LAPORAN PRAKTIKUM GRAFIKA KOMPUTER

"Membuat Lingkaran Bresenham dan Midpoint"

Dosen Pangampu:

Febi Eka Febriansyah, M.T.

Wartariyus, S.Kom., M.T.I.

Putut Aji Nalendro, S.Pd., M.Pd.

Disusun Oleh:

Nama: Nabila Fatma Sari

NPM: 2453025003

Kelas: PTI 24 A

PROGRAM STUDI PENDIDIKAN TEKNOLOGI INFORMASI

JURUSAN PENDIDIKAN DAN ILMU PENGETAHUAN

FAKULTAS KEGURUAN DAN ILMU PENDIDIKAN

UNIVERSITAS LAMPUNG

2025

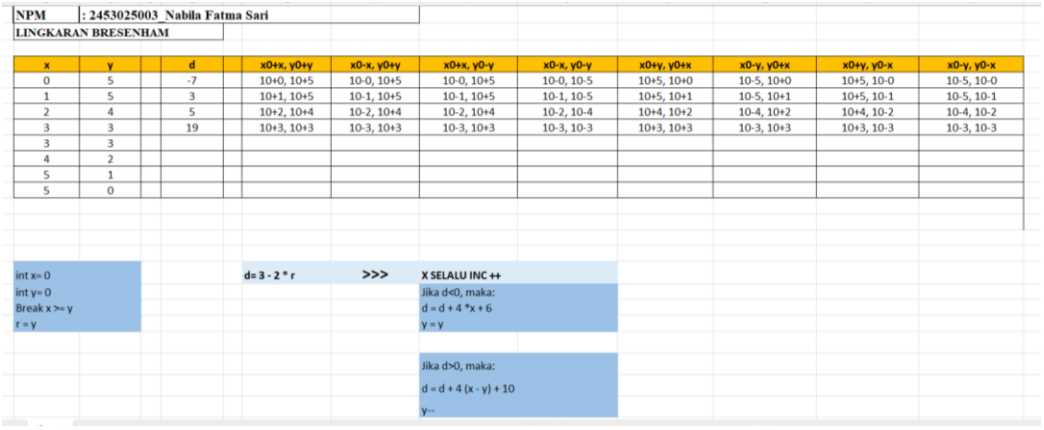
Lingkaran Bresenham

Lingkaran Bresenham adalah metode untuk menggambar lingkaran pada grid raster 2D (piksel layar) dengan kalkulasi integer. Tidak seperti persamaan lingkaran tradisional, yang melibatkan operasi floating-point, algoritma Bresenham hanya menggunakan penjumlahan dan pengurangan, sehingga jauh lebih cepat.

Cara Kerja

- Tentukan pusat lingkaran jari – jari (r)
- Inisialisasi nilai x = 0, y = r, dan p = 3 – 2 * r
- Plot titik (x, y) dan gunakan 8 simetri untuk menggambar seluruh lingkaran
- Ulangi Prosesnya:
    - Jika P < 0, p = p + 4X + 6
    - Jika P ≥ 0, p = p + (x – y) + 10 dan y = y – 1
    - x = x + 1
- Ulangi hingga x ≥ y sampai selesai

Berikut Tabel Lingkaran Bresenham



NPM : 2453025003_Nabila Fatma Sari

LINGKARAN BRESENHAM

x	y	d	x0+x, y0+y	x0-x, y0+y	x0+x, y0-y	x0-x, y0-y	x0+y, y0+x	x0-y, y0+x	x0+y, y0-x	x0-y, y0-x
0	5	-7	10+0, 10+5	10-0, 10+5	10-0, 10+5	10-0, 10-5	10+5, 10+0	10-5, 10+0	10+5, 10-0	10-5, 10-0
1	5	3	10+1, 10+5	10-1, 10+5	10-1, 10+5	10-1, 10-5	10+5, 10+1	10-5, 10+1	10+5, 10-1	10-5, 10-1
2	4	5	10+2, 10+4	10-2, 10+4	10-2, 10+4	10-2, 10-4	10+4, 10+2	10-4, 10+2	10+4, 10-2	10-4, 10-2
3	3	19	10+3, 10+3	10-3, 10+3	10-3, 10+3	10-3, 10-3	10+3, 10+3	10-3, 10+3	10+3, 10-3	10-3, 10-3
3	3									
4	2									
5	1									
5	0									

