MODUL 3

KOMPUTER GRAFIK 2D
LINGKARAN, ELLIPS DAN ATRIBUTE GARIS

D4 TEKNIK INFORMATIKA
JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA
POLITEKNIK NEGERI BANDUNG

MAHASISWA 058 | KOMPUTER GRAFIK | FEBRUARI, 26 2025

CONTENTS

LINGKARAN

Lingkaran merupakan bentuk dasar yang biasa digunakan untuk membuat gambar atau objek yang kompleks, seperti dekorasi, batik, dan lain-lain. Pada paket library komputer grafik, sebagian atau lingkaran penuh dapat dibuat menggunakan sebuah prosedur. Secara general, prosedur tersebut dapat menghasilkan garis dan ellips.

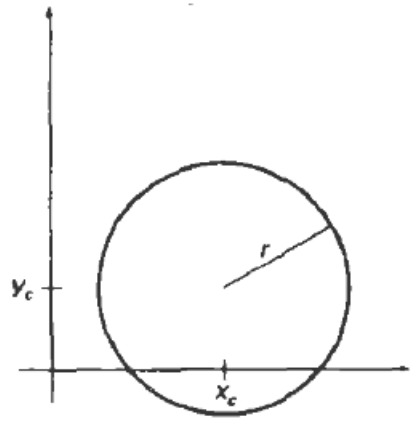


Figure 3-12
Circle with center coordinates (x_c, y_c) and radius r.

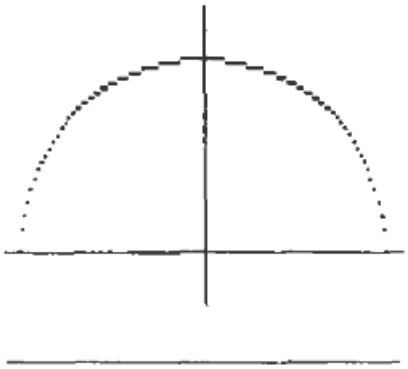


Figure 3-13
Positive half of a circle plotted with Eq. 3-25 and with $(x_c, y_c) = (0, 0)$.

Persamaan Lingkaran

$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

$$y = y_c \pm \sqrt{(r^2 - (x_c - x)^2)}$$

Persamaan Lingkaran Polar Coordinates

$$x = x_c + rcos\theta$$

$$y = y_c + rsin\theta$$

Step size :

$$\frac{1}{r}$$

Komputasi lingkaran dapat direduksi karena lingkaran adalah bentuk yang simetris. Setiap kuadran meliliki bentuk bagian lingkaran yang sama.

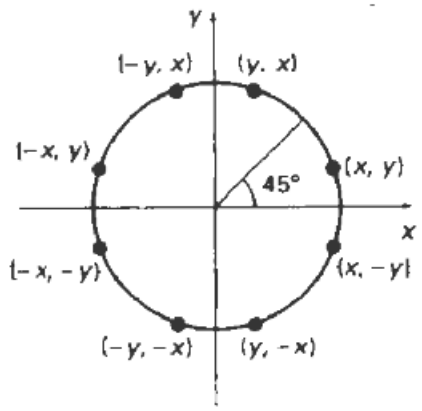


Figure 3-14
Symmetry of a circle. Calculation of a circle point (x, y) in one octant yields the circle points shown for the other seven octants.

Algoritma paling efisien berdasarkan kalkulasi incremental dari decision parameter, seperti bersenham line algorithm, yang hanya membutuhkan operasi integer sederhana. Algoritma line bersenham diadaptasi untuk pembentukan lingkaran dengan melakukan setup decision parameter untuk mencari pixel terdekat untuk mendapatkan lingkar untuk setiap sampling step.

Metode untuk mendapatkan jarak secara langsung pada lingkaran, adalah dengan melakukan pengecekan posisi tengah dari dua pixel untuk menentukan apakah titik "midpoint" ini ada pada dalam atau luar lingkaran. Metode midpoint dapat diaplikasikan untuk bentuk-bentuk conics.

MIDPOINT CIRCLE ALGORITHM

Seperti pada line bersenham, tujuan dari algoritma untuk mendapatkan sampling titik dan menentukan pixel terdekat pada setiap step. Pada lingkaran, untuk setiap r dan titik posisi center (x_c, y_c). Algoritma dimulai dengan mengkalkulasi posisi pixel dalam jalur lingkaran yang memiliki titik tengah origin $(0,0)$. Lalu untuk setiap posisi (x, y) yang dikalkulasi dipindahkan pada posisi yang benar dengan menambahkan $x_c$pada x dan y_c pada y.

Pada bagian lingkaran kuadran I mulai dari $x = 0$, $x = y$ nilai slope bervariasi dari 0 sampai -1 . Maka pergerakan unit step sesuai arah x positif dan menggunakan decision parameter untuk menentukan dua posisi yang mungkin lebih dekat dengan jalur lingkaran untuk setiap langkah.

$$f_{circ}(x,y)=x^2+y^2-r^2$$

Jika point ada didalam interior lingkaran , fungsi lingkaran negative, dan sebaliknya.

$$f_{circ}(x,y) = \begin{cases} 0, if\,(x,y)\,is\,inside\,the\,boundary \\ 0, if\,(x,y)\,is\,on\,the\,boundary \\ 0, if\,(x,y)\,is\,outside\,the\,boundary \end{cases}$$

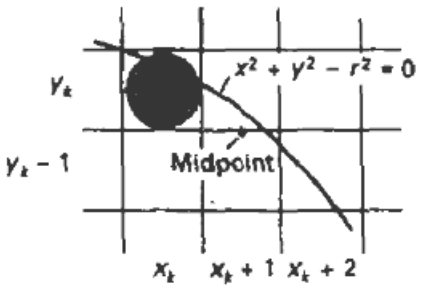


Figure 3-15
Midpoint between candidate pixels at sampling position x_k+1 along a circular path.

$$p_k = f_{\circ}\left(x_k+1, y_k-\frac{1}{2}\right)$$

$$(x+1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2$$

Jika $p_k < 0$ midpoint terletak pada bagian dalam lingkaran dan y_k lebih dekat pada batas lingkaran. Sebaliknya $y_k - 1$ lebih dekat pada batas lingkaran.

Decision parameter selanjutnya didapatkan menggunakan kalkulasi incremental integer, yaitu $x_{k+1} + 1 = x_k + 2$

$$p_{k+1} = f_{\circ}\left(x_{k+1}+1, y_{k+1}-\frac{1}{2}\right)$$

$$[(x_k + 1) + 1]^2 + \left(y_{k+1} - \frac{1}{2}\right)^2 - r^2$$

$$p_{k+1} = p_k + 2(x_k + 1) + \left(y_{k+1}^2 - y_k^2\right) - (y_{k+1} - y_k) + 1$$

Dimana, y_{k+1} antara y_k atau $y_k - 1$, bergantung pada tanda dari p_k

Evaluasi $2x_{k+1}$ dan $2y_{k+1}$ didapatkan secara inkemental menggunakan.

