PERTEMUAN 4 GODOT

LAPORAN PRAKTIKUM

Disusun untuk memenuhi tugas Mata Kuliah Komputer Grafik

Disusun oleh

Nazwa Fitriyani Zahra 211511051

PROGRAM STUDI D3 TEKNIK INFORMATIKA

JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA

POLITEKNIK NEGERI BANDUNG

2022

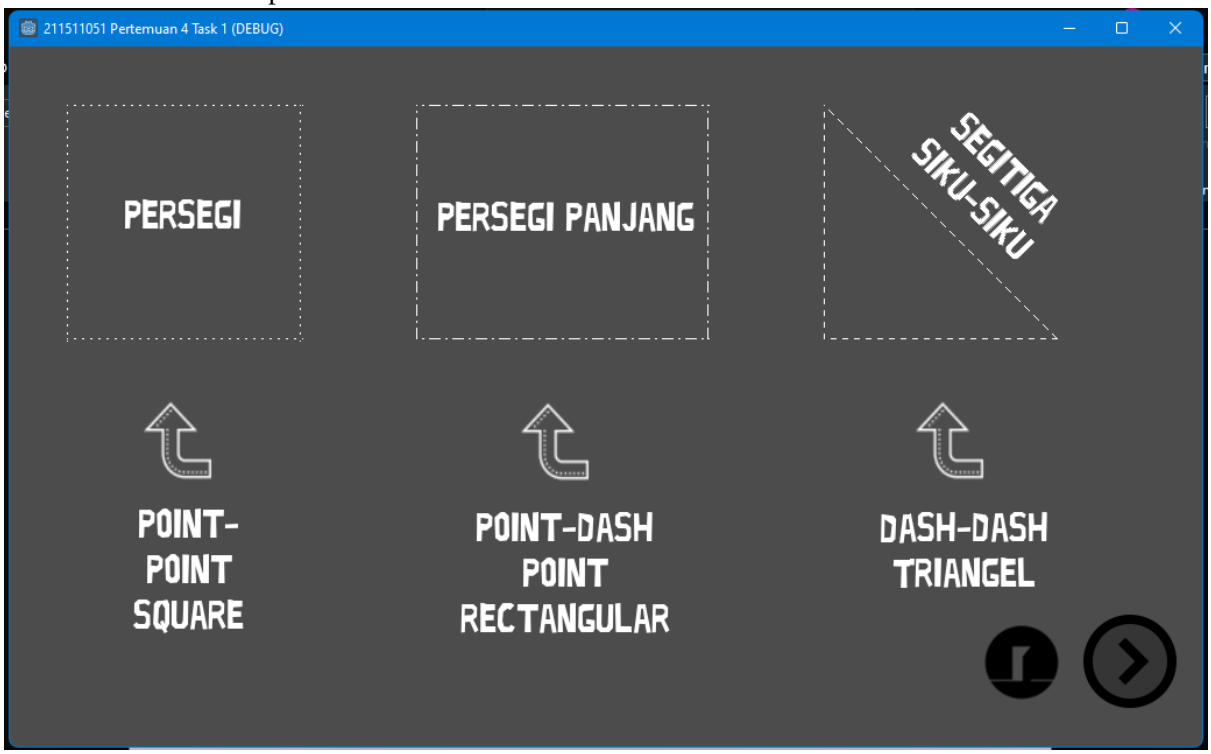
DAFTAR ISI

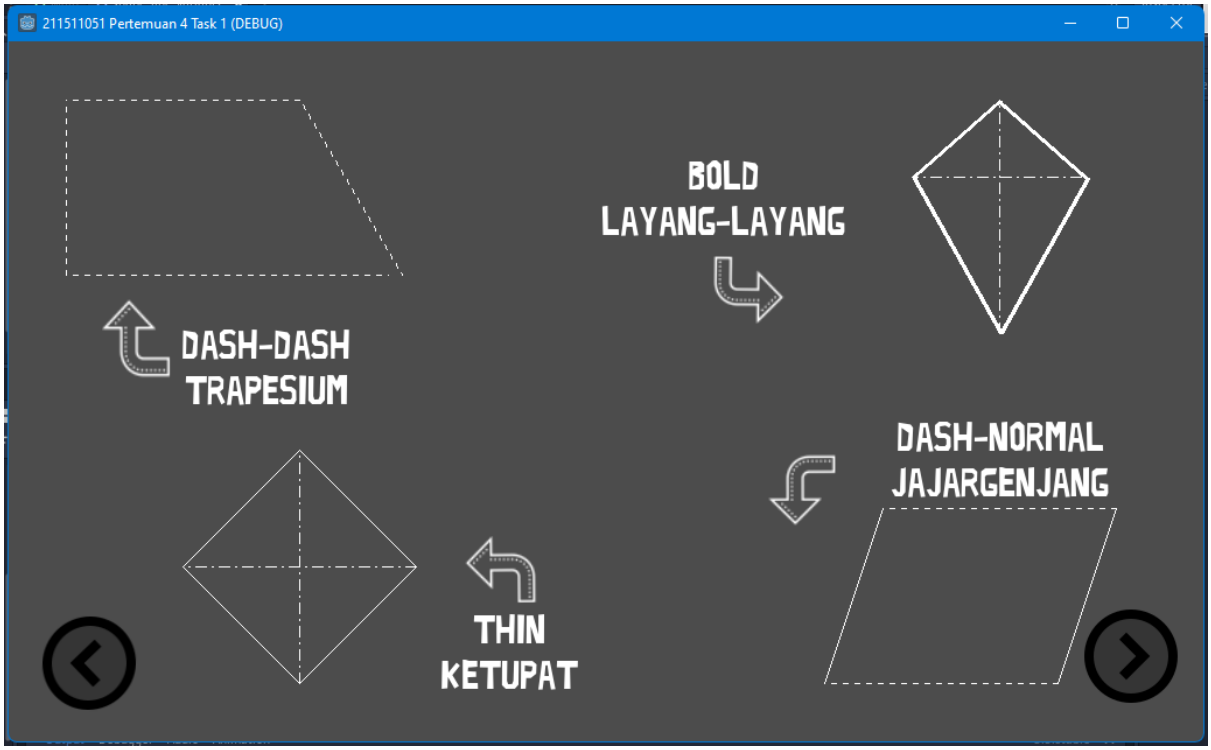
DAFTAR REFERENSI

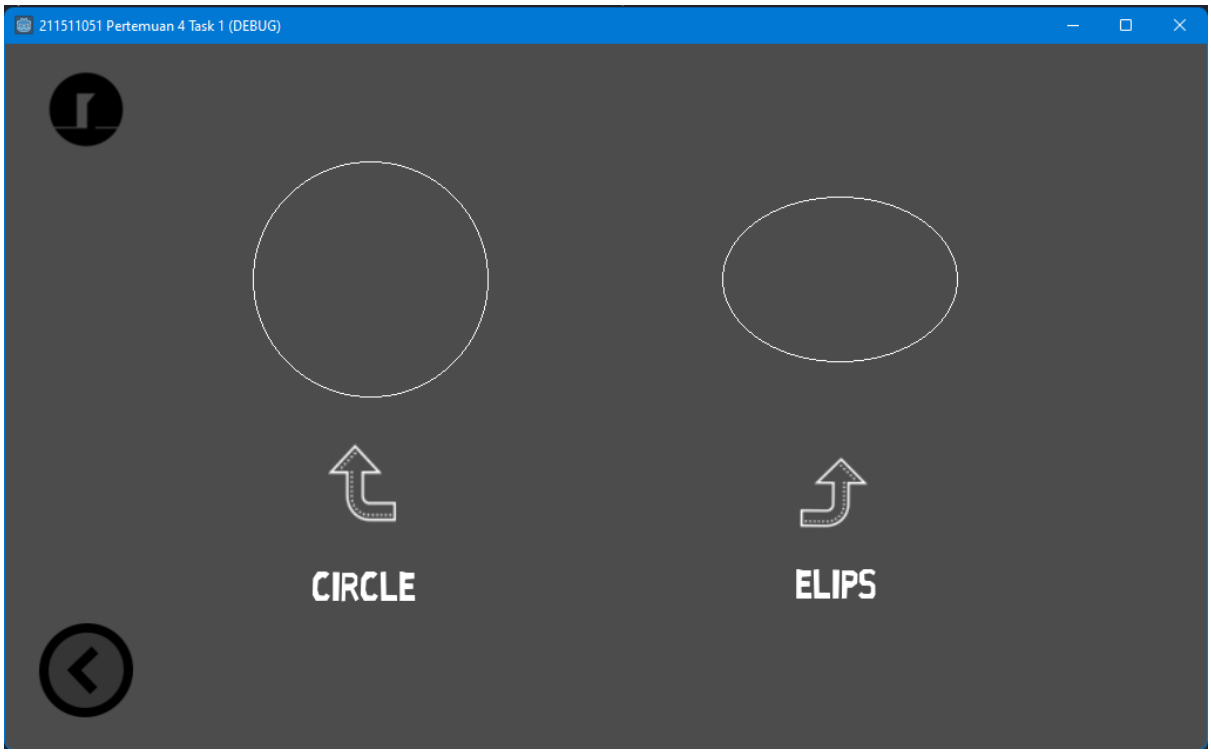
- https://docs.godotengine.org/en/3.5/index.html
- https://github.com/godotengine/godot-demo-projects
- https://docs.godotengine.org/en/3.5/community/tutorials.html
- https://www.youtube.com/watch?v=ZUPBoqC_X_o
- YouTube Channel : Kelas Terbuka – Godot Tutorial
- https://docs.godotengine.org/en/stable/tutorials/2d/custom_drawing_in_2d.html
- https://docs.godotengine.org/en/stable/tutorials/2d/custom_drawing_in_2d.html

A. Task 1

Berikut ini hasil Manipulasi Garis dan Bentuk Dasar







Berikut Source Code dari hasil diatas

//line.gd

Setiap tipe line dibuat function baru seperti dibawah ini :

```
extends Node2D


# Declare member variables here. Examples:
# var a = 2
# var b = "text"


# Called when the node enters the scene tree for the first time.
func _ready():
    pass # Replace with function body.
var color = Color(1,1,1,1)

# Called every frame. 'delta' is the elapsed time since the previous frame.
#func _process(delta):
#    pass
func put_pixel(x, y, color):
    draw_primitive(PoolVector2Array([Vector2(x, y)]),
        PoolColorArray([color]),
        PoolVector2Array())

func put_pixel_all(dot: PoolVector2Array, color):
    for i in dot.size():
        put_pixel(dot[i].x, dot[i].y, color)


