**Shudhanshu Shrotriya**

**500086221**

**Batch-4**

**Experiment-6**

**Computer Graphics Lab**

**Bresenham’s Circle Drawing Algorithm**

**Bresenham's Circle Algorithm:**

**Step1:** Start Algorithm

**Step2:** Declare p, q, x, y, r, d variables  
        p, q are coordinates of the center of the circle  
        r is the radius of the circle

**Step3:** Enter the value of r

**Step4:** Calculate d = 3 - 2r

**Step5:** Initialize       x=0  
          &nbsy= r

**Step6:** Check if the whole circle is scan converted  
            If x > = y  
            Stop

**Step7:** Plot eight points by using concepts of eight-way symmetry. The center is at (p, q). Current active pixel is (x, y).  
                glVertex2i (x+p, y+q)  
                glVertex2i (y+p, x+q)  
                glVertex2i (-y+p, x+q)  
                glVertex2i (-x+p, y+q)  
                glVertex2i (-x+p, -y+q)  
                glVertex2i (-y+p, -x+q)  
                glVertex2i (y+p, -x+q)  
                glVertex2i (x+p, -y-q)

**Step8:** Find location of next pixels to be scanned  
            If d < 0  
            then d = d + 4x + 6  
            increment x = x + 1  
            If d ≥ 0  
            then d = d + 4 (x - y) + 10  
            increment x = x + 1  
            decrement y = y - 1

**Step9:** Go to step 6

**Step10:** Stop Algorithm

**CODE:**

#include <stdio.h>

#include <math.h>

#include <GL/glut.h>

// Center of the cicle = (320, 240)

int xc = 320, yc = 240;

// Plot eight points using circle's symmetrical property

void plot\_point(int x, int y)

{

glBegin(GL\_POINTS);

glVertex2i(xc+x, yc+y);

glVertex2i(xc+x, yc-y);

glVertex2i(xc+y, yc+x);

glVertex2i(xc+y, yc-x);

glVertex2i(xc-x, yc-y);

glVertex2i(xc-y, yc-x);

glVertex2i(xc-x, yc+y);

glVertex2i(xc-y, yc+x);

glEnd();

}

// Function to draw a circle using bresenham's

// circle drawing algorithm

void bresenham\_circle(int r)

{

int x=0,y=r;

int pk= 3 - (2 \* r);

/\* Plot the points \*/

/\* Plot the first point \*/

plot\_point(x,y);

int k;

/\* Find all vertices till x=y \*/

while(x < y)

{

x = x + 1;

if(pk < 0)

pk = pk + 4\*x + 6;

else

{

y = y - 1;

pk = pk + 4\*(x - y) + 10;

}

plot\_point(x,y);

}

glFlush();

}

void a\_circle(void)

{

/\* Clears buffers to preset values \*/

glClear(GL\_COLOR\_BUFFER\_BIT);

int radius1 = 100, radius2 = 200;

bresenham\_circle(radius2);

}

void Init()

{

/\* Set clear color to white \*/

glClearColor(1.0,1.0,1.0,0);

/\* Set fill color to black \*/

glColor3f(0.0,0.0,0.0);

/\* glViewport(0 , 0 , 640 , 480); \*/

/\* glMatrixMode(GL\_PROJECTION); \*/

/\* glLoadIdentity(); \*/

gluOrtho2D(0 , 640 , 0 , 480);

}

int main(int argc, char \*\*argv)

{

/\* Initialise GLUT library \*/

glutInit(&argc,argv);

/\* Set the initial display mode \*/

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

/\* Set the initial window position and size \*/

glutInitWindowPosition(0,0);

glutInitWindowSize(640,480);

/\* Create the window with title "DDA\_Line" \*/

glutCreateWindow("bresenham\_circle");

/\* Initialize drawing colors \*/

Init();

/\* Call the displaying function \*/

glutDisplayFunc(a\_circle);

/\* Keep displaying untill the program is closed \*/

glutMainLoop();

}

**OUTPUT:**

