LAPORAN GRAFIKA KOMPUTER

ALGORITMA LINGKARAN BRESENHAM DAN MIDPOINT

Disusun guna memenuhi tugas mata kuliah Grafika Komputer Dosen Pengampu

Bapak Febi Eka Febriansyah, M.T.,

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

Putut Aji Nalendro, M.PD.

Disusun Oleh:

Nama: Lulu Saputri

NPM: 2413025017

Kelas: PTI 24A

PROGRAM STUDI PENDIDIKAN TEKNOLOGI INFORMASI

JURUSAN PENDIDIKAN MATEMATIKA DAN ILMU PENGETAHUAN ALAM FAKULTAS KEGURUAN DAN ILMU PENDIDIKAN UNIVERSITAS LAMPUNG

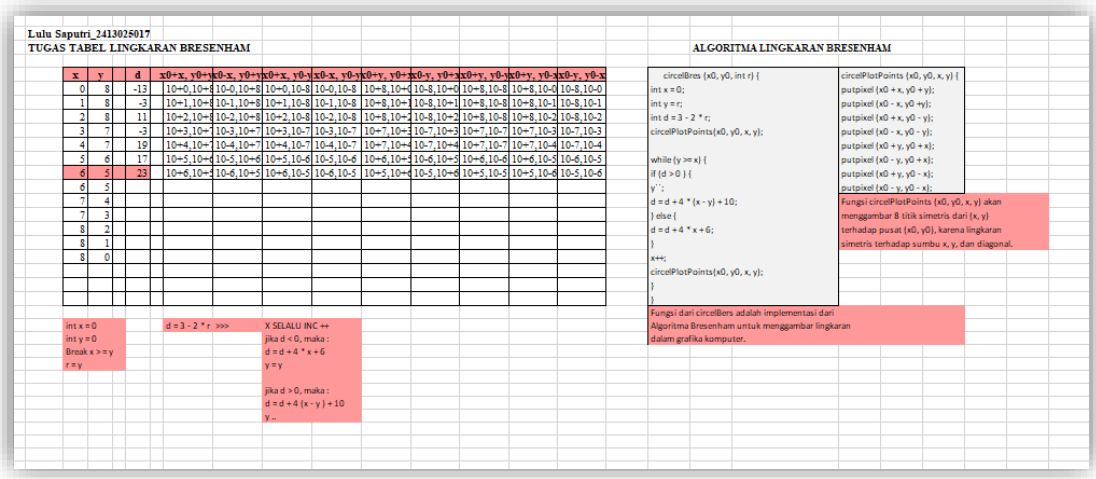
2025

A. Algoritma Lingkaran Bresenham

Algoritma Lingkaran Bresenham adalah metode berbasis perhitungan integer yang digunakan untuk menggambar lingkaran dalam grafika komputer. Algoritma ini dikembangkan oleh Jack Bresenham dan bekerja dengan cara menentukan titik-titik lingkaran menggunakan perhitungan diferensial tanpa perlu operasi trigonometri atau pembagian, sehingga lebih efisien dalam hal komputasi.

