Practical no:3

Title:Implement Bresenham circle drawing algorithm to draw any object. The object should be displayed in all the quadrants with respect to center and radius.

Name:Ujawala Sinha
Roll no:S554

Code:

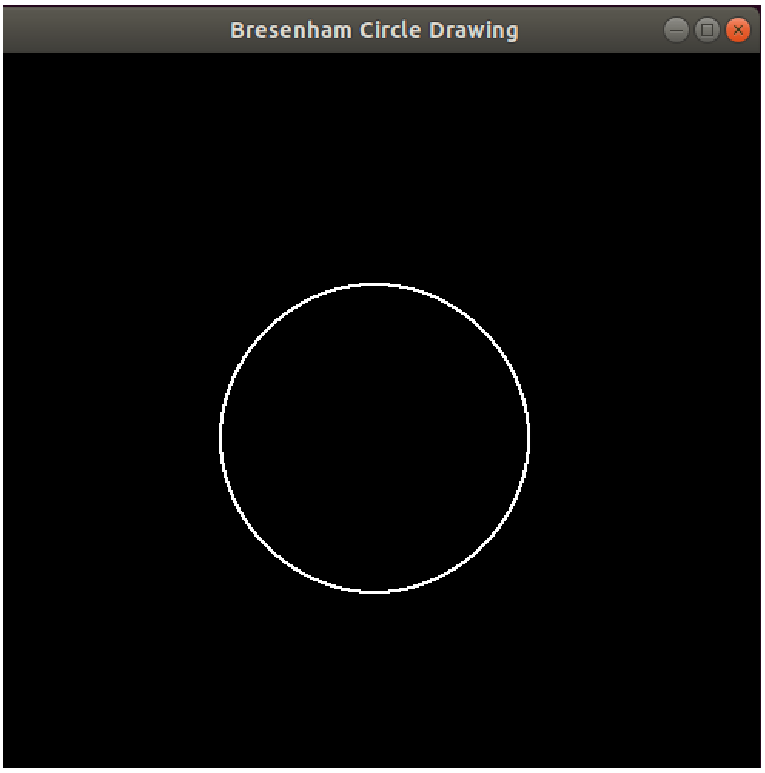
Simple circle

```
#include <GL/glut.h>
#include <iostream>

using namespace std;

// Default circle center and radius
int centerX = 250, centerY = 250, radius = 100;
bool useMouse = false;

// Function to plot 8 symmetrical points of a circle
void plotPoints(int x, int y, int xc, int yc) {
    glBegin(GL_POINTS);
    glVertex2i(xc + x, yc + y);
    glVertex2i(xc - x, yc + y);
    glVertex2i(xc + x, yc - y);
    glVertex2i(xc - x, yc - y);
    glVertex2i(xc + y, yc + x);
    glVertex2i(xc - y, yc + x);
    glVertex2i(xc + y, yc - x);
    glVertex2i(xc - y, yc - x);
    glEnd();
}

// Bresenham's circle drawing algorithm
void drawCircle(int xc, int yc, int r) {
    int x = 0, y = r;
    int d = 3 - 2 * r;

    while (x <= y) {
        plotPoints(x, y, xc, yc);
        x++;
        if (d < 0) {
            d = d + 4 * x + 6;
        } else {
            y--;
            d = d + 4 * (x - y) + 10;
        }
    }
    glFlush();
}
```

```
// Display function
void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    drawCircle(centerX, centerY, radius);  // Default circle
}

// Mouse function to draw a circle on click
void mouse(int button, int state, int x, int y) {
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
        centerX = x;
        centerY = 500 - y;  // Inverting y-axis for OpenGL
        useMouse = true;
        glutPostRedisplay();
    }
}

// OpenGL initialization
void init() {
    glClearColor(0.0, 0.0, 0.0, 1.0);
    glColor3f(1.0, 1.0, 1.0);
    glPointSize(2.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, 500, 0, 500);
}

// Main function
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Bresenham Circle Drawing");

    init();
    glutDisplayFunc(display);
    glutMouseFunc(mouse);
    glutMainLoop();
    return 0;
}
```

output:

```
svpm@svpm-HP-EliteDesk-800-G2-SFF:~$ g++ circlesimple.cpp -lGL -lGLU -lglut
svpm@svpm-HP-EliteDesk-800-G2-SFF:~$ ./a.out
```



