LAPORAN PRAKTIKUM MATA KULIAH GRAFIKA KOMPUTER

"Implementasi Algoritma Bresenham Dan Midpoint Untuk Menggambar Lingkaran"

Dosen Mata Kuliah :

Febi Eka Febriansyah, M.T.

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

Putut Aji Nalendro, M.Pd.

Disusun Oleh :

Nama : Rhosa Thatia Anista

NPM : 2413025022

PROGRAM STUDI PENDIDIKAN TEKNOLOGI INFORMASI

FAKULTAS KEGURUAN DAN ILMU PENDIDIKAN

UNIVERSITAS LAMPUNG

2025

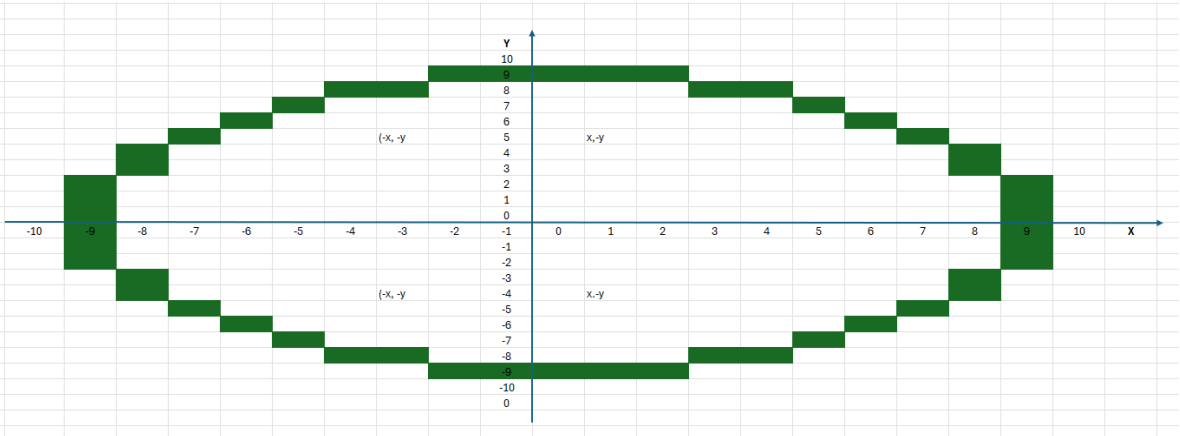
A. Lingkaran Bresenham

Algoritma Bresenham adalah metode efisien untuk menggambar lingkaran tanpa operasi trigonometri atau pembagian, memanfaatkan simetri delapan bagian untuk mengurangi perhitungan. Dimulai dari titik atas lingkaran, algoritma ini menggunakan variabel keputusan untuk menentukan pergeseran titik berikutnya. Jika nilai keputusan negatif, koordinat x bertambah; jika positif, koordinat y berkurang. Pendekatan ini memastikan piksel tersusun rapi dan pemrosesan lebih efisien.

Nama : Rhosa Thatia Anista
NPM : 2413025022
TUGAS TABEL LINGKARAN BRESENHAM

X	Y	d	x0+x, y0+y	x0-x, y0-y	x0+x, y0-y	x0+y, y0+x	x0-y, y0-x
0	9	-15	10+0, 10+5	10-0, 10+5	10+0, 10-5	10+5, 10+0	10-5, 10-0
1	9	-5	10+1, 10+9	10-1, 10+9	10+1, 10-9	10+9, 10+1	10-9, 10-1
2	9	9	10+2, 10+9	10-2, 10+9	10+2, 10-9	10+9, 10+2	10-9, 10-2
3	8	-1	10+3, 10+8	10-3, 10+8	10+3, 10-8	10+8, 10+3	10-8, 10-3
4	8	21	10+4, 10+8	10-4, 10+8	10+4, 10-8	10+8, 10+4	10-8, 10-4
5	7	23	10+5, 10+7	10-5, 10+7	10+5, 10-7	10+7, 10+5	10-7, 10-5
6	6	33	10+6, 10+6	10-6, 10+6	10+6, 10-6	10+6, 10+6	10-6, 10-6
6	6						
7	5						
8	4						
8	3						
int x=0		d=3-2*r		jika d < maka y tetap			
int y=0				d = d+4* x+6			
					x selalu inc ++		
				jika d > 0 maka y --			
				d = d+4 (x+y)+10			

Tabel di atas berisi perhitungan iteratif algoritma Bresenham untuk menggambar lingkaran, dimulai dari koordinat (x, y) = (0, r) dan menggunakan variabel keputusan d untuk menentukan pergeseran titik berikutnya. Kolom x menunjukkan peningkatan nilai koordinat x, sementara kolom g (y) menunjukkan perubahan koordinat y sesuai kondisi d. Jika d < 0, hanya x yang bertambah, sedangkan jika d ≥ 0, maka y berkurang. Kolom d mencatat nilai variabel keputusan pada setiap iterasi. Kolom koordinat (x0 ± x, y0 ± y) menampilkan delapan titik simetris berdasarkan pusat (x0, y0), karena lingkaran memiliki sifat simetri terhadap sumbu x, sumbu y, dan diagonal.