$$2x_{k+1} = 2x_k + 2$$

$$2y_{k+1} = 2y_k - 2$$

Pada titik awal $(0, r)$, decision parameter pertama didapatkan dengan melakukan evaluasi pada fungsi lingkaran dengan nilai $(x_0, y_0) = (0, r)$:

$$p_o = f_{\circ}\left(1, r-\frac{1}{2}\right)$$

$$1 + \left(r - \frac{1}{2}\right)^2 - r^2$$

$$\frac{5}{4} - r$$

Midpoint Circle Algorithm

1. Input radius r and circle center (x_c, y_c), and obtain the first point on the circumference of a circle centered on the origin as

$$(x_0, y_0) = (0, r)$$

2. Calculate the initial value of the decision parameter as

$$p_0 = \frac{5}{4} - r$$

3. At each x_k position, starting at $k = 0$, perform the following test: If $p_k < 0$, the next point along the circle centered on $(0, 0)$ is (x_{k+1}, y_k) and

$$p_{k+1} = p_k + 2x_{k+1} + 1$$

   Otherwise, the next point along the circle is $(x_k + 1, y_k - 1)$ and

$$p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$$

   where $2x_{k+1} = 2x_k + 2$ and $2y_{k+1} = 2y_k - 2$.
4. Determine symmetry points in the other seven octants.
5. Move each calculated pixel position (x, y) onto the circular path centered on (x_c, y_c) and plot the coordinate values:

$$x = x + x_c, \qquad y = y + y_c$$

6. Repeat steps 3 through 5 until $x \geq y$.

```
#include "device.h"

void circleMidpoint (int xCenter, int yCenter, int radius)
{
  int x = 0;
  int y = radius;
  int p = 1 - radius;
  void circlePlotPoints (int, int, int, int);

  /* Plot first set of points */
  circlePlotPoints (xCenter, yCenter, x, y);

  while (x < y) {
    x++;
    if (p < 0)
      p += 2 * x + 1;
    else {
      y--;
      p += 2 * (x - y) + 1;
    }
    circlePlotPoints (xCenter, yCenter, x, y);
  }
}

void circlePlotPoints (int xCenter, int yCenter, int x, int y)
{
  setPixel (xCenter + x, yCenter + y);
  setPixel (xCenter - x, yCenter + y);
  setPixel (xCenter + x, yCenter - y);
  setPixel (xCenter - x, yCenter - y);
  setPixel (xCenter + y, yCenter + x);
  setPixel (xCenter - y, yCenter + x);
  setPixel (xCenter + y, yCenter - x);
  setPixel (xCenter - y, yCenter - x);
}
```

Implementasi Circle Algorithm menggunakan C# pada bentukdasar.cs
Silakan lengkapi

Contoh Pemanggilan Lingkaran yang akan digunakan pada karya1

ELLIPS

Untuk menggambarkan sebuah ellips, kita dapat mengadopsi pola penggambaran lingkaran. Ellips bisa juga disebut lingkaran yang pipih, dengan modifikasi lingkaran yang memiliki dimensi vertikal dan horizontal yang berbeda atau disebut major dan minor axes.

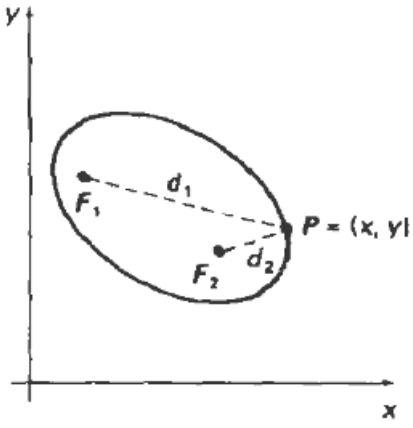


Figure 3-17
Ellipse generated about foci F_1 and F_2.

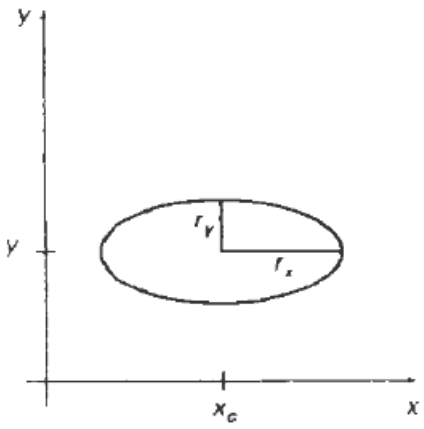


Figure 3-18
Ellipse centered at (x_c, y_c) with semimajor axis r_x and semiminor axis r_y.

Ellips dapat didefinisikan sebagai Kumpulan dari titik yang sedemikian hingga jumlah dari jarak antara dua titik tetap (foci atau fixed position) adalah sama untuk seluruh titik. Jika jarak dari dua foci dari sebuah titik $P = (x, y)$ pada sebuah ellips diberi label d_1 dan d_2, maka persamaan ellips dapat dinyatakan sebagai:

$$d_1 + d_2 = constant$$

Eksprsi dari jarak d1 dan d2 dalam foci $F_1 = (x_1, y_1)$ dan $F_2 = (x_2, y_2)$ maka didapatkan :

$$\sqrt{((x - x_1)^2 + (y - y_1)^2)} + \sqrt{((x - x_2)^2 + (y - y_2)^2)} = constant$$

Persamaan ellips lain ketika posisi mayor axes dan minor axes pada posisi standar dapat dinyatakan sebagai:

$$\left(\frac{x - x_c}{r_x}\right)^2 + \left(\frac{y - y_c}{r_y}\right)^2 = 1$$

Dengan menggunakan koordinat polar ellips pada posisi standar dapat dinyatakan dengan:

$$x = x_c + r_x cos\theta$$

$$y = y_c + r_y sin\theta$$

MIDPOINT ELLIPS ALGORITHM

Seperti pada penggambaran lingkaran, Proses sampling pada satu unit koordinat (x atau y) dan menentukan nilai integer terdekat yang sesuai dengan jalur garis dari koordinat lain.

Metode midpoint ellips diaplikasikan pada quadran I untuk dua bagian yaitu Region I (Mayor axes) dan Region II (Minor Axis)

Steps atau iterasi dilakukan pada arah x jika $m < 1$ sebaliknya iterasi dilakukan pada arah y jika $m > 1$. Titik awal pada posisi $(0, r_y)$ dan arah mengikuti jarum jam atau (CW) pada kuadran I ellips. Perubahan unit step dari x ke unit step y

ketika $m < -1$. Lalu, mirip dengan lingkaran ada property simetris pada penggambaran ellips, secara parallel untuk dua region.

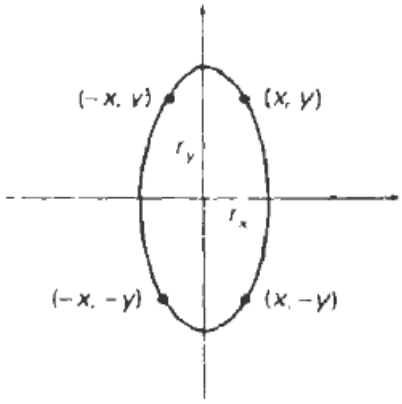


Figure 3-19
Symmetry of an ellipse
Calculation of a point (x, y) in one quadrant yields the ellipse points shown for the other three quadrants.

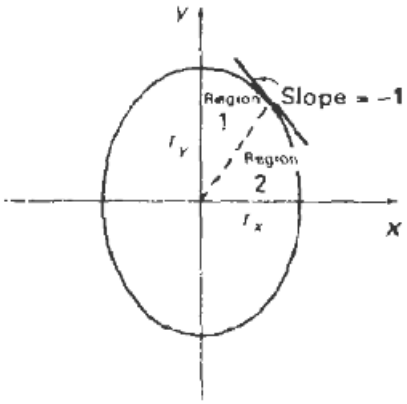


Figure 3-20
Ellipse processing regions. Over region 1, the magnitude of the ellipse slope is less than 1; over region 2, the magnitude of the slope is greater than 1.

Didefinisikan fungsi ellips pada sebuah titik $(x_c, y_c) = (0,0)$ sebagai berikut:

$$f_{ellipse}(x,y) = r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2$$

Yang mengikuti syarat-syarat sebagai berikut:

$$f_{ellipse}(x,y)\begin{cases} 0, if(x,y) isinside the ellipse boundary \\ 0, if(x,y) is on the ellipse boundary \\ 0, if(x,y) is outside the ellipse boundary \end{cases}$$

fungsi ellipse berfungsi sebagai decision parameter pada algoritma midpoint. Untuk setiap posisi sampling, pemilihan pixel selanjutnya pada jalur ellips tergantung pada kondisi dari fungsi ellips yang dievaluasi pada midpoint untuk dua kandidat pixel.

