Graphics Assignment No:- 2

1. Bresenhams circle drawing algorithm

These algorithms are based on the idea of determining the subsequent points required to draw the circle.Let us discuss the algorithms in detail - We cannot display a continuous arc on the raster display. Instead, we have to choose the nearest

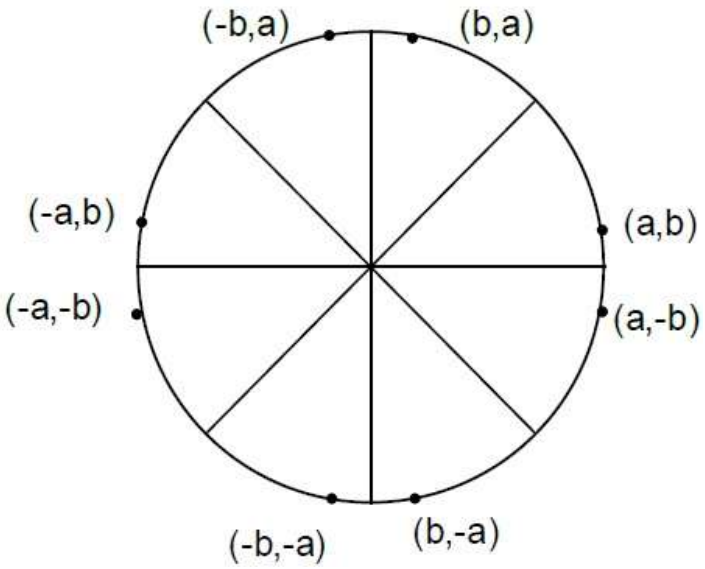

Figure 1: circle coordinate's

pixel position to complete the arc.From the following illustration, you can see that we have put the pixel at (X, Y) location and now need to decide where to put the next pixel at N (X+1, Y) or at S (X+1, Y-1).

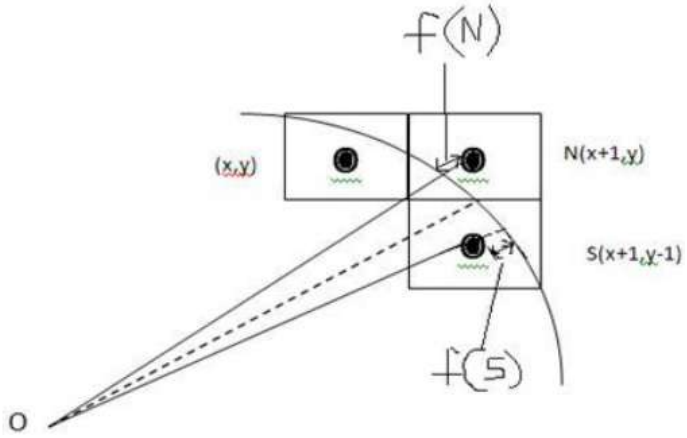

Figure 2: circle coordinate's

This can be decided by the decision parameter d.

1. If d ¡= 0, then N(X+1, Y) is to be chosen as next pixel.
2. If d ¿ 0, then S(X+1, Y-1) is to be chosen as the next pixel.

2. Algorithm

1. Step 1 Get the coordinates of the center of the circle and radius, and store them in x, y, and R respectively. Set P=0 and Q=R.

2. Step 2 Set decision parameter D = 3 2R.
3. Step 3 Repeat through step-8 while P Q.
4. Step 4 Call Draw Circle (X, Y, P, Q).
5. Step 5 Increment the value of P.
6. Step 6 If D ¡ 0 then D = D + 4P + 6.
7. Step 7 Else Set R = R - 1, D = D + 4(P-Q) + 10.

3. Code

step 1: C++ OpenGL code

```
// C-program for circle drawing
// using Bresenham s Algorithm
// in computer-graphics

#include <stdlib.h>
#include <iostream>
#include <fstream>
#ifdef __APPLE__
#include <GLUT/glut.h>
#include <OpenGL/gl.h>
#include <OpenGL/glu.h>
#else
#include <GL/glut.h>
#include <GL/glu.h>
#include <GL/gl.h>
#endif
#include <OpenGL/gl3.h>
#include <stdio.h>
#include <math.h>

// Center of the cicle = (320, 240)
int x_last = 310, y_last = 220;
// Function to draw a circle using bresenham's
// circle drawing algorithm
void bresenham_circle(int r);
void Bresenham_circle()
{
//      Clears buffers to preset values
    glClear(GL_COLOR_BUFFER_BIT);
    int radius1 =50;
    bresenham_circle(radius1);
}
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(1366, 768);
//      Create the window with title "DDA_Line"
    glutCreateWindow("bresenham_circle_agorithms");
//      Initialize drawing colors

//      Set clear color to white
    glClearColor(0.0,0.0,1.0,0);
//      Set fill color to black
    glColor3f(1.0,0.0,0.0);
    glPointSize(1.5);
//      glViewport(0 , 0 , 640 , 480);
//      glMatrixMode(GL_PROJECTION);
//      glLoadIdentity();
    gluOrtho2D(0 , 500 , 0 , 500);

//      Call the displaying function
    glutDisplayFunc(Bresenham_circle);
//      Keep displaying untill the program is closed
    glutMainLoop();
}
void bresenham_circle(int r)
{
    int x=0,y=r;
    float h=(5.0/4.0)-r;

    /* Plot the points */
    /* Plot the first point */
    glBegin(GL_POINTS);
    glVertex2i(x_last+x, y_last+y);
    glVertex2i(x_last+x, y_last-y);
    glVertex2i(x_last+y, y_last+x);
    glVertex2i(x_last+y, y_last-x);
    glVertex2i(x_last-x, y_last-y);
    glVertex2i(x_last-y, y_last-x);
    glVertex2i(x_last-x, y_last+y);
    glVertex2i(x_last-y, y_last+x);
    glEnd();

    /* Find all vertices till x=y */
    while(x < y)
    {
        x = x + 1;
        if(h < 0)
            h = h + 2*x+3;
        else
        {
            y = y - 1;
            h = h + 2*(x - y) + 5;
        }
        glBegin(GL_POINTS);
        glVertex2i(x_last+x, y_last+y);
        glVertex2i(x_last+x, y_last-y);
        glVertex2i(x_last+y, y_last+x);
        glVertex2i(x_last+y, y_last-x);
        glVertex2i(x_last-x, y_last-y);
```

```
            glVertex2i(x_last-y, y_last-x);
            glVertex2i(x_last-x, y_last+y);
            glVertex2i(x_last-y, y_last+x);
            glEnd();
        }
        glFlush();
}
```