```

```
#line_dda
func lineDDA(xa : float, ya : float, xb : float, yb : float):
    var dx = xb - xa
    var dy = yb - ya
    var steps
    var xIncrement
    var yIncrement
    var x = xa
    var y = ya
    var res = PoolVector2Array()

    if (abs(dx) > abs(dy)) :
        steps = abs(dx)
    else :
        steps = abs(dy)

    xIncrement = dx/ float(steps)
    yIncrement = dy/ float(steps)
    res.append(Vector2(round(x), round(y)))

    for k in steps:
        x += xIncrement
        y += yIncrement
        res.append(Vector2(round(x), round(y)))
    #Dipanggil agar garis tampil
    put_pixel_all(res, color)
    return res

func lineBres(xa, ya, xb, yb):
    var dx = abs(xa - xb)
    var dy = abs(ya - yb)
    var p = 2 * dy - dx
    var twoDy = 2 * dy
    var twoDyDx = 2 * (dy - dx)
    var x
    var y
    var xEnd
    var k = 1
    var res = PoolVector2Array()

    if xa > xb :
        x = xb
        y = yb
        xEnd = xa
    else :
        x = xa
        y = ya
        xEnd = xb

    res.append(Vector2(round(x), round(y)))

    while x < xEnd :
        x += 1
        if p < 0 :
            p += twoDy
        else :
            y += 1
            p += twoDyDx
        res.append(Vector2(round(x), round(y)))
        k += 1
    #Dipanggil agar garis tampil
    put_pixel_all(res, color)
    return res
```

```gdscript
func dash_dash(xa : float, ya : float, xb : float, yb : float):
    var dx = xb - xa
    var dy = yb - ya
    var steps
    var xIncrement
    var yIncrement
    var x = xa
    var y = ya
    var res = PoolVector2Array()


    if (abs(dx) > abs(dy)) :
        steps = abs(dx)
    else :
        steps = abs(dy)

    xIncrement = dx/ float(steps)
    yIncrement = dy/ float(steps)
    res.append(Vector2(round(x), round(y)))
```

```gdscript
    if(ya == yb):
        for k in steps :
            if(k % 5 == 0):
                x += 5
            x += xIncrement
            y += yIncrement
            res.append(Vector2(round(x), round(y)))
            if(x >= xb):
                break
            print(k)
    elif(xa == xb) :
        for k in steps :
            if(k % 5 == 0):
                y += 5
            y += yIncrement
            res.append(Vector2(round(x), round(y)))
            if(x >= xb && y >= yb):
                break
            print(k)
    else:
        for k in steps:
            if(k % 5 == 0):
                x += 3
                y += 3
            x += xIncrement
            y += yIncrement
            put_pixel(round(x), round(y), color)
            if(x >= xb && y >= yb):
                break
            print(k)
    #Dipanggil agar garis tampil
    put_pixel_all(res, color)
```

```gdscript
func point_dash(xa : float, ya : float, xb : float, yb : float):
    var dx = xb - xa
    var dy = yb - ya
    var steps
    var xIncrement
    var yIncrement
    var x = xa
    var y = ya
    var res = PoolVector2Array()
    var tampung = 0

    if (abs(dx) > abs(dy)) :
        steps = abs(dx)
    else :
        steps = abs(dy)

    xIncrement = dx/ float(steps)
    yIncrement = dy/ float(steps)
    res.append(Vector2(round(x), round(y)))
```

```
	if(ya == yb):
		for k in steps:
			if(int(x) % 10 == 0):
				x+= 4
				tampung = x
			if(x == tampung + 2):
				x+= 4
			x += xIncrement
			y += yIncrement
			res.append(Vector2(round(x), round(y)))
			if(x >= xb && y >= yb):
				break
	else :
		for k in steps:
			if(int(y) % 10 == 0):
				y+= 4
				tampung = y
			if(y == tampung + 2):
				y+= 4
			y += yIncrement
			res.append(Vector2(round(x), round(y)))
			if(x >= xb && y >= yb):
				break
	#Dipanggil agar garis tampil
	put_pixel_all(res, color)
	return res
```

```
func point_point(xa : float, ya : float, xb : float, yb : float):
	var dx = xb - xa
	var dy = yb - ya
	var steps
	var xIncrement
	var yIncrement
	var x = xa
	var y = ya
	var res = PoolVector2Array()

	if (abs(dx) > abs(dy)) :
		steps = abs(dx)
	else :
		steps = abs(dy)

	xIncrement = dx/ float(steps)
	yIncrement = dy/ float(steps)
	res.append(Vector2(round(x), round(y)))
```

```
	if(ya == yb):
		for k in steps :
			if(k % 2 == 0):
				x += 5
			x += xIncrement
			y += yIncrement
			res.append(Vector2(round(x), round(y)))
			if(x >= xb && y >= yb):
				break
			print(k)
	elif(xa == xb) :
		for k in steps :
			if(k % 2 == 0):
				y += 5
			y += yIncrement
			res.append(Vector2(round(x), round(y)))
			if(x >= xb && y >= yb):
				break
			print(k)

	#Dipanggil agar garis tampil
	put_pixel_all(res, color)
	return res
```

```
func bold(xa : float, ya : float, xb : float, yb : float):
    var dx = xb - xa
    var dy = yb - ya
    var steps
    var xIncrement
    var yIncrement
    var x = xa
    var y = ya
    var res = PoolVector2Array()

    if (abs(dx) > abs(dy)) :
        steps = abs(dx)
    else :
        steps = abs(dy)

    xIncrement = dx/ float(steps)
    yIncrement = dy/ float(steps)
    res.append(Vector2(round(x), round(y)))
    for i in steps:
        x += xIncrement #11
        y += yIncrement #10
        res.append(Vector2(round(x), round(y)))
        for n in 4:
            if dx > dy:
                res.append(Vector2(round(x), round(y+n)))
            else:
                res.append(Vector2(round(x+n), round(y)))

    #Dipanggil agar garis tampil
    put_pixel_all(res, color)
    return res
```