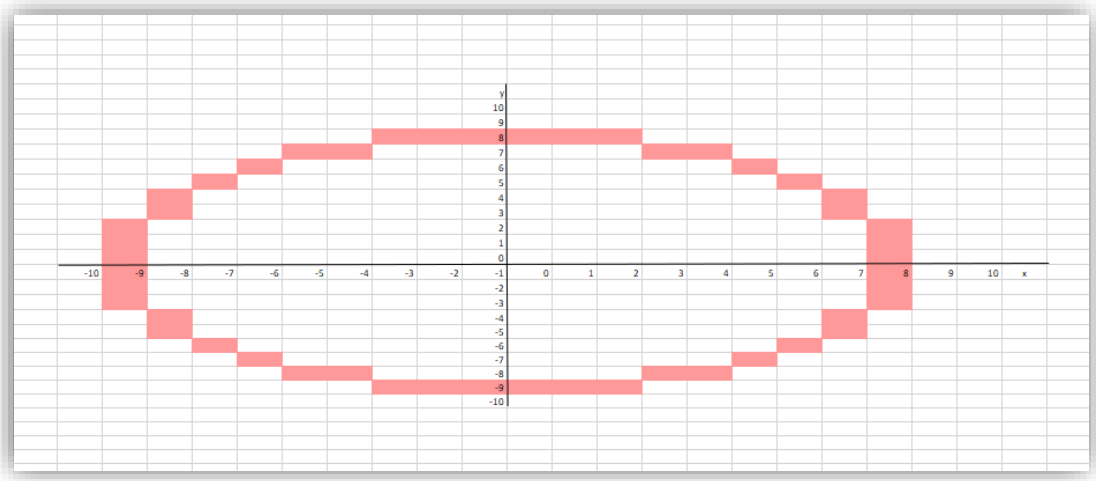
Algoritma ini hanya menghitung titik-titik pada seperdelapan lingkaran dan kemudian menggunakan sifat simetri untuk menggambar bagian lainnya. Dengan pendekatan ini, gambar lingkaran dapat dibuat dengan cepat dan akurat, terutama pada perangkat dengan keterbatasan komputasi.

a) Tabel Algoritma 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	8	-13	10+0,10+8	10-0,10+8	10+0,10-8	10-0,10-8	10+8,10+0	10-8,10+0	10+8,10-0	10-8,10-0
1	8	-3	10+1,10+8	10-1,10+8	10+1,10-8	10-1,10-8	10+8,10+1	10-8,10+1	10+8,10-1	10-8,10-1
2	8	11	10+2,10+8	10-2,10+8	10+2,10-8	10-2,10-8	10+8,10+2	10-8,10+2	10+8,10-2	10-8,10-2
3	7	-3	10+3,10+7	10-3,10+7	10+3,10-7	10-3,10-7	10+7,10+3	10-7,10+3	10+7,10-3	10-7,10-3
4	7	19	10+4,10+7	10-4,10+7	10+4,10-7	10-4,10-7	10+7,10+4	10-7,10+4	10+7,10-4	10-7,10-4
5	6	17	10+5,10+6	10-5,10+6	10+5,10-6	10-5,10-6	10+6,10+5	10-6,10+5	10+6,10-5	10-6,10-5
6	5	23	10+6,10+5	10-6,10+5	10+6,10-5	10-6,10-5	10+5,10+6	10-5,10+6	10+5,10-5	10-5,10-6
6	5									
7	4									
7	3									
8	2									
8	1									
8	0									

b) Diagram Algoritma Lingkaran Bresenham



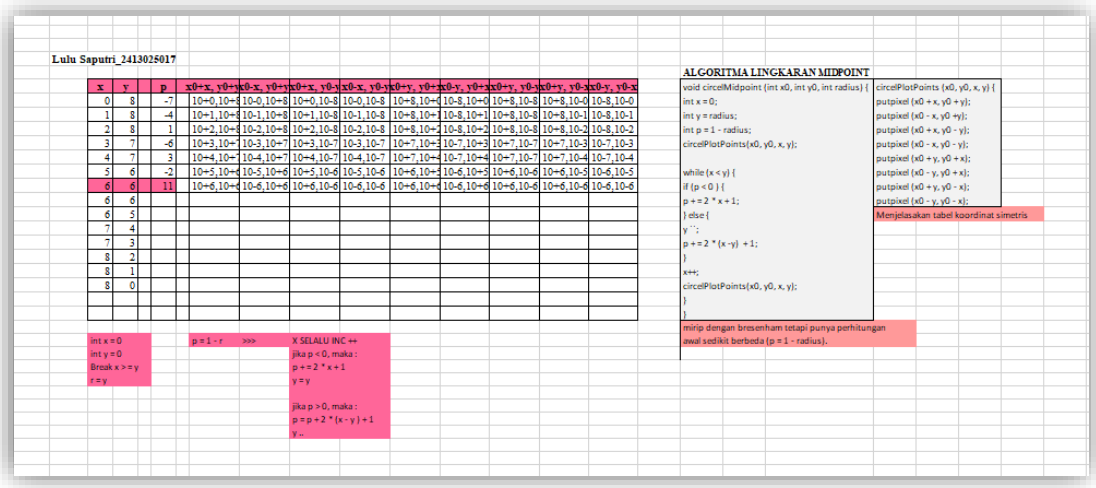
A. Algoritma Lingkaran Midpoint

Algoritma Lingkaran Midpoint adalah metode berbasis perhitungan integer yang digunakan untuk menggambar lingkaran dalam grafika komputer. Algoritma ini merupakan