Ellipse slope atau m didapatkan dari :

$$\frac{dy}{dx} = \frac{-2r_y^2}{2r_x^2}$$

Pada boundary antara region 1 dan region 2, nilai dari $\frac{dy}{dx} = -1$ dan $2r_y^2 x = 2r_x^2 y$

Maka, pergantian region terjadi saat

$$2r_y^2 x \geq 2r_x^2 y$$

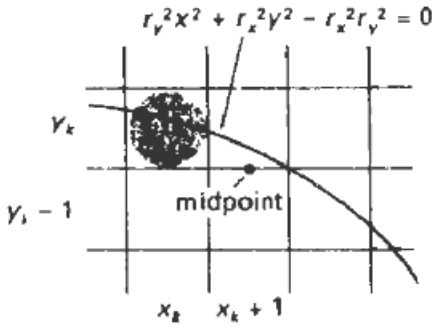


Figure 3-21
Midpoint between candidate pixels at sampling position $x_k + 1$ along an elliptical path.

$$p1_k = f_{ellipse}\left(x_k + 1, y_k - \frac{1}{2}\right)$$

$$r_y^2(x_k + 1)^2 + r_x^2\left(y_k - \frac{1}{2}\right) - r_x^2 r_y^2$$

$$y_{k+1} = \begin{cases} jikap1_k < 0, y_k \\ jikap1_k > 0, y_k - 1 \end{cases}$$

$$p1_{k+1} = f_{ellipse}\left(x_{k+1} + 1, y_{k+1} - \frac{1}{2}\right)$$

$$r_y^2[(x_k + 1) + 1]^2 + r_x^2\left(y_{k+1} - \frac{1}{2}\right)^2 - r_x^2 r_y^2$$

Atau di expand sbb:

$$p1_{k+1} = p1_k + 2r_y^2(x_k + 1) + r_y^2 + r_x^2\left[\left(y_{k+1} - \frac{1}{2}\right)^2 - \left(y_k - \frac{1}{2}\right)^2\right]$$

Subsitusi y_{k+1}antara y_k atau $y_k - 1$ bergantung pada tanda $p1_k$, decision parameter di inkrement sebagai berikut:

$$incerement = \begin{cases} 2r_y^2 x_{k+1} + r_y^2, jikap1_k < 0 \\ 2r_y^2 x_{k+1} + r_y^2 - 2r_x^2 y_{k+1}, jikap1_k > 0 \end{cases}$$

Pada region I nilai pertama pada decision paramater didapatakn dengan mengevaluasi fungsi ellips pada posisi awal $(x_0, y_0) = (0, r_y)$

$$p1_0 = f_{ellipse}\left(1, r_y - \frac{1}{2}\right)$$

$$r_y^2 + r_x^2\left(r_y - \frac{1}{2}\right)^2 - r_x^2 r_y^2$$

Atau

$$p1_0 = r_y^2 - r_x^2 r_y^2 + \frac{1}{4}r_x^2$$

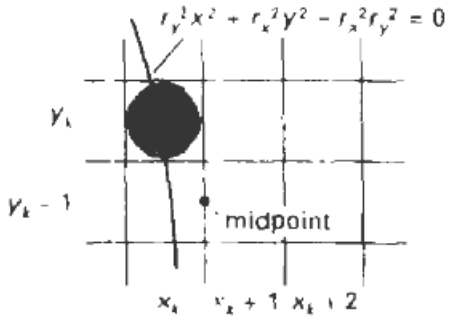


Figure 3-22
Midpoint between candidate pixels at sampling position $y_k - 1$ along an elliptical path.

Untuk region 2, sampling unit step terjadi pada arah y negatif, dan midpoint diambil diantara dua titik kandidat horizontal pixel untuk setiap step. Maka decision parameter dievaluasi sbb:

$$p2_k = f_{ellipse}\left(x_k + \frac{1}{2}, y_k - 1\right)$$

$$r_y^2\left(x_k + \frac{1}{2}\right)^2 + r_x^2(y_k - 1)^2 - r_x^2 r_y^2$$

$$x_{k+1} = \begin{cases} jikap2_k > 0, x_k \\ jikap2_k \leq 0, x_{k+1} \end{cases}$$

$$p2_{k+1} = f_{ellipse}\left(x_{k+1} + \frac{1}{2}, y_{k+1} - 1\right)$$

$$r_y^2\left[x_{k+1} + \frac{1}{2}\right]^2 + r_x^2[(y_k - 1)| \quad |-1]^2 - r_x^2 r_y^2$$

Atau

$$p2_{k+1} = p2_k + 2r_x^2(y_k - 1) + r_x^2 + r_y^2\left[\left(x_{k+1} + \frac{1}{2}\right)^2 - \left(x_k + \frac{1}{2}\right)^2\right]$$

Para region 2, nilai pertama adalah (x_0, y_o) diambil berdasarkan posisi terakhir dari region 1 dan dicision parameter pertama region ke 2 adalah.

$$p2_0 = f_{ellipse}\left(x_0 + \frac{1}{2}, y_0 - 1\right)$$

$$r_y^2\left(x_0 + \frac{1}{2}\right)^2 + r_x^2(y_0 - 1)^2 - r_x^2 r_y^2$$

Untuk menyederhanakan perhitungan, titik pertama dapat diambil pada $(r_x, 0)$. Unit step akan berjalan dengan arah y positif sampai dengan posisi terakhir dari Region 1.

Midpoint Ellipse Algorithm

1. Input r_x, r_y, and ellipse center (x_c, y_c), and obtain the first point on an ellipse centered on the origin as

$$(x_0, y_0) = (0, r_y)$$

2. Calculate the initial value of the decision parameter in region 1 as

$$p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$$

3. At each x_k position in region 1, starting at $k = 0$, perform the following test: If $p1_k < 0$, the next point along the ellipse centered on (0, 0) is (x_{k+1}, y_k) and

$$p1_{k+1} = p1_k + 2r_y^2 x_{k+1} + r_y^2$$

Otherwise, the next point along the circle is $(x_k + 1, y_k - 1)$ and

$$p1_{k+1} = p1_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2$$

with

$$2r_y^2 x_{k+1} = 2r_y^2 x_k + 2r_y^2, \qquad 2r_x^2 y_{k+1} = 2r_x^2 y_k - 2r_x^2$$

and continue until $2r_y^2 x \geq 2r_x^2 y$.

and continue until $2r_y^2 x \geq 2r_x^2 y$.

4. Calculate the initial value of the decision parameter in region 2 using the last point (x_0, y_0) calculated in region 1 as

$$p2_0 = r_y^2 \left(x_0 + \frac{1}{2}\right)^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$$

5. At each y_k position in region 2, starting at $k = 0$, perform the following test: If $p2_k > 0$, the next point along the ellipse centered on (0, 0) is $(x_k, y_k - 1)$ and

$$p2_{k+1} = p2_k - 2r_x^2 y_{k+1} + r_x^2$$

Otherwise, the next point along the circle is $(x_k + 1, y_k - 1)$ and

$$p2_{k+1} = p2_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_x^2$$

using the same incremental calculations for x and y as in region 1.

6. Determine symmetry points in the other three quadrants.
7. Move each calculated pixel position (x, y) onto the elliptical path centered on (x_c, y_c) and plot the coordinate values:

$$x = x + x_c \qquad y = y + y_c$$

8. Repeat the steps for region 1 until $2r_y^2 x \geq 2r_x^2 y$.

```
#include "device.h"

#define ROUND(a) ((int)(a+0.5))

void ellipseMidpoint (int xCenter, int yCenter, int Rx, int Ry)
{
  int Rx2 = Rx*Rx;
  int Ry2 = Ry*Ry;
  int twoRx2 = 2*Rx2;
  int twoRy2 = 2*Ry2;
  int p;
  int x = 0;
  int y = Ry;
  int px = 0;
  int py = twoRx2 * y;
  void ellipsePlotPoints (int, int, int, int);

  /* Plot the first set of points */
  ellipsePlotPoints (xCenter, yCenter, x, y);

  /* Region 1 */
  p = ROUND (Ry2 - (Rx2 * Ry) + (0.25 * Rx2));
  while (px < py) {
    x++;
    px += twoRy2;
    if (p < 0)
      p += Ry2 + px;
    else {
      y--;
      py -= twoRx2;
      p += Ry2 + px - py;
    }
    ellipsePlotPoints (xCenter, yCenter, x, y);
  }

  /* Region 2 */
  p = ROUND (Ry2*(x+0.5)*(x+0.5) + Rx2*(y-1)*(y-1) - Rx2*Ry2);
  while (y > 0) {
    y--;
    py -= twoRx2;
    if (p > 0)
      p += Rx2 - py;
    else {
      x++;
      px += twoRy2;
      p += Rx2 - py + px;
```

```
        }
        ellipsePlotPoints (xCenter, yCenter, x, y);
    }
}

void ellipsePlotPoints (int xCenter, int yCenter, int x, int y)
{
    setPixel (xCenter + x, yCenter + y);
    setPixel (xCenter - x, yCenter + y);
    setPixel (xCenter + x, yCenter - y);
    setPixel (xCenter - x, yCenter - y);
}
```

