then the next point along the is $(x_{k+1}\ ,\ y_k)$ and $p1_{k+1}=p1_k+2r_y^2x_{k+1}+r_y^2$ Else, the next point is $(x_{k+1},\ y_{k-1}\ )$
And $p1_{k+1}=p1_k+2r_y^2x_{k+1}\ -\ 2r_x^2y_{k+1}+r_y^2$

Step 5: Obtain the initial value in region 2 using the last point $(x_0,\ y_0)$ of region 1 as: $p2_0=r_y^2(x_0+1/2)^2+r_x^2\ (y_0-1)^2-r_x^2r_y^2$

Step 6: At each y_k in region 2 starting at k =0 perform the following task. If $p2_k>0$ the next point is $(x_k,\ y_{k-1})$ and $p2_{k+1}=p2_k-2r_x^2y_{k+1}+r_x^2$

Step 7: Else, the next point is $(x_{k+1},\ y_{k\ -1})$ and $p2_{k+1}=p2_k+2r_y^2x_{k+1}\ -2r_x^2y_{k+1}+r_x^2$

Step 8: Now obtain the symmetric points in the three quadrants and plot the coordinate value as: x=x+xc, y=y+yc

Step 9: Repeat the steps for region 1 until $2r_y^2x>=2r_x^2y$

Step 10: End

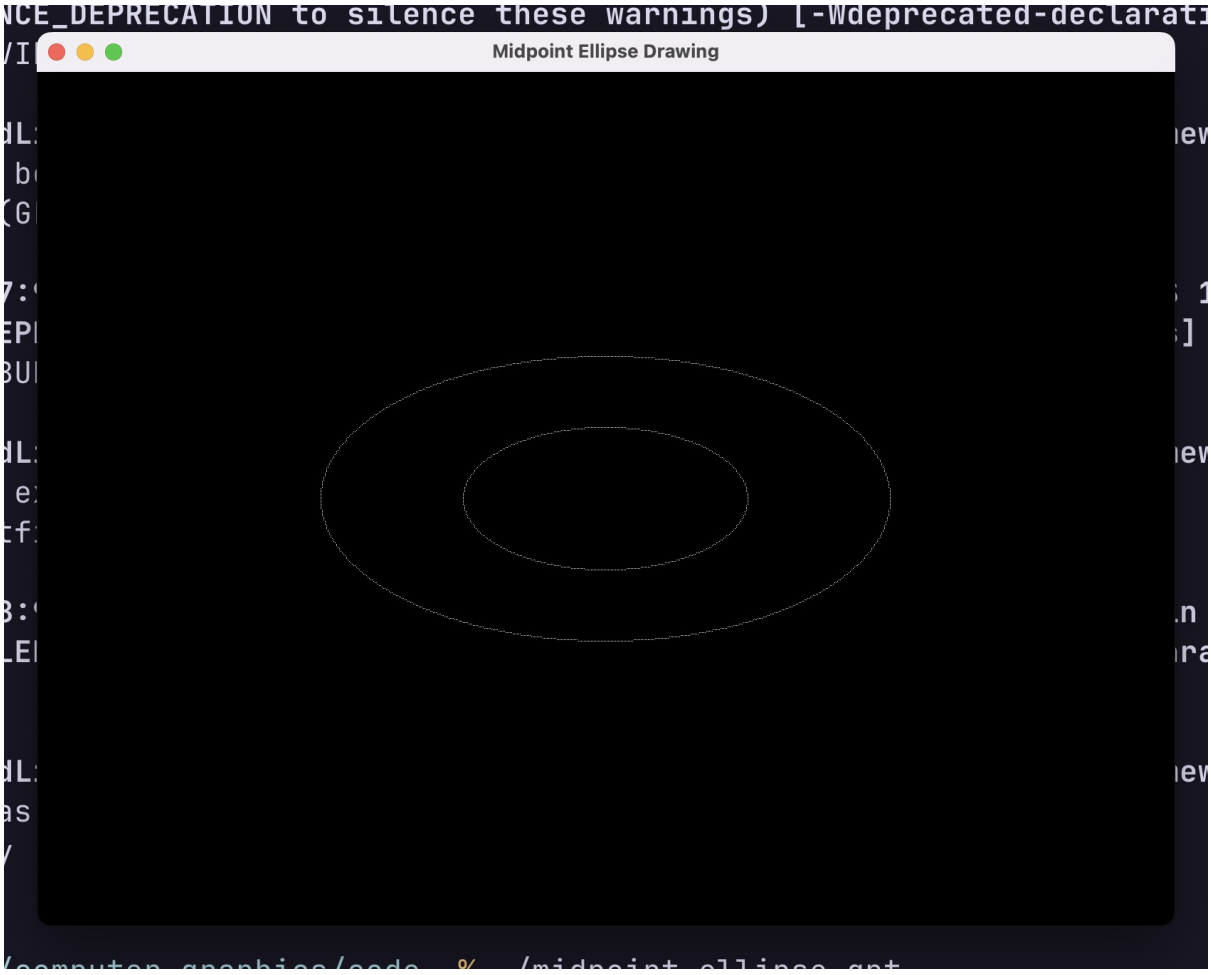
PROGRAM CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <GLFW/glfw3.h>