pengembangan dari algoritma Bresenham, tetapi lebih sederhana karena menggunakan konsep **midpoint test** atau **uji titik tengah** untuk menentukan titik berikutnya dalam proses menggambar lingkaran.

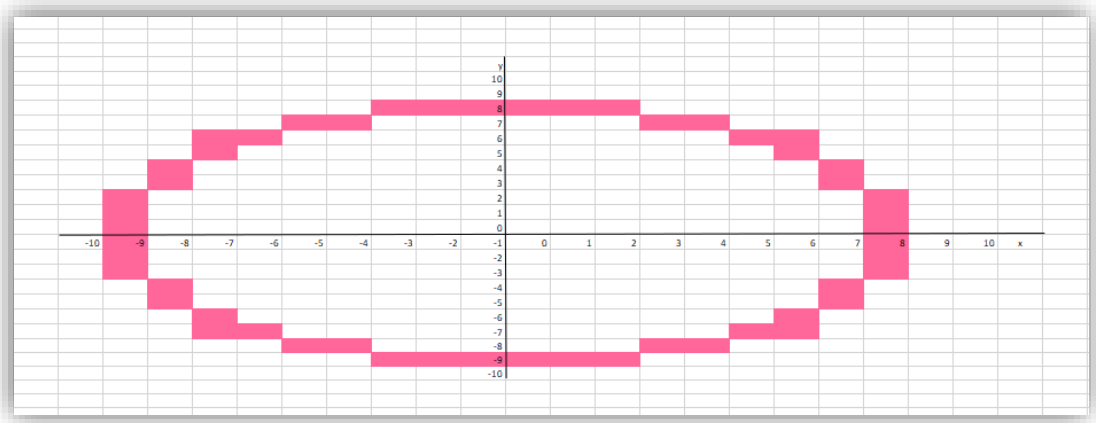
Pendekatan algoritma ini adalah dengan mengevaluasi titik tengah antara dua kemungkinan posisi berikutnya. Jika titik tengah berada di dalam lingkaran, algoritma memilih titik yang lebih dekat dengan kurva lingkaran. Dengan metode ini, Algoritma Midpoint lebih stabil, lebih mudah diimplementasikan, dan memiliki akurasi yang lebih baik dibandingkan Algoritma Bresenham.

a) Tabel Algoritma Lingkaran Midpoint



x	y		p	x0+x, y0+y	x0-x, y0+y	x0+x, y0-y	x0-x, y0-y	y0+y, x0+x	y0+y, x0-x	y0-y, x0+x	y0-y, x0-x
0	8		-7	10+0,10+8	10-0,10+8	10+0,10-8	10-0,10-8	10+8,10+0	10-8,10+0	10+8,10-0	10-8,10-0
1	8		-4	10+1,10+8	10-1,10+8	10+1,10-8	10-1,10-8	10+8,10+1	10-8,10+1	10+8,10-1	10-8,10-1
2	8		1	10+2,10+8	10-2,10+8	10+2,10-8	10-2,10-8	10+8,10+2	10-8,10+2	10+8,10-2	10-8,10-2
3	7		-6	10+3,10+7	10-3,10+7	10+3,10-7	10-3,10-7	10+7,10+3	10-7,10+3	10+7,10-7	10-7,10-3
4	7		3	10+4,10+7	10-4,10+7	10+4,10-7	10-4,10-7	10+7,10+4	10-7,10+4	10+7,10-7	10-7,10-4
5	6		-2	10+5,10+6	10-5,10+6	10+5,10-6	10-5,10-6	10+6,10+5	10-6,10+5	10+6,10-6	10-6,10-5
6	6		11	10+6,10+6	10-6,10+6	10+6,10-6	10-6,10-6	10+6,10+6	10-6,10+6	10+6,10-6	10-6,10-6
6	6										
6	5										
7	4										
7	3										
8	2										
8	1										
8	0										