Implementasi Ellips Algorithm pada C# pada bentukdasar.cs
Silakan lengkapi

Contoh Pemanggilan Ellips pada karya1

TASK PRAKTIKUM

TASK 1: IMPLEMENTASI MIDPOINT LINGKARAN DAN MIDPOINT GARIS DAN LENGKAPI BENTUK DASAR

1. Lanjutkan kode minggu lalu dan rename menjadi [KG2025_2A_D4_2023]_Modul3_000, modifikasi scene karya1 dan karya 2 sesuai dengan tugas modul 3.
2. Amati Algoritma Midpoint untuk menggambar lingkaran dan ellips
3. Implementasi Algoritma tersebut dan buatlah fungsi lingkaran / circle dan ellips / ellipse pada class bentukdasar.cs
4. Karya 2 pada pertemuan minggu lalu dipindahkan menjadi karya 1, selain Persegi, Persegi Panjang, Segitiga Siku-Siku, dan Trapesium Siku-Siku, tambahkan Lingkaran dan Ellips di sembarang kuadran

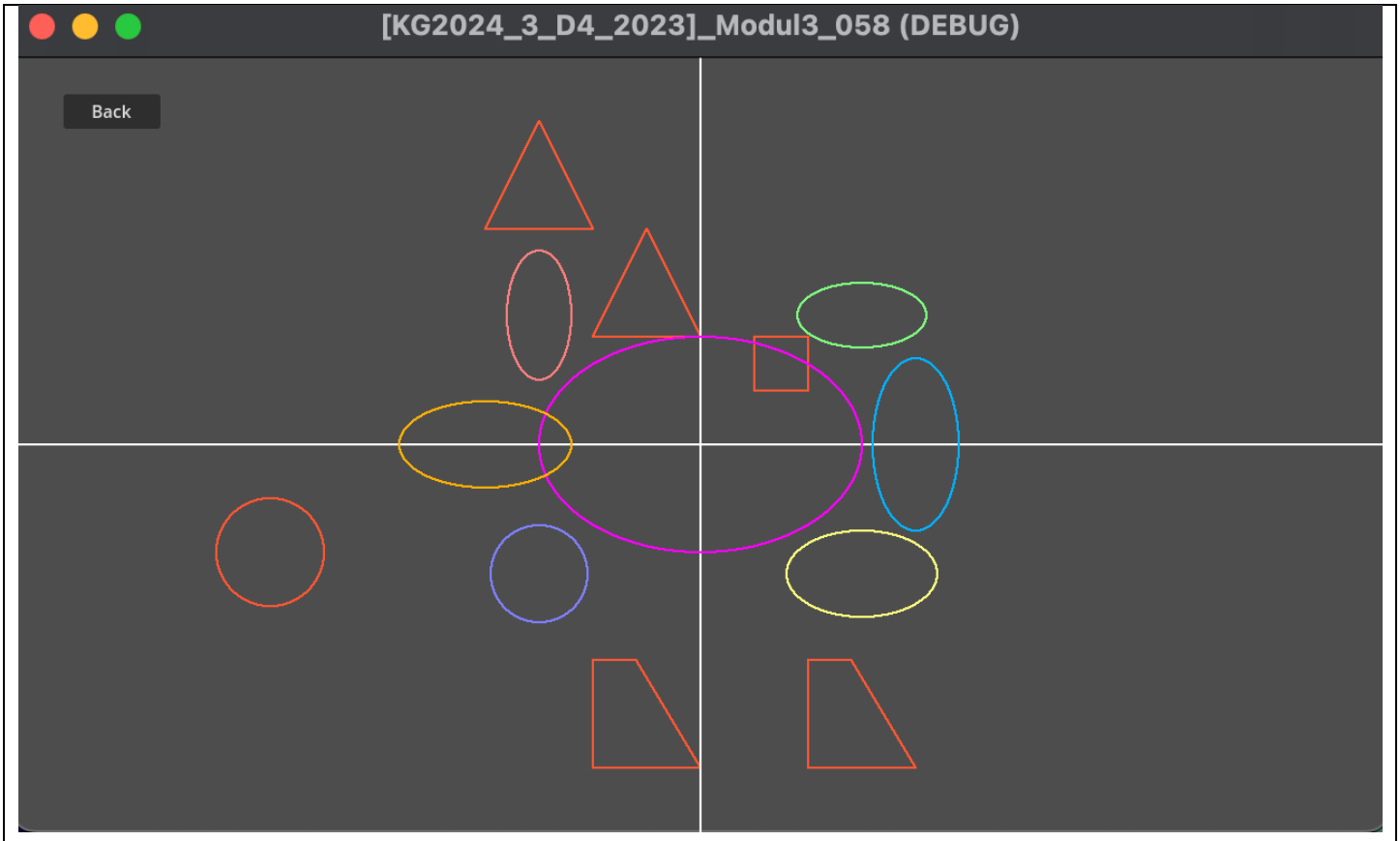
Lesson Learnt (Code, Print Screen Hasil Karya, dan Komentar)

Karya 1
 	
namespace Godot;
 	
using Godot;
using System;
using System.Collections.Generic;
 	