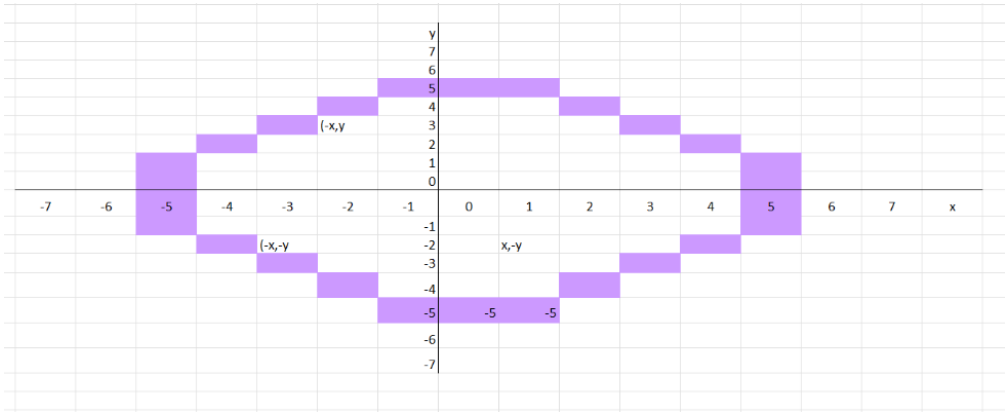
int x= 0
int y= 0
Break x >= y
r = y

d= 3 - 2 * r >>>

X SELALU INC ++
Jika d<0, maka:
d = d + 4 *x + 6
y = y

Jika d>0, maka:
d = d + 4 (x - y) + 10
y--

Berikut Grafik Lingkaran Bresenham



Lingkaran Midpoint

Lingkaran Midpoint adalah metode efesien untuk menggambar lingkaran dalam grafika komputer menggunakan perhitungan berbasis integer. Algoritma ini mirip dengan algoritma Bresenham, tetapi didasarkan pada konsep titik tengan (Midpoint) untuk menentukan piksel berikutnya.

Cara Kerja

- Tentukan pusat lingkaran jari jari (r)
- Inisialisasi $x = 0$, $y = r$, dan $p = 1 – r$
- Plot titik (x, y) dan gunakan 8 simetri untuk menggambar seluruh lingkaran
- Ulangi Prosesnya:
    - Jika $P < 0$, $p = p + 2x + 1$
    - Jika $P \geq 0$, $p = p + 2(x, y) + 5$ dan $y = y – 1$
    - $x = x + 1$
- Ulangi hingga $x \geq y$ sampai selesai

Berikut Tabel Lingkaran Midpoint

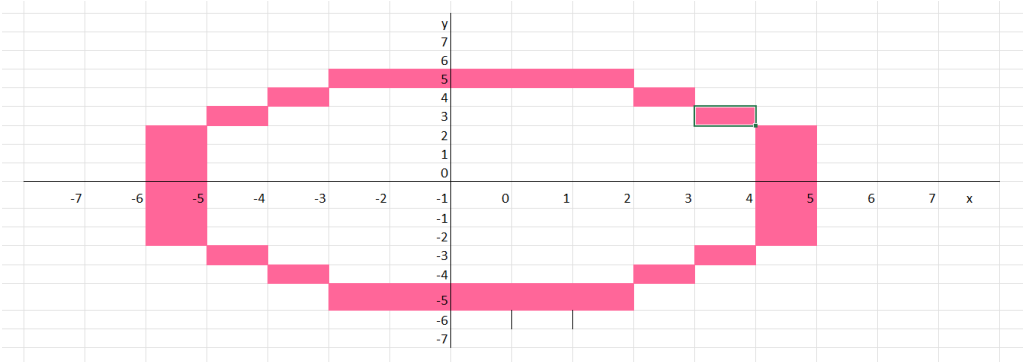
NPM	: 2453025003_Nabila Fatma Sari									
LINGKARAN MIDPOINT										
x	y	p	x0+x, y0+y	x0-x, y0+y	x0+x, y0-y	x0-x, y0-y	x0+y, y0+x	x0-y, y0+x	x0+y, y0-x	x0-y, y0-x
0	5	-4	10+0, 10+5	10-0, 10+5	10-0, 10+5	10-0, 10-5	10+5, 10+0	10-5, 10+0	10+5, 10-0	10-5, 10-0
1	5	-1	10+1, 10+5	10-1, 10+5	10-1, 10+5	10-1, 10-5	10+5, 10+1	10-5, 10+1	10+5, 10-1	10-5, 10-1
2	5	5	10+2, 10+5	10-2, 10+5	10-2, 10+5	10-2, 10-5	10+5, 10+2	10-5, 10+2	10+5, 10-2	10-5, 10-2
3	4	4	10+3, 10+4	10-3, 10+4	10-3, 10+4	10-3, 10-4	10+4, 10+3	10-4, 10+3	10+4, 10-3	10-4, 10-3
4	3	10	10+3, 10+3	10-3, 10+3	10-3, 10+3	10-3, 10-3	10+3, 10+3	10-3, 10+3	10+3, 10-3	10-3, 10-3
5	2									
5	1									
5	0									

