Practical No.:3

Title: Draw a Circle.
Name: Samruddhi Sangram Yadav.
Roll No.:S564

Simple circle drawing Code:

```
#include<GL/glut.h>
#include<iostream>
using namespace std;
int radius;
void MyInit()
{
   glClearColor(1,1,1,1);
   glColor3f(1,0,0);
   glOrtho(-500,500,500,-500,0,1);
}
void drawPoints(int x,int y)
{
   glBegin(GL_POINTS);
      glVertex2i(x,y);
      glVertex2i(-x,y);
      glVertex2i(x,-y);
      glVertex2i(-x,-y);
       glVertex2i(y,x);
      glVertex2i(-y,x);
      glVertex2i(y,-x);
      glVertex2i(-y,-x);
   glEnd();
}
void Circle()
{
   MyInit();
   int x=0, y=radius;
   drawPoints(x,y);
   int d = 3-2*radius;
   while(x<=y)
   {
      if(d < 0)
      {
         x = x+1;
         d = d + 4*x +6;
      }
      else if(d>=0)
      {
         x = x+1;
         y = y-1;
         d = d + 4*(x-y) + 10;
      }

      drawPoints(x,y);
```

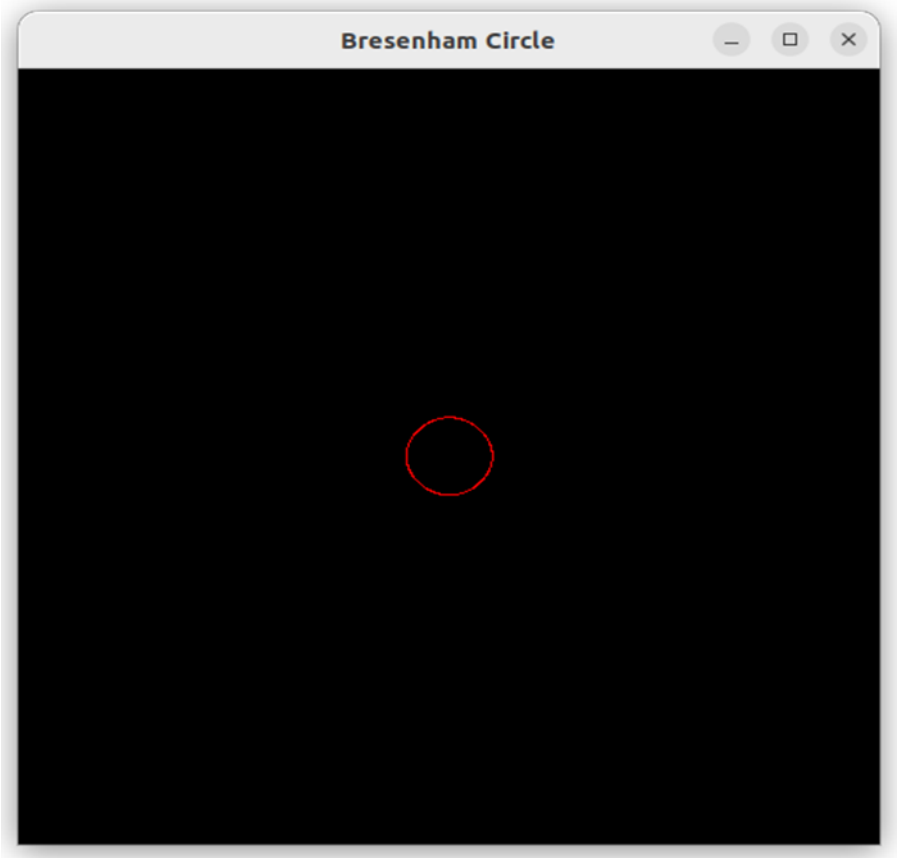
```
    }
     glFlush();
}
int main(int argc,char **argv)
{
   glutInit(&argc,argv);
   glutInitWindowSize(500,500);
   glutInitWindowPosition(100,100);
   glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
   glutCreateWindow("Bresenham Circle");

   cout<<"Enter the radius of Circle:";
   cin>>radius;
   glutDisplayFunc(Circle);

   glutMainLoop();
   return 0;
}
```