public partial class karya1 : Node2D
{
private primitif _primitif = new primitif();
private bentukdasar _bentukdasar;
private const int MarginLeft = 50;
private const int MarginTop = 50;

```
private int WorldOriginX; // Tidak lagi `const`
private int WorldOriginY;

public override void _Ready()
{
	GD.Print("karya1 _Ready() dipanggil");
	_bentukdasar = new bentukdasar();

	if (_bentukdasar == null)
		GD.PrintErr("ERROR: _bentukdasar masih null!");

	// Hitung titik tengah layar secara dinamis
	WorldOriginX = (int)(GetViewportRect().Size.X / 2);
	WorldOriginY = (int)(GetViewportRect().Size.Y / 2);

	QueueRedraw();
}

private void DrawEllipses()
{
	// Get screen dimensions
	Vector2 WindowSize = GetViewportRect().Size;
	float centerX = WindowSize.X / 2;
	float centerY = WindowSize.Y / 2;

	// Draw a centered ellipse
	var ellipsePoints = _bentukdasar.Ellips(centerX, centerY, 150, 100);
	PutPixelAll(ellipsePoints, new Color(1, 0, 1)); // Magenta

	// Draw ellipses in different positions with different dimensions
	// Horizontal ellipse
	var horizontalEllipse = _bentukdasar.Ellips(centerX - 200, centerY, 80, 40);
	PutPixelAll(horizontalEllipse, new Color(1, 0.7f, 0)); // Orange

	// Vertical ellipse
	var verticalEllipse = _bentukdasar.Ellips(centerX + 200, centerY, 40, 80);
	PutPixelAll(verticalEllipse, new Color(0, 0.7f, 1)); // Light blue

	// Draw smaller ellipses in each quadrant
	float quadrantOffsetX = 150;
	float quadrantOffsetY = 120;

	var ellipse1 = _bentukdasar.Ellips(centerX + quadrantOffsetX, centerY - quadrantOffsetY, 60, 30);
	var ellipse2 = _bentukdasar.Ellips(centerX - quadrantOffsetX, centerY - quadrantOffsetY, 30, 60);
	var ellipse3 = _bentukdasar.Ellips(centerX - quadrantOffsetX, centerY + quadrantOffsetY, 45, 45);
	var ellipse4 = _bentukdasar.Ellips(centerX + quadrantOffsetX, centerY + quadrantOffsetY, 70, 40);

	PutPixelAll(ellipse1, new Color(0.5f, 1, 0.5f)); // Light green
	PutPixelAll(ellipse2, new Color(1, 0.5f, 0.5f)); // Light red
	PutPixelAll(ellipse3, new Color(0.5f, 0.5f, 1)); // Light blue
	PutPixelAll(ellipse4, new Color(1, 1, 0.5f)); // Light yellow
}
```

```
	public override void _Draw()
	{
		Vector2 WindowSize = GetViewportRect().Size;
		int ScreenWidth = (int)WindowSize[0];
		int ScreenHeight = (int)WindowSize[1];
		int MarginRight = ScreenWidth - MarginLeft;
		int MarginBottom = ScreenHeight - MarginTop;

		//MarginPixel(MarginLeft, MarginTop, MarginRight, MarginBottom);
		DrawShapes();
		DrawAxes();
		DrawEllipses();

	}

	private void DrawShapes()
	{
		Godot.Color colorShape = new Godot.Color("#FF5733"); // Warna bentuk

		// Kuadran I (kanan atas)
		List<Vector2> persegi1 = _bentukdasar.Persegi(WorldOriginX + 50, WorldOriginY - 100, 50);

		// Kuadran II (kiri atas)
		List<Vector2> segitiga = _bentukdasar.SegitigaSamaKaki(WorldOriginX - 100, WorldOriginY - 100,
100, 100);
		List<Vector2> segitiga2 = _bentukdasar.SegitigaSamaKaki(WorldOriginX - 200, WorldOriginY - 200,
100, 100);
		// Kuadran III (kiri bawah)
		List<Vector2> lingkaran = _bentukdasar.Lingkaran(WorldOriginX - 400, WorldOriginY + 100, 50);
		// Kuadran IV (kanan bawah)
		List<Vector2> trapesium = _bentukdasar.TrapesiumSikuSiku(WorldOriginX + 100, WorldOriginY +
300, 100,40,100);
		List<Vector2> trapesium2 = _bentukdasar.TrapesiumSikuSiku(WorldOriginX - 100, WorldOriginY +
300, 100,40,100);
		PutPixelAll(persegi1, colorShape);
		PutPixelAll(segitiga, colorShape);
		PutPixelAll(segitiga2, colorShape);
		PutPixelAll(lingkaran, colorShape);
		PutPixelAll(trapesium, colorShape);
		PutPixelAll(trapesium2, colorShape);
	}

		//private void MarginPixel(int MarginLeft, int MarginTop, int MarginRight, int MarginBottom){
		//Godot.Color color = new Godot.Color("#32CD30");
		//var margin = _bentukDasar.Margin(MarginLeft, MarginTop, MarginRight, MarginBottom);
		//PutPixelAll(margin, color);
	//}

	private void PutPixel(float x, float y, Godot.Color? color = null)
	{
		if (x < 0 || y < 0 || x > GetViewportRect().Size.X || y > GetViewportRect().Size.Y)
		{
			GD.PrintErr($"Warning: Titik di luar layar ({x}, {y})");
			return;
		}
```

```
                Godot.Color actualColor = color ?? Godot.Colors.White;
                DrawCircle(new Vector2(x, y), 1, actualColor);
        }

        private void DrawAxes()
        {
                Godot.Color axisColor = new Godot.Color(1, 1, 1); // Warna putih untuk sumbu

                // Pastikan `_primitif` telah diinisialisasi sebelum digunakan
                if (_primitif == null)
                {
                        GD.PrintErr("ERROR: _primitif masih null!");
                        return;
                }

                // Garis sumbu X (horizontal, dari kiri ke kanan)
                List<Vector2> axisX = _primitif.LineDDA(0, WorldOriginY, (int)GetViewportRect().Size.X,
WorldOriginY);

                // Garis sumbu Y (vertikal, dari atas ke bawah)
                List<Vector2> axisY = _primitif.LineDDA(WorldOriginX, 0, WorldOriginX,
(int)GetViewportRect().Size.Y);

                PutPixelAll(axisX, axisColor);
                PutPixelAll(axisY, axisColor);
        }

        private void PutPixelAll(List<Vector2> dots, Godot.Color? color = null)
        {
                foreach (Vector2 point in dots)
                {
                        PutPixel(point.X, point.Y, color);
                }
        }

        public override void _Process(double delta)
        {
                QueueRedraw();
        }
}
```



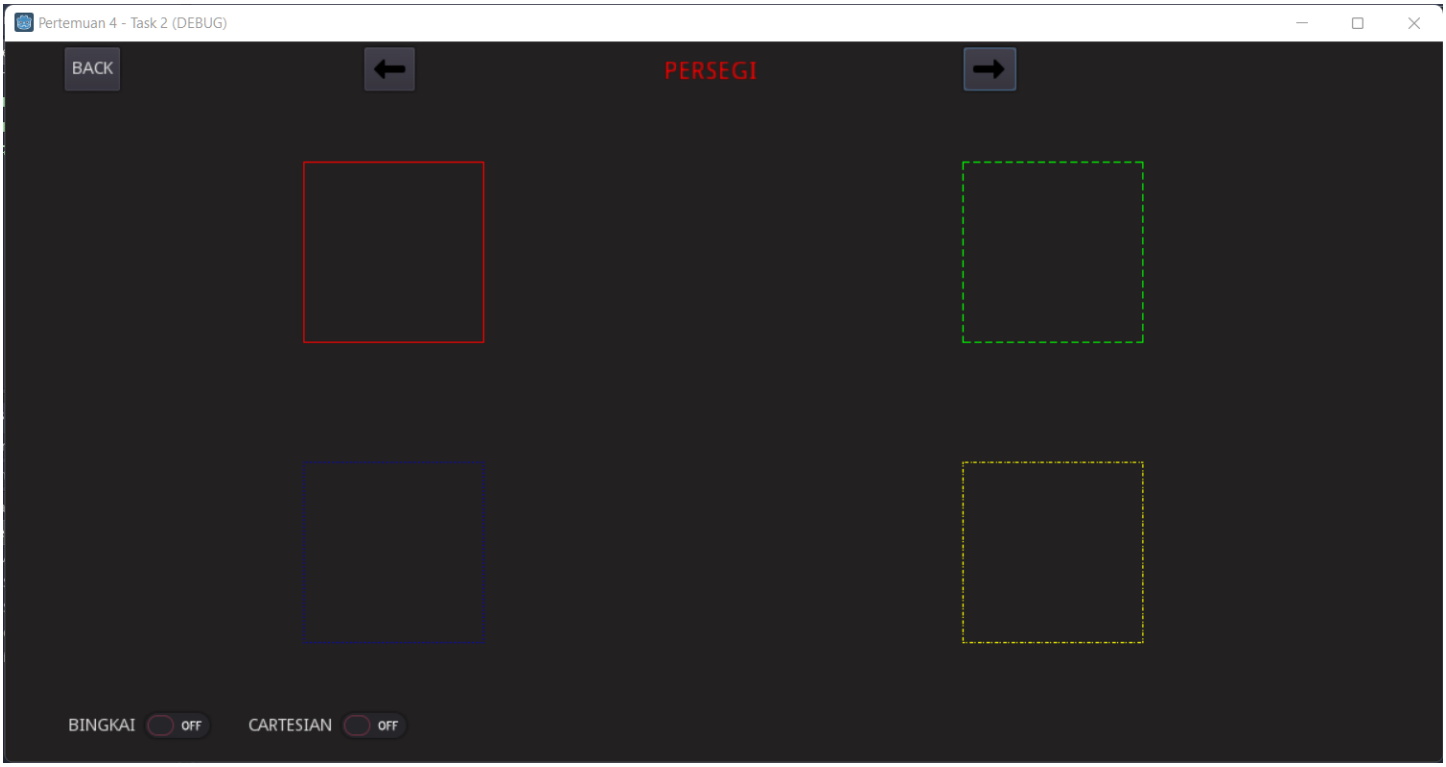
Dari Karya 1 ini, saya menampilkan bermacam shape dari elips, trapezium hingga persegi, saya belajar