int x= 0
int y= 0
Break x >= y
r = y

p = 1 - r >>> X SELALU INC ++
Jika p < 0, maka
p += 2 * x + 1
y = y

Jika p > 0 maka,
p = p + 2 * (x - y) + 1
y --

Berikut Grafik Lingkaran Midpoint



Code Program dan Penjelasan

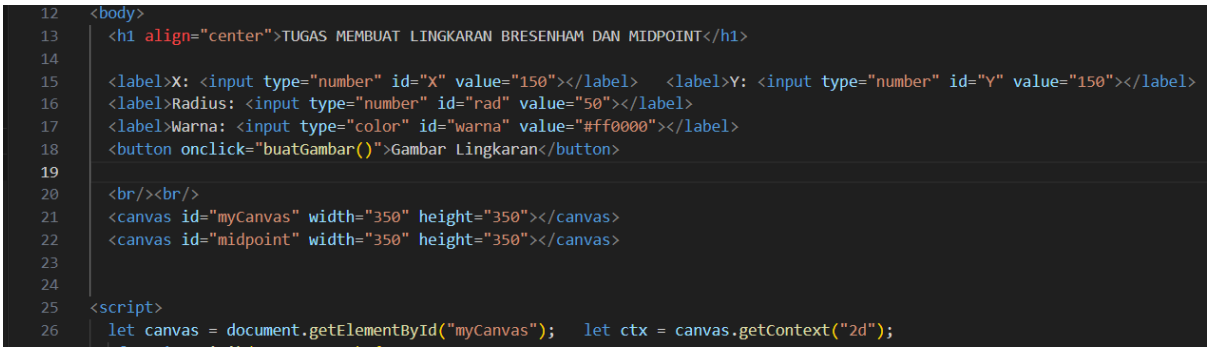
1. Kode HTML

```
!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>2453025003_NabilaFatmaSari</title>
  <style>     canvas {
      border: 1px solid #000000;       margin: 10px;
    }
  </style>
</head>
```

- <!DOCTYPE html>, ini menentukan bahwa dokumen ini adalah HTML5
- <meta charset=”UTF-8”>, memungkinkan pengguna karakter Unicode

- <meta name=”viewport”>, menyesuaikan tampilan agar reponsif di perangkat mobile
- <tittle>, ini untuk menampilkan judul halaman di browser
- Canvas
    - Border: 1px solid #000000; (Memberikan garis tepi hitam pada area canvas)
    - Margin: 10px; (Memberikan jarak agar tampilan lebih rapi)

2. Elemen Canvas dan Input Data



```
12  <body>
13    <h1 align="center">TUGAS MEMBUAT LINGKARAN BRESENHAM DAN MIDPOINT</h1>
14
15    <label>X: <input type="number" id="X" value="150"></label>   <label>Y: <input type="number" id="Y" value="150"></label>
16    <label>Radius: <input type="number" id="rad" value="50"></label>
17    <label>Warna: <input type="color" id="warna" value="#ff0000"></label>
18    <button onclick="buatGambar()">Gambar Lingkaran</button>
19
20    <br/><br/>
21    <canvas id="myCanvas" width="350" height="350"></canvas>
22    <canvas id="midpoint" width="350" height="350"></canvas>
23
24
25  <script>
26    let canvas = document.getElementById("myCanvas");   let ctx = canvas.getContext("2d");
```

- Judul halaman ditampilkan di tengah (align=”center”)
- Menjelasakan tugas tentang pembuatan lingkaran dengan metode Bresenham dan Midpoint
- Input X & Y, menentukan titik pusat lingkaran
- Input Radius, menentukan ukuran lingkaran
- Input Warna, memilih warna lingkaran
- Button “Gambar Lingkaran”
    - Ketika diklik, fungsi buatGambar() akan dipanggil. Fungsi ini bertanggung jawab untuk menggambar lingkaran dengan parameter yang diberikan.
- Canvas pertama (myCanvas), digunakan untuk menggambar lingkaran dengan algoritma Bresenham
- Canvas kedua (midpoint), digunakan untuk menggambar lingkaran dengan algoritma Midpoint
- Ukuran masing – masing 350x350 pixel
- Mengambil elemen <<canvas>> dengan id “myCanvas”