Output:

```
it@it-HP-EliteDesk-800-G2-SFF:~$ g++ BreCir.cpp -lGL -lGLU -lglut -lm
it@it-HP-EliteDesk-800-G2-SFF:~$ ./a.out
Enter the radius of Circle:50
```



Bresenham Circle with Mouse Click

```
#include <GL/glut.h>
#include <cmath>
// Global variables to store the center, radius, and the second mouse click coordinates
int centerX = 0, centerY = 0, radius = 0;
int secondClickX = 0, secondClickY = 0;
bool isFirstClick = true;
// Function to plot points using the symmetry of the circle
void plotCirclePoints(int h, int k, int x, int y) {
    glBegin(GL_POINTS);
    glVertex2i(h + x, k + y);
    glVertex2i(h - x, k + y);
    glVertex2i(h + x, k - y);
    glVertex2i(h - x, k - y);
    glVertex2i(h + y, k + x);
    glVertex2i(h - y, k + x);
    glVertex2i(h + y, k - x);
    glVertex2i(h - y, k - x);
    glEnd();
}
// Function to calculate the radius based on the distance between two points
int calculateRadius(int x1, int y1, int x2, int y2) {
    return static_cast<int>(sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1)));
}
// Bresenham's Circle Algorithm to draw a circle
void drawCircle(int h, int k, int r) {
    int x = 0, y = r;
    int p = 3 - 2 * r;  // Initial decision parameter
    plotCirclePoints(h, k, x, y);
    while (x <= y) {
        x++;
        // Update the decision parameter and move the point
        if (p < 0) {
            p = p + 4 * x + 6;  // Move horizontally
        } else {
            y--;
            p = p + 4 * (x - y) + 10;
        }
        // Plot the points using symmetry
        plotCirclePoints(h, k, x, y);
    }
}

// Function to initialize OpenGL settings
void initOpenGL() {
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);  // Set background color to white
    glPointSize(2.0);  // Set point size for better visibility
}
// Function to display the scene
void display() {
```

```
    glClear(GL_COLOR_BUFFER_BIT);  // Clear the screen
    glColor3f(1.0f, 0.0f, 0.0f);  // Set color to red
    // Draw the circle with the current center and radius
    if (radius > 0) {
        drawCircle(centerX, centerY, radius);
    }
    glutSwapBuffers();  // Swap buffers for double buffering
}
// Function to handle mouse click events
void mouseClick(int button, int state, int x, int y) {
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
        int adjustedX = x - 250;  // Adjust x-coordinate to OpenGL coordinate system
        int adjustedY = 250 - y;  // Adjust y-coordinate to OpenGL coordinate system
        if (isFirstClick) {
            // First click sets the center
            centerX = adjustedX;
            centerY = adjustedY;
            isFirstClick = false;  // Switch to second click mode
        } else {
            // Second click sets the radius
            secondClickX = adjustedX;
            secondClickY = adjustedY;
            radius = calculateRadius(centerX, centerY, secondClickX, secondClickY);
            isFirstClick = true;  // Reset to first click mode for next circle
        }
        glutPostRedisplay();  // Redraw the window with updated circle
    }
}
// Function to set up the OpenGL viewport and projection
void setupViewport() {
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-250, 250, -250, 250, -1, 1);  // Set orthogonal projection
    glMatrixMode(GL_MODELVIEW);
}

// Main function to initialize GLUT and start the program
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Bresenham Circle with Mouse Click");
    initOpenGL();
    setupViewport();
    glutDisplayFunc(display);
    glutMouseFunc(mouseClick);  // Mouse click event to set center and radius
    glutMainLoop();  // Start the main loop
    return 0;
}
```

Output:

```
it@it-HP-EliteDesk-800-G2-SFF:~$ g++ c5.cpp -lGL -lGLU -lglut -lm
it@it-HP-EliteDesk-800-G2-SFF:~$ ./a.out
```