const int WIDTH = 800, HEIGHT = 600;

void putpixels(GLFWwindow* window, float x, float y) {
   float x_ndc = (2.0f * x) / WIDTH - 1.0f;
   float y_ndc = 1.0f - (2.0f * y) / HEIGHT;

   glBegin(GL_POINTS);
   glVertex2f(x_ndc, y_ndc);
   glEnd();
}

void midPointEllipseDraw(GLFWwindow* window, int xc, int yc, int rx, int ry) {
   float dx, dy, d1, d2, x, y;
   x = 0;
   y = ry;

   d1 = (ry * ry) - (rx * rx * ry) + (0.25 * rx * rx);
   dx = 2 * ry * ry * x;
   dy = 2 * rx * rx * y;

   while (dx < dy) {
      putpixels(window, xc + x, yc + y);
      putpixels(window, xc - x, yc + y);
      putpixels(window, xc + x, yc - y);
      putpixels(window, xc - x, yc - y);

      if (d1 < 0) {
         x++;
         dx += 2 * ry * ry;
         d1 += dx + ry * ry;
      } else {
         x++;
         y--;
         dx += 2 * ry * ry;
         dy -= 2 * rx * rx;
         d1 += dx - dy + ry * ry;
      }
   }

   d2 = (ry * ry) * (x + 0.5) * (x + 0.5) + (rx * rx) * (y - 1) * (y - 1) - (rx * rx * ry * ry);
```

```
    while (y >= 0) {
        putpixels(window, xc + x, yc + y);
        putpixels(window, xc - x, yc + y);
        putpixels(window, xc + x, yc - y);
        putpixels(window, xc - x, yc - y);

        if (d2 > 0) {
            y--;
            dy -= 2 * rx * rx;
            d2 += rx * rx - dy;
        } else {
            y--;
            x++;
            dx += 2 * ry * ry;
            dy -= 2 * rx * rx;
            d2 += dx - dy + rx * rx;
        }
    }
}

int main() {
    if (!glfwInit()) {
        printf("GLFW initialization failed!\n");
        return -1;
    }

    GLFWwindow* window = glfwCreateWindow(WIDTH, HEIGHT, "Midpoint Ellipse
Drawing", NULL, NULL);
    if (!window) {
        printf("Window creation failed!\n");
        glfwTerminate();
        return -1;
    }
    glfwMakeContextCurrent(window);

    glViewport(0, 0, WIDTH, HEIGHT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-1, 1, -1, 1, -1, 1);
    glMatrixMode(GL_MODELVIEW);

    while (!glfwWindowShouldClose(window)) {
        glClear(GL_COLOR_BUFFER_BIT);
        glLoadIdentity();

        midPointEllipseDraw(window, WIDTH / 2, HEIGHT / 2, 200, 100);
        midPointEllipseDraw(window, WIDTH / 2, HEIGHT / 2, 100, 50);
```

```
        glfwSwapBuffers(window);
        glfwPollEvents();
    }

    glfwDestroyWindow(window);
    glfwTerminate();
    return 0;
}
```

SAMPLE OUTPUT:



CONCLUSION:

From this lab, we were able to draw an ellipse using Midpoint Ellipse Drawing Algorithm and display it using C graphic functions.