- Mendapatkan konteks 2D (getContext("2d")) agar bisa menggambar di dalamnya

3. Fungsi Titik

```
27    function titik(x, y, warna) {
28        ctx.fillStyle = warna;        ctx.fillRect(x, y, 1, 3);
29    }
```

- Titik(x, y, warna), fungsi yang digunakan untuk menggambar titik pada canvas
- Ctx.fillStyle = warna;
  - Mengatur warna yang digunakan untuk menggambar titik
  - Warna diambil dari parameter fungsi (warna)
- Ctx.fillRect(x, y, 1, 3);
  - Menggambar sebuah persegi panjang kecil, dengan lebar 1 pixel dan tinggi 3 pixel
  - (x,y) adalah lokasi yang akan digambar

4. Fungsi "gambarTitikSimetris" dan Fungsi LinkBre(Bresenham)

```
31    function gambarTitikSimetris(x0, y0, x, y, warna) {
32      titik(x0 + x, y0 + y, warna);
33      titik(x0 - x, y0 + y, warna);
34      titik(x0 + x, y0 - y, warna);
35      titik(x0 - x, y0 - y, warna);
36      titik(x0 + y, y0 + x, warna);
37      titik(x0 - y, y0 + x, warna);
38      titik(x0 + y, y0 - x, warna);
39      titik(x0 - y, y0 - x, warna);
40    }
41    function linkBre(x0, y0, r, warna) {       var d = 3 - 2 * r;       var x = 0, y = r;
42
43      while (x <= y) {
44        gambarTitikSimetris(x0, y0, x, y, warna);        if (d <= 0) {          d = d + 4 * x + 6;
45        } else {
46        d = d + 4 * (x - y) + 10;         y--;
47      }        x++;
48    }
49    }
```

- Setiap iterasi menggambar 8 titik yang simeteris terhadap pusat (x0, y0)
- Menggunakan aturan simetri (x, y) → (y, x) untuk memanfaatkan efesiensi perhitungan
- Fungsi titik(x, y, warna) digunakan untuk menggambar titik pada canvas dengan warna tertentu
- Inisialisasi Variabel:

- d = 3 – 2 * r → Parameter keputusan awal dalam algoritma bresenham
- x = 0, y = r → Memulai dari titk paling atas lingkaran

- Perulangan while (x <= y)
  - Jika d <= 0, berarti titik berikutnya masih dalam/batas lingkaran, hanya perlu memperbesar x
  - Jika d > 0, titik keluar dari batas, harus juga mengurangi y juga
  - X selalu bertambah dalam setiap iterasi

5. Fungsi buatGambar

```
function buatGambar() {
  ctx.clearRect(0, 0, canvas.width, canvas.height);       ctxmidpoint.clearRect(0, 0, midpoint.width, midpoint.height);

  let x0 = parseInt(document.getElementById("X").value);
  let y0 = parseInt(document.getElementById("Y").value);
  let r = parseInt(document.getElementById("rad").value);
  let warna = document.getElementById("warna").value;

  linkBre(x0, y0, r, warna);       drawmidpoint(x0, y0, r);
}

let midpoint = document.getElementById("midpoint");     let ctxmidpoint = midpoint.getContext("2d");
```

- Ctx.clearRect(0, 0, canvas,width, canvas.height);
  - Membersihkan area canvas utama sebelum menggambar ulang menggunakan algoritma Bresenham
- Ctxmidpoint.clearRect(0, 0, midpoint.width, midpoint.height;
  - Membersihkan canvas kedua sebelum menggambar ulang menggunakan algoritma Midpoint
- Mengambil nilai koordinat pusat (x0, y0), jari – jari r, dan warna dari input HTML
- parseInt, digunakan untuk memastikan nilai yang diambil adalah bilangan bulat (integer)
- Warna diambil dari elemen input color picker
- linkBre(x0, y0, r, warna);
  - Memanggil fungsi Bresenham untuk menggambar lingkaran pada canvas pertama
- drawmidpoint(x0, y0, r);
  - Memanggil fungsi Midpoint unutuk menggambar lingkaran pada canvas kedua
- let midpoint = document.getElemenById(“midpoint”);

- Mengambil elemen <canvas> kedua untuk metode Midpoint

- Let ctxmidpoint = midpoint.getcontext(“2d”);
  - Mendapatkan context 2D untuk menggambar pada canvas kedua

