CS352 Assignment-4

Krishanu Saini
190001029

Write a program using glut library to draw the circle by implementing the following algorithms:

1. Bresenham's circle drawing algorithm

2. Midpoint circle drawing algorithm

Problem-1

Bresenham circle drawing algorithm.

We solve the problem by dividing the circle into 8 octants and just iterating in the 1st octant.
We start at x=0, y=r (wrt center coordinates). The decision parameter p = 3 - 2r.
Xi+1 = xi+1
If p > 0: yi+1 = yi - 1, p = p + 4*(xi-yi) + 10
Else p < 0: yi+1 = yi, p = p + 4*xi + 6
We iterate till x < y

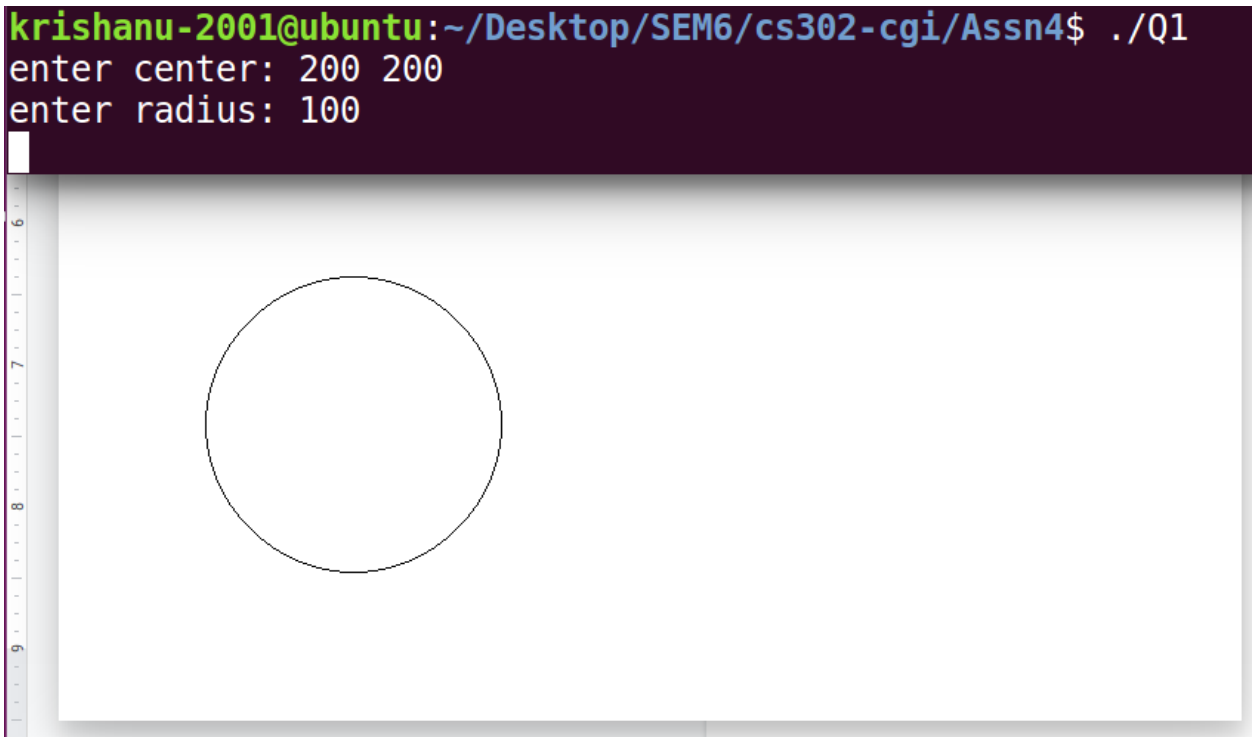
Code

```
#include <math.h>
#include <GL/glut.h>
#include <iostream>
#include <cassert>
#define debug(x,y) cout << #x << " " << x << " " << #y << " " << y << "\n";

using namespace std;

/*
 * Name: Krishanu Saini
 * Roll: 190001029
 * Ques: Problem 1) Bresenham circle
 * Date: 2/02/22
 */

int xcord1,ycord1;
int r;

void BresenhamCircle(int x1, int y1, int r) {
   glColor3f(0,0,0);

   int p = 3 - 2*r;
   int xi = 0, yi = r;
   // innitialize parameters
   // if d > 0 then d = d + 4 * (x - y) + 10; and go down
   // else d < 0 then d = d + 4 * x + 6; and stay
   glBegin(GL_POINTS);

   /*-------- draw all 8 octants --------*/
   glVertex2i(x1+xi, y1+yi);
```

```
    glVertex2i(x1-xi, y1+yi);
    glVertex2i(x1+xi, y1-yi);
    glVertex2i(x1-xi, y1-yi);
    glVertex2i(x1+yi, y1+xi);
    glVertex2i(x1-yi, y1+xi);
    glVertex2i(x1+yi, y1-xi);
    glVertex2i(x1-yi, y1-xi);

    while(xi <= yi) {
        xi = xi+1;
        if(p > 0) {
            // go down
            p = p+4*(xi-yi)+10;
            yi=yi-1;
        } else if(p <= 0) {
            // stay
            p = p+4*xi+6;
        }
        /*-------- draw all 8 octants --------*/
        glVertex2i(x1+xi, y1+yi);
        glVertex2i(x1-xi, y1+yi);
        glVertex2i(x1+xi, y1-yi);
        glVertex2i(x1-xi, y1-yi);
        glVertex2i(x1+yi, y1+xi);
        glVertex2i(x1-yi, y1+xi);
        glVertex2i(x1+yi, y1-xi);
        glVertex2i(x1-yi, y1-xi);
    }

    glEnd();
}

void Draw() {
    glClear(GL_COLOR_BUFFER_BIT);
    glPointSize(1);

    BresenhamCircle(xcord1,ycord1,r);

    glFlush();
}

int main(int argc, char *argv[]) {
    printf("enter center: ");
    cin >> xcord1 >> ycord1;
```

```
    printf("enter radius: ");
    cin >> r;
    glutInit(&argc, argv);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(800, 800);
    glutInitDisplayMode(GLUT_RGB);

    glutCreateWindow("Bresenham Circle");
    gluOrtho2D(0, 800, 0, 800);
    glClearColor(1,1,1,0.0);
    glutDisplayFunc(Draw);
    glutMainLoop();
    return 0;
}
```