B. Lingkaran MidPoint

Algoritma Midpoint Circle menggambar lingkaran secara efisien dengan menentukan titik berdasarkan variabel keputusan. Jika keputusan negatif, titik bergeser ke kanan, sedangkan jika positif, titik juga turun. Memanfaatkan simetri delapan bagian, algoritma ini hanya menghitung seperdelapan lingkaran dan merefleksikannya, sehingga lebih cepat dan akurat tanpa operasi trigonometri.

Nama : Rhosa Thatia Anista
NPM : 24132025022
TUGAS TABEL LINGKARAN MINDPOINT

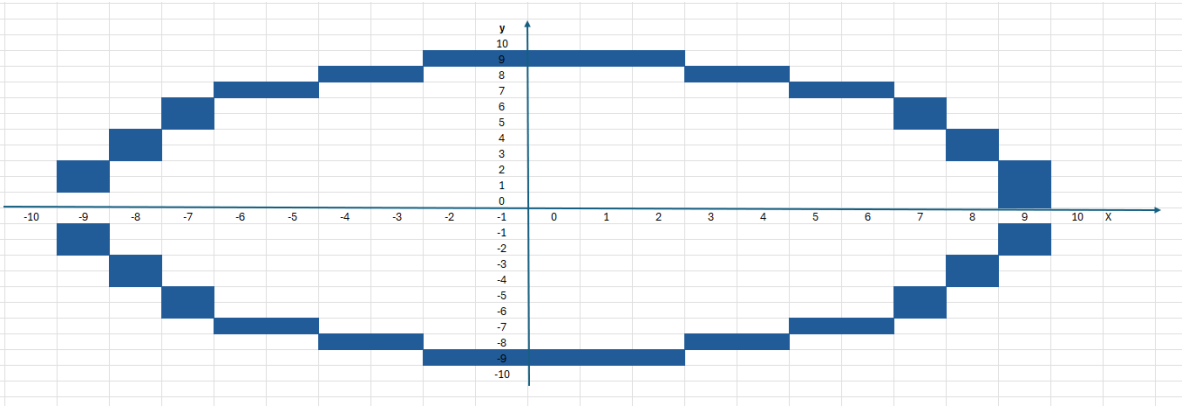
X	Y	P	x0+x, y0+y	x0-x, y0+y	x0+x, y0-y	x0-x, y0-y	x0+y, y0+x	x0-y, y0+x	x0+y, y0-x	x0-y, y0-x
0	9	-8	10+0, 10+5	10-0, 10+5	10+0, 10-5	10-0, 10-5	10+5, 10+0	10-5, 10+0	10+5, 10-0	10-5, 10-0
1	9	-5	10+1, 10+9	10-1, 10+9	10+1, 10-9	10-1, 10-9	10+9, 10+1	10-9, 10+1	10+9, 10-1	10-9, 10-1
2	9	0	10+2, 10+9	10-2, 10+9	10+2, 10-9	10-2, 10-9	10+9, 10+2	10-9, 10+2	10+9, 10-2	10-9, 10-2
3	8	-9	10+3, 10+8	10-3, 10+8	10+3, 10-8	10-3, 10-8	10+8, 10+3	10-8, 10+3	10+8, 10-3	10-8, 10-3
4	8	0	10+4, 10+8	10-4, 10+8	10+4, 10-8	10-4, 10-8	10+8, 10+4	10-8, 10+4	10+8, 10-4	10-8, 10-4
5	7	-3	10+5, 10+7	10-5, 10+7	10+5, 10-7	10-5, 10-7	10+7, 10+5	10-7, 10+5	10+7, 10-5	10-7, 10-5
6	7	10	10+6, 10+7	10-6, 10+7	10+6, 10-7	10-6, 10-7	10+7, 10+6	10-7, 10+6	10+7, 10-6	10-7, 10-6
7	6	13	10+7, 10+6	10-7, 10+6	10+7, 10-6	10-7, 10-6	10+6, 10+7	10+-, 10+7	10+6, 10-7	10+-, 10-7
6	7									
7	6									
7	5									
8	4									
8	3									
9	2									
9	1									

jika p < 0 maka y tetap
p+=2*+1

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

Tabel di atas menampilkan koordinat titik-titik yang dihitung menggunakan algoritma Midpoint Circle untuk menggambar lingkaran, dengan kolom x dan y menunjukkan koordinat yang diperoleh, serta kolom P berisi parameter keputusan yang menentukan arah pergerakan titik selanjutnya. Dengan memanfaatkan sifat simetri lingkaran, titiktitik tersebut dipantulkan ke delapan posisi berbeda berdasarkan pusat lingkaran (x0, y0). Di samping tabel, terdapat implementasi kode C untuk algoritma Midpoint, yang menggunakan perhitungan awal p=1−rp = 1 - r dan aturan pembaruan nilai x dan y berdasarkan nilai p.

