Nama : M Brian Yusuf

NIM : 20051397036

Kelas : 2020B D4MI

MEMBUAT LINGKARAN DENGAN ALGORITMA BRESENHAM

PENGERTIAN ALGORITMA BRESENHAM

Algoritma bresenham merupakan suatu algoritma (pendekatan) yang dikreasikan oleh bresenham yang tidak kalah akurat dan efisien dengan algoritma primitif lainnya (seperti DDA). Bagian pengkonversian (scan-knversi) garis akan melakukan kalkulasi untuk penambahan nilai-nilai integer (yang dibutuhkan untuk membentuk garis) yang disesuaikan dengan tipe grafik yang dipakai oleh layar komputer (keadaan monitor pc) kita. Untuk mengilustrasikan pendekatan bresenham, pertama kita harus memperhatikan proses scan- konvensi untuk garis dengan slope positif yang lebih kecil dari 1. Posisi pixel sepanjang line-path kemudian ditentukan dengan penyamplingan pada unit interval x.dimulai dari endpoint kiri (Xo,Yo) dari garis yang diberikan, kita pindahkan beberapa kolom berturut-turut (berdasarkan posisi x) dan plot pixel-pixel yang mempunyai nilai scan-line y ke jarak yang paling dekat dengan line-path.

ATURAN BRESENHAM

1. Jika Pk bernilai positif (+), maka tambahkan hasilnya dengan B dan nilai x dan y ditambah 1.
2. Jika Pk bernilai negatif (-), maka tambahkan hasilnya dengan A dan nilai x ditambah 1, sedangkan y ditambah 0 (tetap).
3. Putaran dihentikan jika koordinat x dan y sudah mencapai batas akhir.

PRINSIP DARI ALGORITMA BRESENHAM

1. Sumbu vertikal memperlihatkan posisi scan line.
2. Sumbu horizontal memperlihatkan kolom pixel.
3. Pada tiap langkah, penentuan pixel selanjutnya didasari oleh parameter integer yang nilainya proporsional dengan pengurangan antara vertical separations dari dua posisi piksel dari nilai actual.

SOURCE CODE :

```
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *


def init():
    glClearColor(0.0, 0.0, 0.0, 0.0)
    gluOrtho2D(-50.0, 50.0, -50.0, 50.0)
    glPointSize(5)


def plot(x, y):
    glBegin(GL_POINTS)
    glVertex2f(x, y)
    glEnd()


def bresenham_drawing_circle(r):

    # contoh soal jika nilai center circle (3,3) dan r = 9
    x_center = 3
    y_center = 3

    r = 9
    x = 0
    y = r

    # parameter keputusan bresenham
    d = 3 - 2 * r

    # membuat titik pada koordinat yang ditentukan
    plot(x + x_center, y + y_center)

    while y > x:

        if d < 0:
            x += 1
            d += 4 * x + 6
        else:
            x += 1
            y -= 1
            d += (4 * (x - y)) + 10

        # Jika x <= y sudah terpenuhi, maka perulangan akan di hentikan.
        # jika nilai (x,y)sudah didapatkan, lalu membalikan nilai (y, x)

        # Untuk pixel (x, y)

        # Quadrant 1
        plot(x + x_center, y + y_center)

        # Quadrant 2
        plot(x + x_center, -y + y_center)

        # Quadrant 3
```

```
        plot(-x + x_center, -y + y_center)

        # Quadrant 4
        plot(-x + x_center, y + y_center)

        # Untuk pixel (y, x)

        # Quadrant 1
        plot(y + x_center, x + y_center)

        # Quadrant 2
        plot(-y + x_center, x + y_center)

        # Quadrant 3
        plot(-y + x_center, -x + y_center)

        # Quadrant 4
        plot(y + x_center, -x + y_center)


def plotpoints():

    glClear(GL_COLOR_BUFFER_BIT)
    glColor4f(1.0, 1.0, 0.0, 0.0)

    glBegin(GL_LINES)

    glVertex2f(-50, 0)
    glVertex2f(50, 0)

    glVertex2f(0, -50)
    glVertex2f(0, 50)

    glEnd()

    bresenham_drawing_circle(40)

    glFlush()

def main():
    glutInit(sys.argv)
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB)
    glutInitWindowSize(500, 500)
    glutInitWindowPosition(100, 100)
    glutCreateWindow("Bresenham Cricle")
    glutDisplayFunc(plotpoints)

    init()
    glutMainLoop()

main()
```

OUTPUT :