LAPORAN

TUGAS GRAFIKA KOMPUTER

DISUSUN OLEH

Ari Kurniawan (20051397057)

UNIVESITAS NEGERI SURABAYA

FAKULTAS VOKASI

MANAJEMEN INFORMATIKA

2020

- **Cara Kerja Algoritma Bresenham**

  Algoritma Bresenham adalah sebuah algoritma yang dibentuk oleh bresenham yang tidak kalah akurat/efisien dengan algoritma primitif lainnya (DDA). Algoritma Bresenham ini dikembangkan oleh Jack Bresenham pada tahun 1962. Cara kerja Algoritma Bresenham adalah mengecek garis yang sudah diubah hanya dengan menggunakan metode perhitungan integer yang nantinya akan terus bertambah sehingga bisa menampilkan bentuk lingkaran dan bentuk kurva lainnya. Langkah-langkah sebagai berikut:

  1. Masukkan 2 titik, kemudian simpan titik yang paling kiri sebagai nilai (X,Y)
  2. Plotkan titik yang pertama.
  3. Hitung Δx, Δy, $2\Delta y$ dan $2\Delta y-2\Delta x$ dan dapatkan nilai awal parameter keputusan sbb: p0= $2\Delta y-\Delta x$
  4. Setiap Xk sepanjang garis, mulai dari k=0, lakukan langkah pengujian sbb: jika pk<0, maka titik selanjutnya yang akan diplot adalah (Xk+1, Yk), kemudian :

     Pk+1=Pk+ $2\Delta y$ jika sebaliknya, maka titik selanjutnya memiliki nilai (Xk+1, Yk+1), lalu perhitungannya : pk+1=pk+ $2\Delta y- 2\Delta x$
  5. Ulangi langkah ke-4 sebanyak Δx kali.

- **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(-100.0, 100.0, -100.0, 100.0)
    glPointSize(5)


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

```
def bresenham_circle_drawing(r):

    # lokasi lingkaran yang akan dibuat

    x_position = -50
    y_position = 50

    x = 0
    y = r

    # decision parameter
    d = 3 - 2 * r

    # membuat sebuah titik pada koordinat yang ditentukan
    plot(x + x_position, y + y_position)
    while y > x:

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

        # Tidak perlu mencari semua nilai koordinat
        # cukup dapatkan nilai (x, y) saja
        # lalu balik menjadi (y, x)

        # Untuk pixel (x, y)

        # Quadrant 1
        plot(x + x_position, y + y_position)

        # Quadrant 2
        plot(x + x_position, -y + y_position)

        # Quadrant 3
        plot(-x + x_position, -y + y_position)

        # Quadrant 4
        plot(-x + x_position, y + y_position)

        # Untuk pixel (y, x)

        # Quadrant 1
        plot(y + x_position, x + y_position)
```

```
        # Quadrant 2
        plot(-y + x_position, x + y_position)

        # Quadrant 3
        plot(-y + x_position, -x + y_position)

        # Quadrant 4
        plot(y + x_position, -x + y_position)


def plotpoints():

    glClear(GL_COLOR_BUFFER_BIT)
    glColor3f(1, 1.0, 1.0)

    glBegin(GL_LINES)

    glVertex2f(-100, 0)
    glVertex2f(100, 0)

    glVertex2f(0, -100)
    glVertex2f(0, 100)

    glEnd()

    bresenham_circle_drawing(40)

    glFlush()


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

    init()
    glutMainLoop()


main()
```

- **Output**