b) Diagram Algoritma Lingkaran Midpoint



B. Kode Program

```
!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>2413025017_Lulu saputri</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>Radius: <input type="number" id="rad" value="80"></label>
  <label>Warna: <input type="color" id="warna" value="#fd74bf"></label>
  <button onclick="buatGambar()">Gambar Lingkaran</button>

  <br/><br/>
  <canvas id="myCanvas" width="350" height="350"></canvas>
  <canvas id="midpoint" width="350" height="350"></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);
  }
   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");
```

```
59
60      function drawmidpoint(x0, y0, r) {       let x = r;       let y = 0;       let d = r - 1;
61
62      while (x >= y) {
63        ctxmidpoint.fillRect(x0 + x, y0 + y, 1, 1);       ctxmidpoint.fillRect(x0 - x, y0 + y, 1, 1);
64        ctxmidpoint.fillRect(x0 + x, y0 - y, 1, 1);        ctxmidpoint.fillRect(x0 - x, y0 - y, 1, 1);
65          ctxmidpoint.fillRect(x0 + y, y0 + x, 1, 1);          ctxmidpoint.fillRect(x0 - y, y0 + x, 1, 1);
66
67          if (d >= 2 * y) {           d -= 2 * y + 1;           y++;
68          } else if (d < 2 * (r - x)) {           d += 2 * x - 1;           x--;         } else {
69            d += 2 * (x - y - 1);           x--;           y++;         }
70        }
71      }
72    </script>
73  </body>
74  </html>
```

C. Penjelasannya

1. Struktur Dasar HTML

```
!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>2413025017_Lulu saputri</title>
```

<!DOCTYPE html> → Menandakan bahwa ini adalah dokumen HTML5.

<html lang="en"> → Menunjukkan bahwa halaman ini menggunakan bahasa Inggris.

<meta charset="UTF-8"> → Memungkinkan penggunaan karakter khusus seperti é, ó, ü, dll.

<meta name="viewport" content="width=device-width, initial-scale=1.0"/> → Agar halaman responsif di perangkat mobile.

<title>2413025017_Lulu saputri</title> → Menampilkan judul tab browser dengan nama "2413025017_Lulu Saputri".

2. CSS untuk Styling

```
<style>     canvas {
    border: 1px solid #000000;       margin: 10px;
  }
</style>
```

canvas { border: 1px solid #000000; } → Memberikan batas hitam pada elemen <canvas>.

margin: 10px; → Memberikan jarak antara elemen <canvas> agar lebih rapi.

3. Form Input untuk Parameter Lingkaran

```
<label>X: <input type="number" id="X" value="150"></label>   <label>Y: <input type="number" id="Y" value="150"
<label>Radius: <input type="number" id="rad" value="80"></label>
<label>Warna: <input type="color" id="warna" value="#fd74bf"></label>
<button onclick="buatGambar()">Gambar Lingkaran</button>
```

<input type="number"> → Untuk menentukan koordinat **X, Y, dan radius** lingkaran.

<input type="color"> → Untuk memilih warna lingkaran.

<button onclick="buatGambar()"> → Memanggil fungsi buatGambar() untuk menggambar lingkaran.

4. Elemen <canvas> untuk Menampilkan Lingkaran

```
<canvas id="myCanvas" width="350" height="350"></canvas>
<canvas id="midpoint" width="350" height="350"></canvas>
```

Canvas pertama (id="myCanvas") digunakan untuk menggambar Algoritma Bresenham.

Canvas kedua (id="midpoint") digunakan untuk menggambar Algoritma Midpoint.