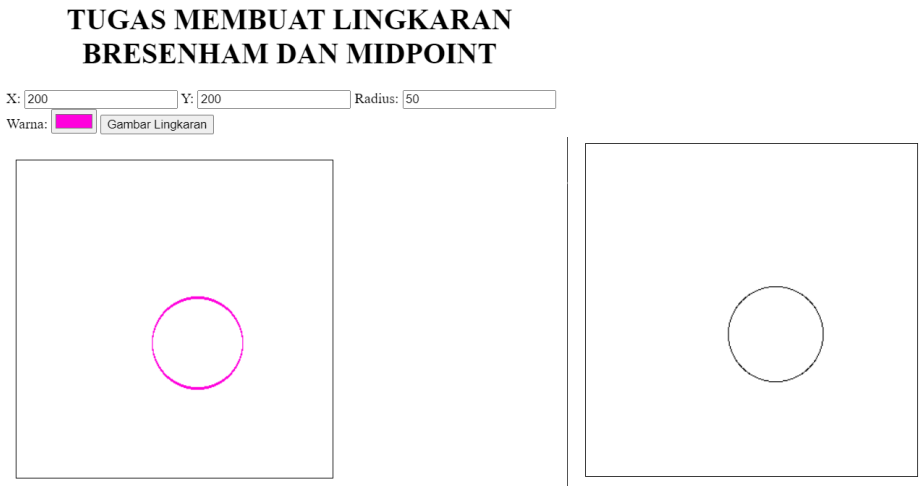
6. Fungsi drawpoint

```
function drawmidpoint(x0, y0, r) {      let x = r;      let y = 0;      let d = r - 1;

while (x >= y) {
  ctxmidpoint.fillRect(x0 + x, y0 + y, 1, 1);
  ctxmidpoint.fillRect(x0 - x, y0 + y, 1, 1);
  ctxmidpoint.fillRect(x0 + x, y0 - y, 1, 1);
  ctxmidpoint.fillRect(x0 - x, y0 - y, 1, 1);
  ctxmidpoint.fillRect(x0 + y, y0 + x, 1, 1);
  ctxmidpoint.fillRect(x0 - y, y0 + x, 1, 1);
  ctxmidpoint.fillRect(x0 + y, y0 - x, 1, 1);
  ctxmidpoint.fillRect(x0 - y, y0 - x, 1, 1);

    if (d >= 2 * y) {
      d -= 2 * y + 1;
      y++;
    } else if (d < 2 * (r - x)) {
      d += 2 * x - 1;      x--;      } else {      d += 2 * (x - y - 1);      x--;      y++;      }
  }
}
```

- let x = r; → Inisialisasi x dengan radius lingkaran
- let y = 0; → Inisialisasi y sebagai 0 (memulai dari titik kanan lingkaran)
- let d = r – 1; → Inisialisasi nilai keputusan untuk menentukan posisi pixel berikutnya
- Loop while (x >= y)
  - Loop ini terus berjalan selama x lebih besar atau sama dengan y, untuk menggambar satu per satu titik dalam seperdelapan lingkaran, yang kemudian direplikasikan ke 8 bagian dengan simetri
- Menggambar 8 simetri titik:
  - Menggunakan sifat simetri lingkaran untuk menggambar titik di delapangan posisi berbeda berdasarkan perhitungan (x, y)
  - fillRect(x, y, 1, 1) menggambar 1 pixel kecil di posisi yang dihitung
- Perhitungan algoritma Midpoint:
  - Jika nilai d lebih besar atau sama dengan 2 * y, berarti pixel selanjutknya harus lebih tinggi (y bertamabah)
  - Perubahan keputusan d diperbarui dengan d = d – 2 * y + 1
  - Jika nilai d lebih kecil dari 2 * (r – x), berarti pixel selanjutnya harus lebih ke dalam (x berkurang)
  - Perubahan keputusan d diperbarui dengan d += 2 * x – 1

- Jika tidak memenuhi kedua kondisi di atas, berarti titik selanjutnya berada di diagonal, sehingga x berkurang dan y bertambah
- d diperbarui dengan d+= 2 * (x – y – 1);

Output Yang Dihasilkan



Kesimpulan

Lingkaran Bresenham dan Lingkaran Midpoint sama – sama digunakan untuk menggambar lingkaran dengan perhitungan yang cepat dan efesien. Lingkaran Midpoint lebih sederhana dan mudah diemplementasikan, sedangkan Bresenham lebih optimal dalam beberapa kasus karena mempertimbangkan lebih banyak perubahan posisi piksel. Keduanya sering digunakan dalam grafika komputer untuk menggambar lingkaran dengan presisi tinggi tanpa operasi akar atau pembagian.