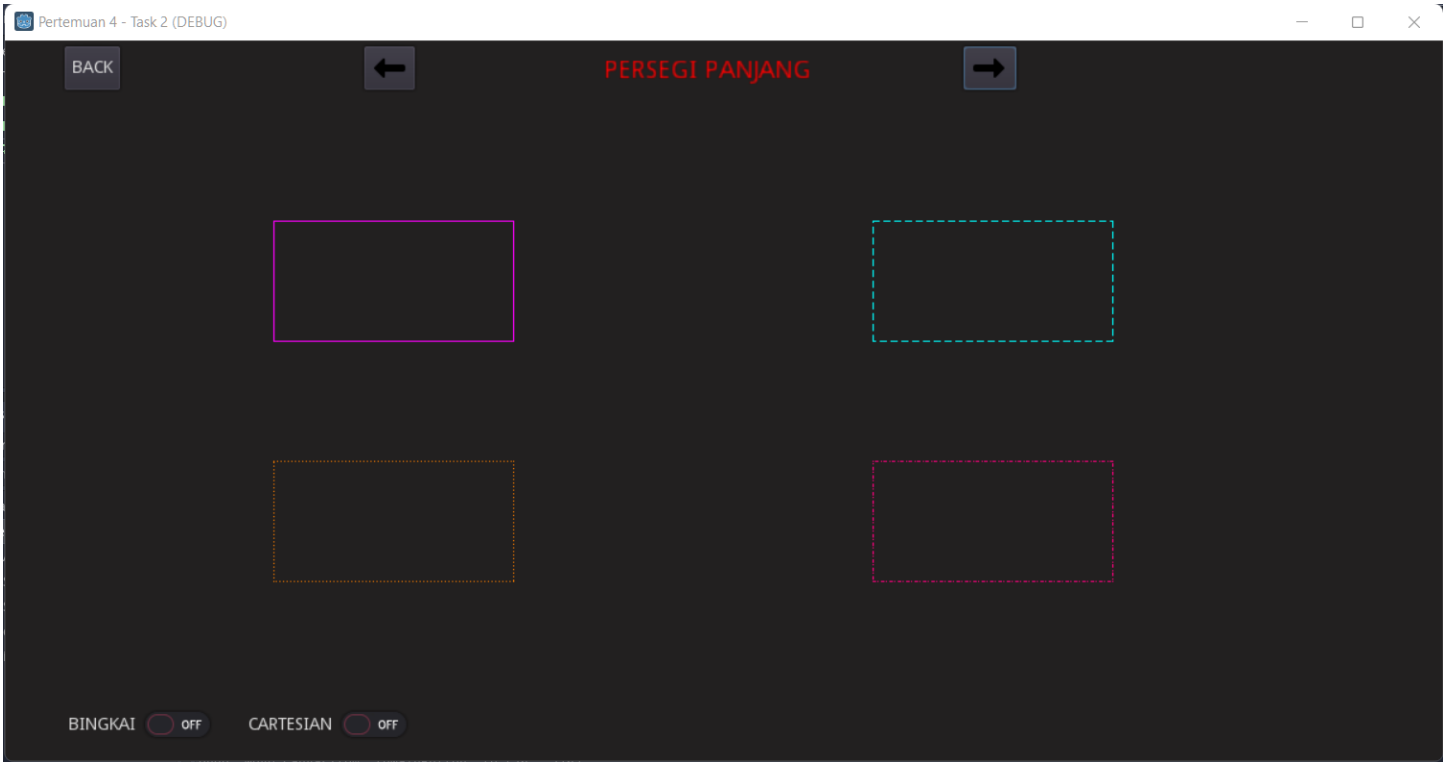
1. **Pemanfaatan Godot Node2D**
    - Kelas karya1 mewarisi Node2D, yang memungkinkan untuk menggambar bentuk langsung di dalam metode _Draw().
    - Penggunaan QueueRedraw() memastikan bahwa tampilan diperbarui secara berkala.
2. **Manajemen Titik Pusat (World Origin)**
    - World Origin dihitung secara dinamis berdasarkan ukuran viewport menggunakan GetViewportRect().Size.
    - Hal ini memastikan fleksibilitas dalam berbagai ukuran layar dan resolusi.