Mouse click bresenham Circle

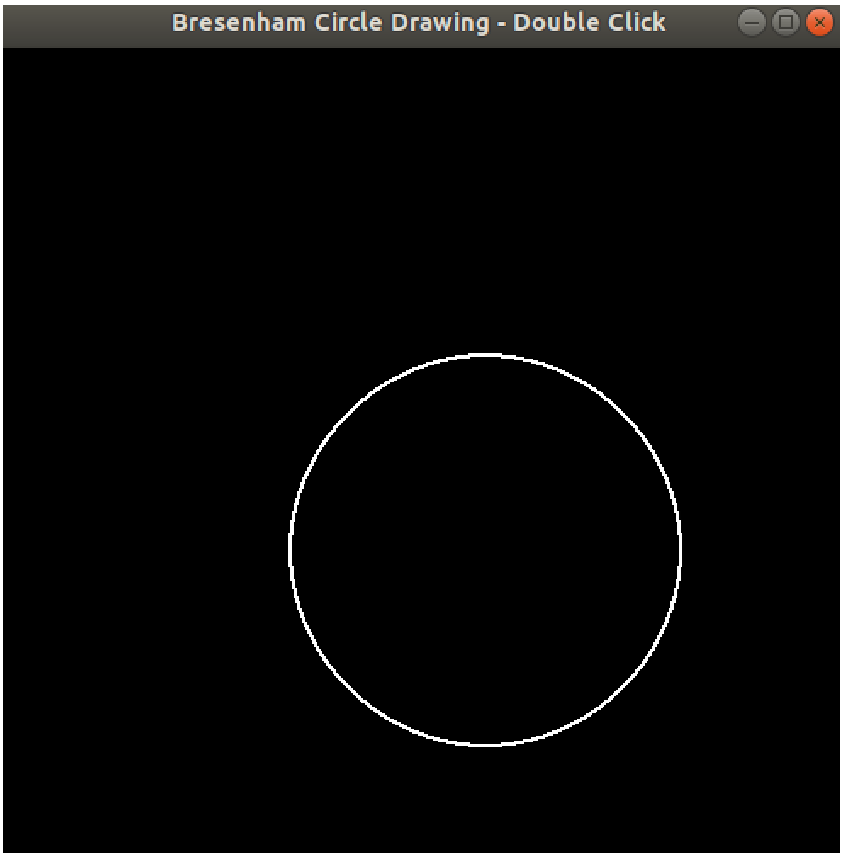
```
#include <GL/glut.h>
#include <cmath>
#include <iostream>

using namespace std;

int centerX = -1, centerY = -1; // Center of the circle (set by first click)
int radius = 0;                 // Radius of the circle (set by second click)
bool firstClick = true;         // Flag to track clicks

// Function to plot 8 symmetric points of a circle
void plotPoints(int x, int y, int xc, int yc) {
    glBegin(GL_POINTS);
    glVertex2i(xc + x, yc + y);
    glVertex2i(xc - x, yc + y);
    glVertex2i(xc + x, yc - y);
    glVertex2i(xc - x, yc - y);
    glVertex2i(xc + y, yc + x);
    glVertex2i(xc - y, yc + x);
    glVertex2i(xc + y, yc - x);
    glVertex2i(xc - y, yc - x);
    glEnd();
}

// Bresenham's Circle Drawing Algorithm
```

```
void drawCircle(int xc, int yc, int r) {
   int x = 0, y = r;
   int d = 3 - 2 * r;

   while (x <= y) {
      plotPoints(x, y, xc, yc);
      x++;
      if (d < 0) {
         d = d + 4 * x + 6;
      } else {
         y--;
         d = d + 4 * (x - y) + 10;
      }
   }
   glFlush();
}

// Mouse Click Function
void mouse(int button, int state, int x, int y) {
   if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
      if (firstClick) {
         // First click: Set the center
         centerX = x;
         centerY = 500 - y; // Convert from OpenGL coordinates
         firstClick = false;
         cout << "Center set at: (" << centerX << ", " << centerY << ")\n";
      } else {
         // Second click: Calculate the radius and draw the circle
         int dx = x - centerX;
         int dy = (500 - y) - centerY;
         radius = sqrt(dx * dx + dy * dy);
         cout << "Radius set: " << radius << "\n";
         glClear(GL_COLOR_BUFFER_BIT);
         drawCircle(centerX, centerY, radius);
         firstClick = true; // Reset for next circle
      }
   }
}

// OpenGL Initialization
void init() {
   glClearColor(0.0, 0.0, 0.0, 1.0);
   glColor3f(1.0, 1.0, 1.0);
   glPointSize(2.0);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(0, 500, 0, 500);
}

// Display Function
void display() {
   glClear(GL_COLOR_BUFFER_BIT);
```

```
    glFlush();
}

// Main Function
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Bresenham Circle Drawing – double click");

    init();
    glutDisplayFunc(display);
    glutMouseFunc(mouse);
    glutMainLoop();
    return 0;
}
```

Output:

```
svpm@svpm-HP-EliteDesk-800-G2-SFF:~$ g++ breshcircle.cpp -lGL -lGLU -lglut
svpm@svpm-HP-EliteDesk-800-G2-SFF:~$ ./a.out
Center set at: (289, 204)
Radius set: 115
```


Bresenham Circle Drawing - Double Click