4. Result

Circle using Bresenhams circle drawing algorithm

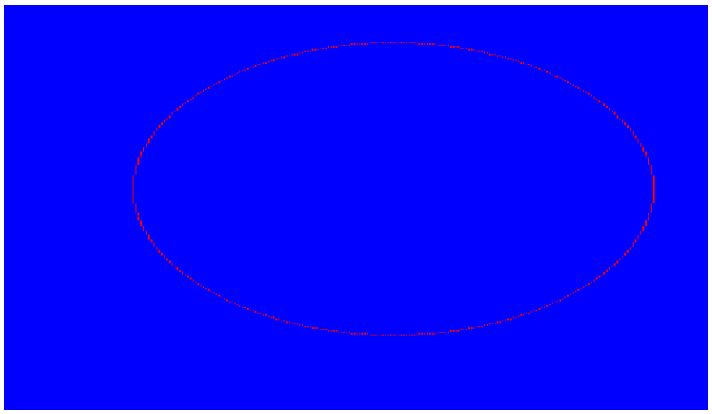

Figure 3: Circle of radius 100 cm

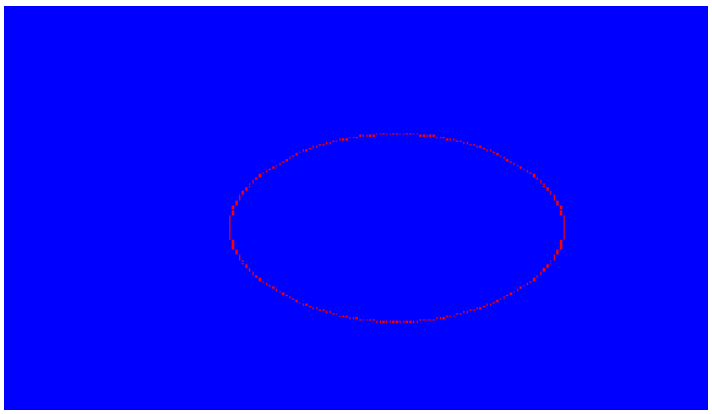

Figure 4: Circle of radius 50 cm

5. Python

Circle using numpy and skimage

step 2: Python numpy code

```
import matplotlib.pyplot as plt
from skimage import draw
arr = np.zeros((200, 200))
rr, cc = draw.circle_perimeter(100, 100, radius=80, shape=arr.shape)
arr[rr, cc] = 1
```

```
plt.imshow(arr)
plt.show()
```

5. Python Result

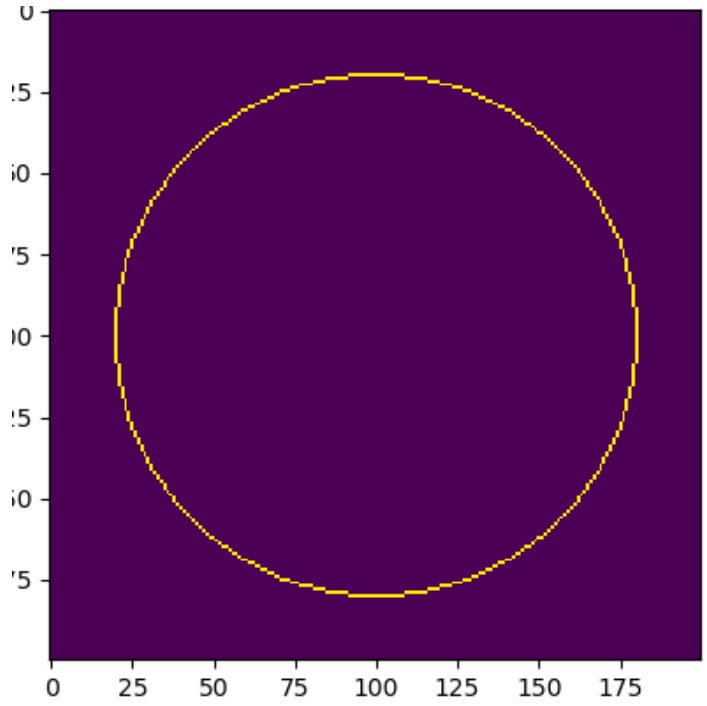

Figure 5: Circle using python