Output



Problem-2

Mid point circle drawing algorithm.

We choose 2 points East and S East, we check if the distance between the center and circle is

1. d < 0 - inside the circle, yi+1 = yi, d = d + 2*xi + 1;
2. d >= 0 - outside or boundary of the circle, yi+1 = yi - 1, d = d - 2*yi + 2*xi + 1;

We initialize d with 5/4 - r.
no floating calculation is needed as we do approximation 1.25 to 1.

Code

```
#include <math.h>
#include <GL/glut.h>
#include <iostream>
#include <cassert>
#define debug(x,y) cout << #x << " " << x << " " << #y << " " << y << "\n";

using namespace std;

/*
 * Name: Krishanu Saini
 * Roll: 190001029
 * Ques: Problem 2) Mid Point circle
 * Date: 2/02/22
 */

float xcord1,ycord1;
float r;

void MidCircle(float x1, float y1, float r) {
   glColor3f(0,0,0);

   int xi = 0, yi = r;
   int p = 1-r;

   glBegin(GL_POINTS);

   /*-------- draw all 8 octants --------*/
   glVertex2i(x1+xi, y1+yi);
   glVertex2i(x1-xi, y1+yi);
   glVertex2i(x1+xi, y1-yi);
   glVertex2i(x1-xi, y1-yi);
   glVertex2i(x1+yi, y1+xi);
```

```
    glVertex2i(x1-yi, y1+xi);
    glVertex2i(x1+yi, y1-xi);
    glVertex2i(x1-yi, y1-xi);

    while(xi <= yi) {
        xi++;
        // increment x normally
        if(p < 0) {
            // point lies inside circle - stay at level
            p = p + 2*xi + 1;
        } else {
            // go down
            yi--;
            p = p - 2*yi + 2*xi + 1;
        }
        /*-------- draw all 8 octants --------*/
        glVertex2i(x1+xi, y1+yi);
        glVertex2i(x1-xi, y1+yi);
        glVertex2i(x1+xi, y1-yi);
        glVertex2i(x1-xi, y1-yi);
        glVertex2i(x1+yi, y1+xi);
        glVertex2i(x1-yi, y1+xi);
        glVertex2i(x1+yi, y1-xi);
        glVertex2i(x1-yi, y1-xi);
    }

    glEnd();
}

void Draw() {
    glClear(GL_COLOR_BUFFER_BIT);
    glPointSize(1);

    MidCircle(xcord1,ycord1,r);

    glFlush();
}

int main(int argc, char *argv[]) {
    printf("enter center: ");
    cin >> xcord1 >> ycord1;
    printf("enter radius: ");
    cin >> r;
    glutInit(&argc, argv);
```

```
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(800, 800);
    glutInitDisplayMode(GLUT_RGB);

    glutCreateWindow("Mid Point Circle");
    gluOrtho2D(0, 800, 0, 800);
    glClearColor(1,1,1,0.0);
    glutDisplayFunc(Draw);
    glutMainLoop();
    return 0;
}
```

Output