LAPORAN TUGAS GRAFIKA KOMPUTER

“Algoritma Pembentukan Lingkaran Bresenham dan Midpoint”

Dosen Pengampu :

Febi Eka Febriansyah, M.T.

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

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

Disusun Oleh:

Nama: Anissa Vika Anandari

NPM: 2413025005

PROGRAM STUDI PENDIDIKAN TEKNOLOGI INFORMASI

JURUSAN PENDIDIKAN MATEMATIKA DAN ILMU PENGETAHUAN ALAM

FAKULTAS KEGURUAN DAN ILMU PENDIDIKAN

UNIVERSITAS LAMPUNG

2025

1. Algoritma Bresenham

Algoritma Bresenham adalah metode untuk menggambar lingkaran yang dikembangkan oleh Jack E. Bresenham. Algoritma ini menggunakan operasi perhitungan sederhana seperti penjumlahan dan pengurangan untuk menentukan titik-titik lingkaran tanpa menggunakan operasi pembagian atau perkalian yang kompleks.

Cara kerja algoritma:

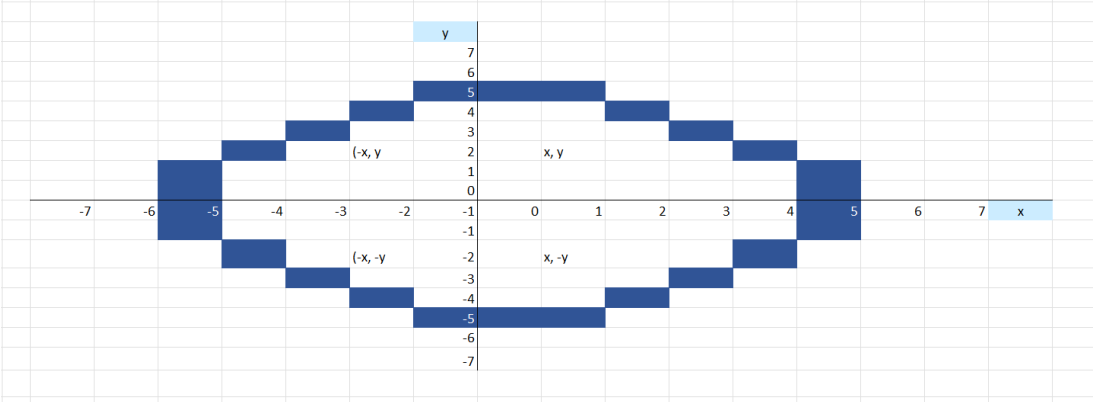
- Memulai dari titik atas lingkaran dan bergerak searah jarum jam dengan mengevaluasi keputusan berdasarkan nilai variabel keputusan (decision parameter).
- Jika titik yang dihitung berada lebih dekat ke lingkaran, maka tetap pada posisi horizontal. Jika tidak, maka turun satu piksel ke bawah.
- Menggunakan prinsip simetri agar hanya perlu menghitung satu per delapan lingkaran, lalu merefleksikan titik-titik ke bagian lainnya.

1) Tabel Bresenham

NPM : 2413025005_Anissa Vika Anandari

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	11		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
4	2										
5	1										
5	0										

2) Diagram



2. Algoritma Midpoint

Algoritma Midpoint juga digunakan untuk menggambar lingkaran dengan menentukan titik tengah antara dua piksel dan memutuskan apakah titik berikutnya berada di dalam atau di luar lingkaran.

Cara kerja algoritma:

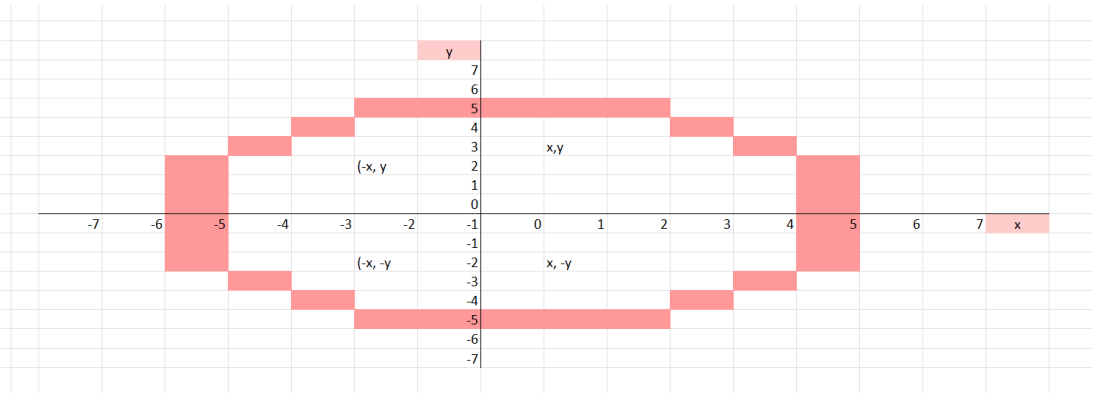
- Menggunakan titik awal di bagian atas lingkaran.
- Menggunakan perhitungan berbasis titik tengah (midpoint) untuk menentukan apakah piksel selanjutnya tetap di jalur lingkaran atau harus bergeser.
- Memanfaatkan simetri delapan bagian lingkaran untuk mempercepat proses perhitungan.