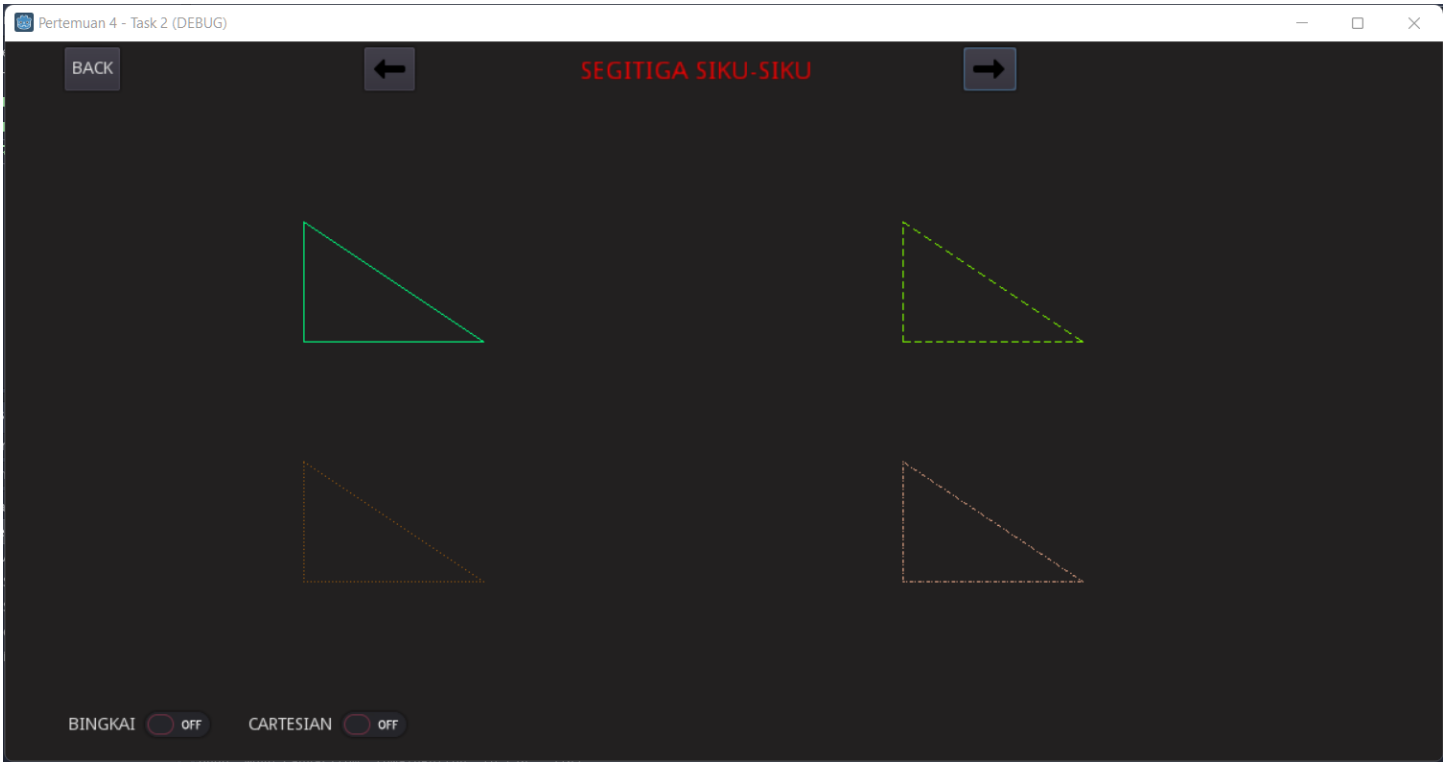
TASK 2: MODIFIKASI GARIS

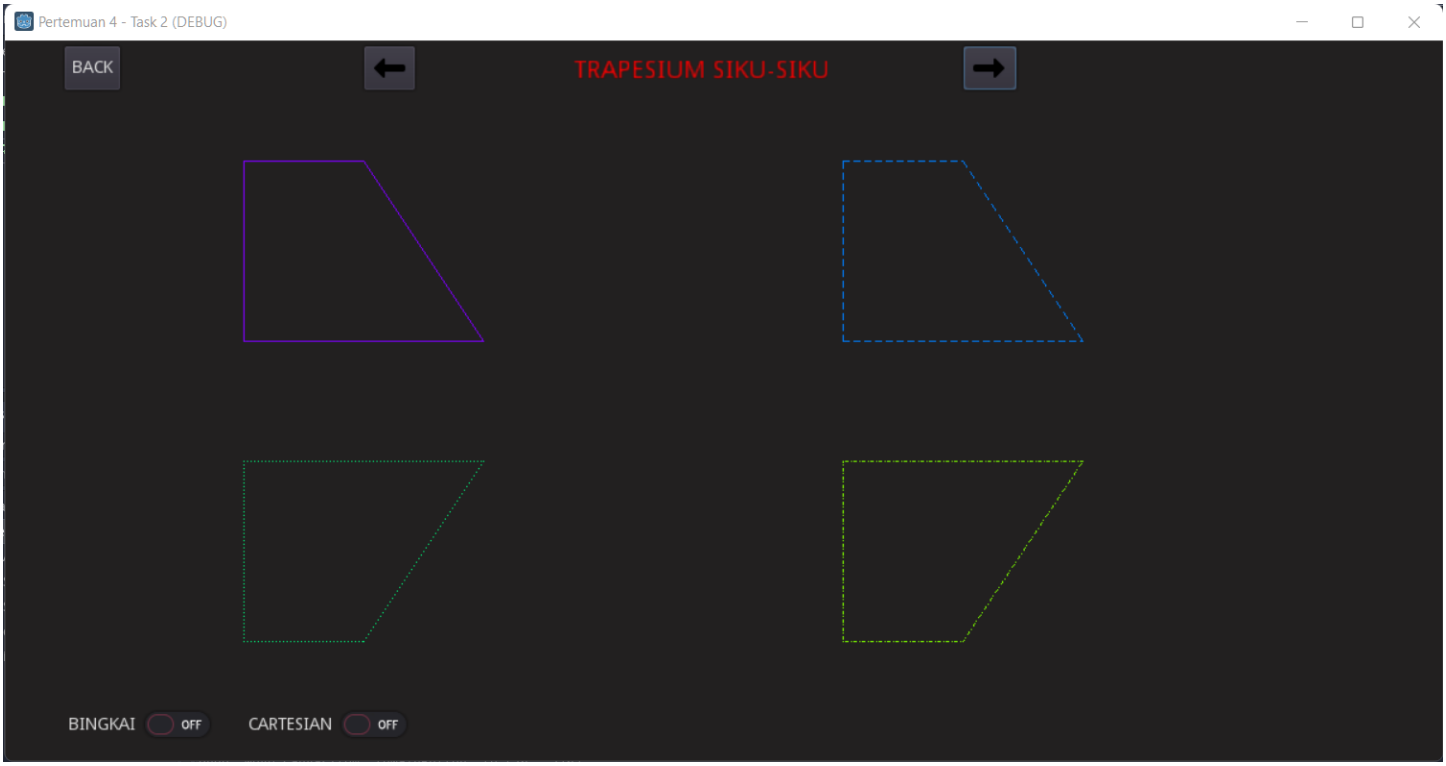
1. Buatlah berbagai macam attribute garis seperit titik-titik, titik-garis-titik, garis-garis dan lain-lain lalu buatlah scene karya2 seperti berikut

2. Buatlah Scene baru, karya3 yang merupakan Modifikasi Bentuk Dasar Persegi, Persegi Panjang, Segitiga Siku-Siku, dan Trapesium Siku-Siku, Lingkaran dan Ellips yang attribute garisnya sesuai tipe yang diiginkan, normal, titik-titik, titik-garis-titik, garis-garis.
3. Contoh Scene sbb:




Pertemuan 4 - Task 2 (DEBUG)
BACK
PERSEGI PANJANG
BINGKAI
OFF
CARTESIAN
OFF


Pertemuan 4 - Task 2 (DEBUG)
BACK
SEGITIGA SIKU-SIKU
BINGKAI
OFF
CARTESIAN
OFF



Lesson Learnt (Code, Print Screen Hasil Karya, dan Komentar)

Karya 2

```
namespace Godot;

using Godot;
using System;

public partial class karya2 : Node2D
{

	private bentukdasar _bentukDasar = new bentukdasar();

	private const int MarginLeft = 50;
	private const int MarginTop = 50;

	public override void _Ready()
	{

	}

	public override void _Draw()
	{
		Vector2 WindowSize = GetViewportRect().Size;
		int ScreenWidth = (int)WindowSize[0];
		int ScreenHeight = (int)WindowSize[1];
		int MarginRight = ScreenWidth - MarginLeft;
		int MarginBottom = ScreenHeight - MarginTop;

		MarginPixel(MarginLeft, MarginTop, MarginRight, MarginBottom);
```

```
            MyGarisVariasi();
        }


        private void MarginPixel(int MarginLeft, int MarginTop, int MarginRight, int MarginBottom){
            Godot.Color color = new Godot.Color("#32CD30");
            var margin = _bentukDasar.Margin(MarginLeft, MarginTop, MarginRight, MarginBottom);
            PutPixelAll(margin, color);
        }

        private void PutPixel(float x, float y, Godot.Color? color = null)
        {
            // Provide a default color if 'color' is null
            Godot.Color actualColor = color ?? Godot.Colors.White;
            Godot.Vector2[] points = new Godot.Vector2[]{new Godot.Vector2(Mathf.Round(x),
Mathf.Round(y))};
            Godot.Vector2[] uvs = new Godot.Vector2[]
            {
                Vector2.Zero, Vector2.Down, Vector2.One, Vector2.Right
            };

            DrawPrimitive(points, new Godot.Color[]{ actualColor }, uvs);
        }

        private void MyGarisVariasi()
        {
            Godot.Color color1 = new Godot.Color("#FF2D00"); // Merah
            Godot.Color color2 = new Godot.Color("#FFBD00"); // Biru
            Godot.Color color3 = new Godot.Color("#00FF00"); // Hijau

            var garis1 = _bentukDasar.GarisPola(new Vector2(100, 200), new Vector2(300, 200), "dotted");
            var garis2 = _bentukDasar.GarisPola(new Vector2(100, 250), new Vector2(300, 250), "dashed");
            var garis3 = _bentukDasar.GarisPola(new Vector2(100, 300), new Vector2(300, 300), "dash-dot");

            var gariskanan1 = _bentukDasar.GarisPola(new Vector2(200, 100), new Vector2(200, 300), "dotted");
            var gariskanan2 = _bentukDasar.GarisPola(new Vector2(250, 100), new Vector2(250, 300), "dashed");
            var gariskanan3 = _bentukDasar.GarisPola(new Vector2(300, 100), new Vector2(300, 300), "dash-
dot");
            PutPixelAll(garis1, color1);
            PutPixelAll(garis2, color2);
            PutPixelAll(garis3, color3);
            PutPixelAll(gariskanan1, color1);
            PutPixelAll(gariskanan2, color2);
            PutPixelAll(gariskanan3, color3);
        }

        private void PutPixelAll(System.Collections.Generic.List<Vector2> dot, Godot.Color? color = null)
        {
            foreach (Vector2 point in dot)
            {
                PutPixel(point[0], point[1], color);
            }
        }

        public override void _ExitTree()
```

```
        {
                GD.Print($"_bentukDasar is null in _ExitTree(): {_bentukDasar == null}");
                _bentukDasar?.Dispose(); // Pastikan _bentukDasar tidak null sebelum Dispose
                _bentukDasar = null;
                GD.Print($"_bentukDasar is null in _ExitTree(): {_bentukDasar == null}");
                base._ExitTree();
        }

}
```