5. JavaScript untuk Menggambar Lingkaran

5.1. Inisialisasi Canvas

```
let canvas = document.getElementById("myCanvas");   let ctx = canvas.getContext("2d");
```

document.getElementById("myCanvas") → Mengambil elemen <canvas> pertama.

getContext("2d") → Untuk menggambar objek 2D pada canvas.

5.2. Fungsi untuk Menggambar Titik

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

ctx.fillStyle = warna; → Mengatur warna titik.

ctx.fillRect(x, y, 1, 3); → Menggambar titik persegi panjang dengan ukuran 1x3 pixel.

5.3. Fungsi untuk Menggambar Titik Simetris

```
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);
}
```

Karena **lingkaran simetris**, kita hanya perlu menghitung **seperdelapan** titik, lalu **merefleksikan** ke 7 bagian lainnya.

5.4. Fungsi Algoritma Bresenham

```
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++;
}
}
```

Algoritma Bresenham digunakan untuk menggambar lingkaran tanpa menggunakan operasi trigonometri.

Nilai keputusan d menentukan apakah titik berikutnya ada di dalam atau luar lingkaran.

5.5. Fungsi untuk Menjalankan Gambar

```
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);
}
```

Menghapus gambar lama sebelum menggambar ulang.

Mengambil nilai input pengguna.

Memanggil kedua algoritma untuk menggambar lingkaran.

5.6. Inisialisasi Canvas untuk Algoritma Midpoint

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

Sama seperti canvas pertama, ini digunakan untuk menggambar lingkaran dengan Algoritma Midpoint.

5.7. Fungsi Algoritma Midpoint

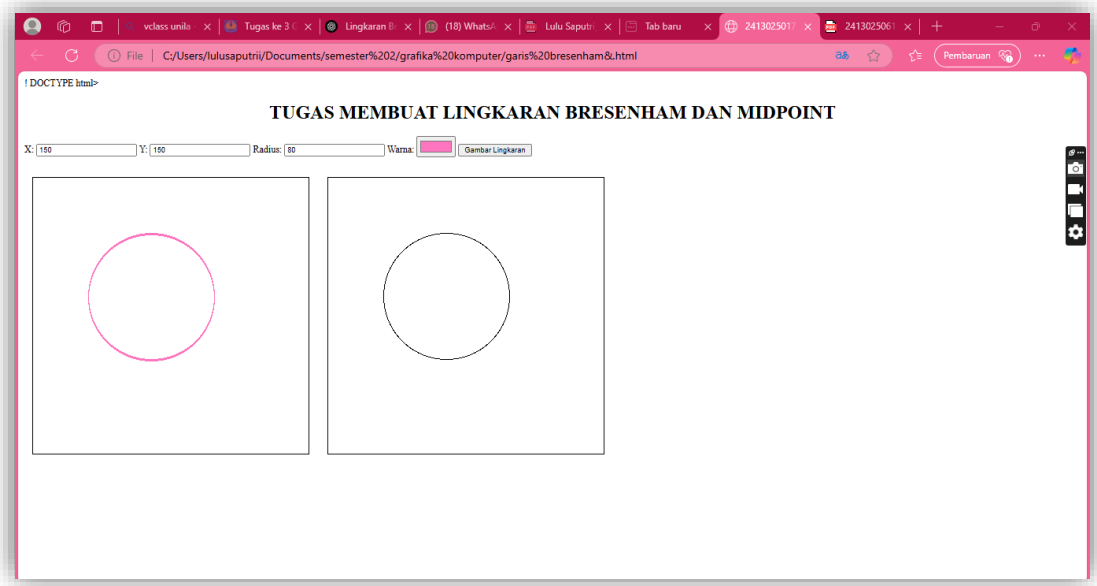
```
function drawmidpoint(x0, y0, r) {      let x = r;      let y = 0;      let d = r - 1;

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);

    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++;          }
  }
}
```

Algoritma ini lebih efisien karena hanya menggunakan penjumlahan dan perkalian integer, tanpa trigonometri.

D. Output yang dihasilkan



https://youtu.be/Adw3YTrRO6I?si=-V4FV2wk-ABcbWRT