1) Tabel Midpoint

NPM : 2413025005_Anissa Vika Anandari

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	4		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	3		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	6		10+4, 10+3	10-4, 10+3	10+4, 10-3	10-4, 10-3	10+3, 10+4	10-3, 10+4	10+3, 10-4	10-3, 10-4
4	3										
5	2										
5	1										
5	0										

2) Diagram



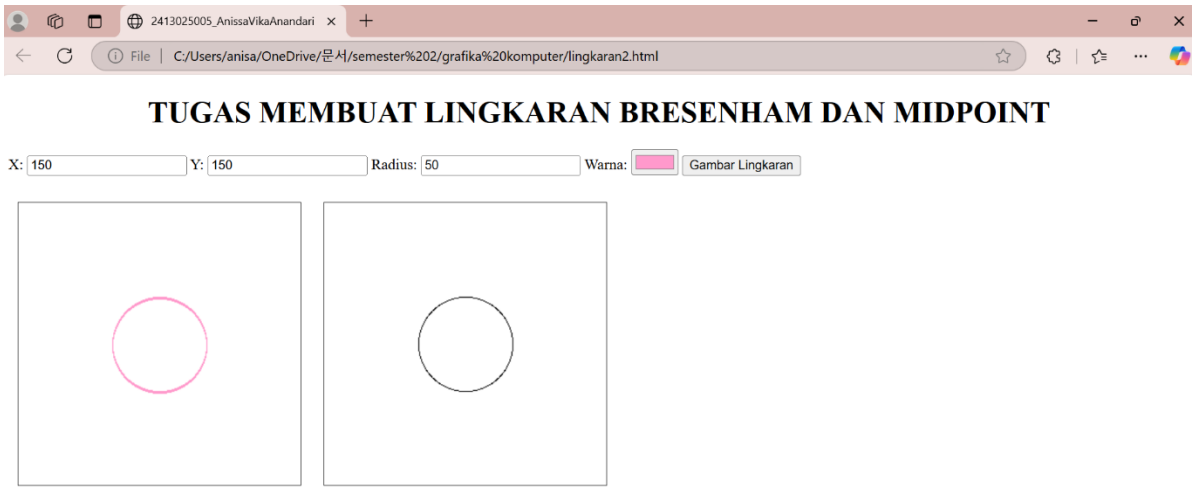
Kode Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>2413025005_AnissaVikaAnandari</title>
  <style>
    canvas {
      border: 1px solid #000000;
      margin: 10px;
    }
  </style>
</head>
<body>
  <h1 align="center">TUGAS MEMBUAT LINGKARAN BRESENHAM DAN MIDPOINT</h1>

  <label>X: <input type="number" id="X" value="150"></label>
  <label>Y: <input type="number" id="Y" value="150"></label>
  <label>Radius: <input type="number" id="rad" value="50"></label>
  <label>Warna: <input type="color" id="warna" value="#ff99cc"></label>
  <button onclick="buatGambar()">Gambar Lingkaran</button>

  <br/><br/>
  <canvas id="myCanvas" width="300" height="300"></canvas>
  <canvas id="midpoint" width="300" height="300"></canvas>

  <script>
    let canvas = document.getElementById("myCanvas");
    let ctx = canvas.getContext("2d");

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

    function gambarTitikSimetris(x0, y0, x, y, warna) {
      titik(x0 + x, y0 + y, warna);
      titik(x0 - x, y0 + y, warna);
      titik(x0 + x, y0 - y, warna);
      titik(x0 - x, y0 - y, warna);
      titik(x0 + y, y0 + x, warna);
      titik(x0 - y, y0 + x, warna);
      titik(x0 + y, y0 - x, warna);
      titik(x0 - y, y0 - x, warna);
    }
```

```
function linkBre(x0, y0, r, warna) {
  var d = 3 - 2 * r;
  var x = 0, y = r;

  while (x <= y) {
    gambarTitikSimetris(x0, y0, x, y, warna);
    if (d <= 0) {
      d = d + 4 * x + 6;
    } else {
      d = d + 4 * (x - y) + 10;
      y--;
    }
    x++;
  }
}

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");

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++;
        }
      }
    }
  </script>
</body>
</html>
```

Output:



Kesimpulan

Algoritma Bresenham lebih optimal untuk perangkat keras dengan keterbatasan memori dan lebih cocok digunakan pada sistem dengan perhitungan berbasis integer, sedangkan Algoritma Midpoint lebih sederhana dalam implementasi dan lebih banyak digunakan dalam komputer grafis modern karena kemudahannya dalam menentukan titik lingkaran.

Tautan Video: https://youtu.be/OjhdSKjeJlU