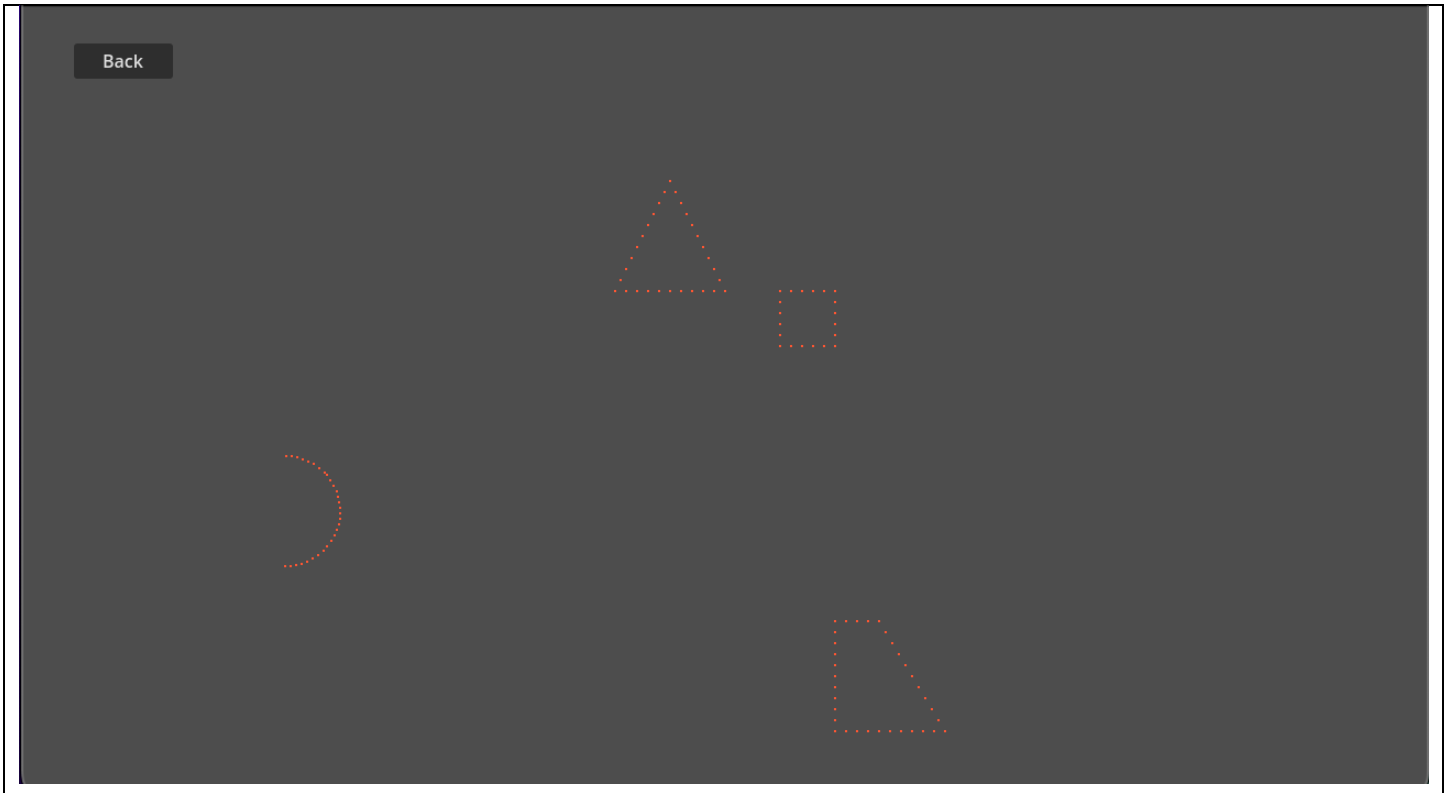
Disini, ada beberapa Garis yang saya buat tegak lurus, sesuai tugas saya buat garis beberapa titik titik dan beberapa garis biasa.

Karya 3

```
namespace Godot;


using Godot;

using System;

using System.Collections.Generic;


public partial class karya3 : Node2D

{
```

```
private primitif _primitif = new primitif();
private bentukdasar _bentukdasar;
private const int MarginLeft = 50;
private const int MarginTop = 50;

private int WorldOriginX;
private int WorldOriginY;

public override void _Ready()
{
	GD.Print("karya3 _Ready() dipanggil");
	_bentukdasar = new bentukdasar();

	if (_bentukdasar == null)
		GD.PrintErr("ERROR: _bentukdasar masih null!");

	WorldOriginX = (int)(GetViewportRect().Size.X / 2);
	WorldOriginY = (int)(GetViewportRect().Size.Y / 2);

	QueueRedraw();
}

public override void _Draw()
{
	DrawShapes();
}

private void DrawShapes()
{
	Godot.Color colorShape = new Godot.Color("#FF5733");

	List<Vector2> persegi1 = _bentukdasar.Persegi(WorldOriginX + 50, WorldOriginY - 100, 50);
```

```
            List<Vector2> segitiga = _bentukdasar.SegitigaSamaKaki(WorldOriginX - 100, WorldOriginY - 100,
100, 100);
            List<Vector2> lingkaran = _bentukdasar.Lingkaran(WorldOriginX - 400, WorldOriginY + 100, 50);
            List<Vector2> trapesium = _bentukdasar.TrapesiumSikuSiku(WorldOriginX + 100, WorldOriginY +
300, 100, 40, 100);


            PutPixelDotted(persegi1, colorShape);
            PutPixelDotted(segitiga, colorShape);
            PutPixelDotted(lingkaran, colorShape);
            PutPixelDotted(trapesium, colorShape);
        }


        private void PutPixel(float x, float y, Godot.Color? color = null)
        {
            if (x < 0 || y < 0 || x > GetViewportRect().Size.X || y > GetViewportRect().Size.Y)
            {
                GD.PrintErr($"Warning: Titik di luar layar ({x}, {y})");
                return;
            }
            Godot.Color actualColor = color ?? Godot.Colors.White;
            DrawCircle(new Vector2(x, y), 1, actualColor);
        }


        private void PutPixelDotted(List<Vector2> dots, Godot.Color? color = null)
        {
            for (int i = 0; i < dots.Count; i += 10) // Hanya menggambar titik-titik dengan selang 2 pixel
            {
                PutPixel(dots[i].X, dots[i].Y, color);
            }
        }
}
```



Ini adalah karya 3 dimana saya meletakkan shape secara sembarang dan membuat shapenya titiktitik, sayabelajar dari task 2 ini adalah

Optimasi Gambar dengan Titik (PutPixel Method)

- Menggunakan metode PutPixel() untuk menggambar titik individu dengan warna yang dapat dikustomisasi.
- Metode PutPixelAll() memungkinkan menggambar banyak titik secara efisien dalam satu panggilan.
- Pada Karya 3, metode PutPixelDotted() digunakan untuk menggambar titik-titik dengan selang tertentu guna menghemat rendering dan meningkatkan performa.
- Validasi batas layar diterapkan untuk menghindari rendering di luar viewport dengan GD.PrintErr() jika koordinat keluar dari batas layar.

Implementasi Algoritma Grafik

- Pada Karya 2, garis dibuat menggunakan metode _bentukDasar.GarisPola() dengan berbagai variasi pola garis.
- Pada Karya 3, berbagai bentuk geometri seperti persegi, segitiga, lingkaran, dan trapesium dibuat menggunakan metode _bentukdasar.Persegi(), _bentukdasar.SegitigaSamaKaki(), _bentukdasar.Lingkaran(), dan _bentukdasar.TrapesiumSikuSiku().

Debugging dan Penanganan Error

- Log error ditampilkan menggunakan GD.PrintErr() untuk mendeteksi objek yang tidak terinisialisasi.
- Pada Karya 3, peringatan diberikan ketika ada titik yang digambar di luar batas layar.
- Pada Karya 2, _bentukDasar dipastikan tidak null sebelum Dispose() dipanggil dalam _ExitTree().

Fleksibilitas dalam Desain Grafis

- Bentuk dapat dibuat pada berbagai kuadran dengan koordinat yang dihitung secara dinamis.
- Pada Karya 3, sistem koordinat dunia digunakan untuk menyesuaikan letak bentuk relatif terhadap pusat layar.
- Warna dapat dikustomisasi dengan mudah untuk setiap bentuk yang digamba

PENGUMPULAN

Ikuti Format yang diberikan di Google Classroom.