Algoritma ini mirip dengan Bresenham's Circle Algorithm, tetapi memiliki pendekatan awal yang sedikit berbeda. Secara keseluruhan, tabel ini berfungsi sebagai pencatatan hasil perhitungan koordinat lingkaran beserta simetri yang diterapkan untuk membentuk keseluruhan lingkaran.



Selain menggunakan excel dalam pembuatannya, penyusun juga membuat lingkaran Bresenham dan Midpoint menggunakan program html serta java script. Berikut kode programnya :

lingkaran.html > html > body > br

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Rhosa Thatia Anista_2413025022</title>
  <style>
    canvas {
      border: 1px solid #000000;
      margin: 10px;
    }
  </style>
</head>
<body>
  <h1 align="center">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="#E4AB16"></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);
```

```
36      function gambarTitikSimetris(x0, y0, x, y, warna) {
37        titik(x0 + x, y0 + y, warna);
38        titik(x0 - x, y0 + y, warna);
39        titik(x0 + x, y0 - y, warna);
40        titik(x0 - x, y0 - y, warna);
41        titik(x0 + y, y0 + x, warna);
42        titik(x0 - y, y0 + x, warna);
43        titik(x0 + y, y0 - x, warna);
44        titik(x0 - y, y0 - x, warna);
45      }
46
47      function linkBre(x0, y0, r, warna) {
48        var d = 3 - 2 * r;
49        var x = 0, y = r;
50
51        while (x <= y) {
52          gambarTitikSimetris(x0, y0, x, y, warna);
53          if (d <= 0) {
54            d = d + 4 * x + 6;
55          } else {
56            d = d + 4 * (x - y) + 10;
57            y--;
58          }
59          x++;
60        }
61      }
62
63      function buatGambar() {
64        ctx.clearRect(0, 0, canvas.width, canvas.height);
65        ctxmidpoint.clearRect(0, 0, midpoint.width, midpoint.height);
```

```
67          let x0 = parseInt(document.getElementById("x").value);
68          let y0 = parseInt(document.getElementById("y").value);
69          let r = parseInt(document.getElementById("rad").value);
70          let warna = document.getElementById("warna").value;
71
72          linkBre(x0, y0, r, warna);
73          drawmidpoint(x0, y0, r);
74        }
75
76        let midpoint = document.getElementById("midpoint");
77        let ctxmidpoint = midpoint.getContext("2d");
78
79        function drawmidpoint(x0, y0, r) {
80          let x = r;
81          let y = 0;
82          let d = r - 1;
83
84          while (x >= y) {
85            ctxmidpoint.fillRect(x0 + x, y0 + y, 1, 1);
86            ctxmidpoint.fillRect(x0 - x, y0 + y, 1, 1);
87            ctxmidpoint.fillRect(x0 + x, y0 - y, 1, 1);
88            ctxmidpoint.fillRect(x0 - x, y0 - y, 1, 1);
89            ctxmidpoint.fillRect(x0 + y, y0 + x, 1, 1);
90            ctxmidpoint.fillRect(x0 - y, y0 + x, 1, 1);
91            ctxmidpoint.fillRect(x0 + y, y0 - x, 1, 1);
92            ctxmidpoint.fillRect(x0 - y, y0 - x, 1, 1);
93
94            if (d >= 2 * y) {
95              d -= 2 * x + 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>
```

Berikut output dari kode program di atas :

C. Kesimpulan

Algoritma Bresenham dan Midpoint Circle adalah dua metode efisien untuk menggambar lingkaran tanpa menggunakan operasi trigonometri atau pembagian, dengan memanfaatkan simetri delapan bagian untuk mengurangi perhitungan. Keduanya bekerja dengan menentukan koordinat titik berdasarkan variabel keputusan, di mana algoritma Bresenham menggunakan pendekatan pergeseran titik berdasarkan nilai d, sedangkan algoritma Midpoint menggunakan parameter p untuk menentukan arah pergerakan titik selanjutnya. Tabel dalam Excel mencatat hasil perhitungan koordinat yang diperoleh dari kedua algoritma ini, yang kemudian juga diimplementasikan menggunakan HTML dan JavaScript untuk menghasilkan visualisasi lingkaran. Dengan memanfaatkan prinsip iteratif dan simetri, kedua metode ini memungkinkan pembuatan lingkaran yang lebih akurat dan efisien dalam berbagai aplikasi grafis.