//shape.gd

Dari line.gd yang telah dibuat, dikembangkan untuk digunakan dalam membuat bidang. Bidang dibuat dengan mengextend fungsi yang ada di line.gd. Berikut adalah source code pembuatan bidang 2d

```
func trapesium_siku_siku(titik_awal: Vector2, atas, tinggi, bawah):
    var res = PoolVector2Array()
    dash_dash(titik_awal.x, titik_awal.y, titik_awal.x + atas, titik_awal.y) #sisi atas
    dash_dash(titik_awal.x, titik_awal.y + tinggi, titik_awal.x + bawah + round(abs(bawah-atas)/2), titik_awal.y + tinggi) #sisi bawah
    dash_dash(titik_awal.x, titik_awal.y, titik_awal.x, titik_awal.y + tinggi) #sisi kiri
    dash_dash(titik_awal.x + atas, titik_awal.y, titik_awal.x + bawah, titik_awal.y + tinggi) #sisi kanan
    return res

func jajargenjang(titik_awal: Vector2, panjang, tinggi, geser):
    var res = PoolVector2Array()
    dash_dash(titik_awal.x, titik_awal.y, titik_awal.x + panjang, titik_awal.y) #sisi atas
    dash_dash((titik_awal.x - geser), titik_awal.y + tinggi, (titik_awal.x + panjang - geser), titik_awal.y + tinggi) #sisi bawah
    res.append_array(lineDDA(titik_awal.x, titik_awal.y, (titik_awal.x - geser), titik_awal.y + tinggi)) #sisi kiri
    res.append_array(lineDDA(titik_awal.x + panjang, titik_awal.y, ((titik_awal.x + panjang) - geser), titik_awal.y + tinggi)) #sisi k
    return res

func ketupat (titik_awal: Vector2, diagonal):
    var res = PoolVector2Array()
    point_dash(titik_awal.x, titik_awal.y, titik_awal.x, titik_awal.y + diagonal)
    point_dash(titik_awal.x - (diagonal/2), titik_awal.y + (diagonal/2), titik_awal.x + (diagonal/2), titik_awal.y + (diagonal/2))
    res.append_array(lineDDA(titik_awal.x, titik_awal.y, titik_awal.x  - (diagonal/2), titik_awal.y + (diagonal/2)))
    res.append_array(lineDDA(titik_awal.x, titik_awal.y, titik_awal.x  + (diagonal/2), titik_awal.y + (diagonal/2)))
    res.append_array(lineDDA(titik_awal.x - (diagonal/2), titik_awal.y + (diagonal/2), titik_awal.x, titik_awal.y + diagonal))
    res.append_array(lineDDA(titik_awal.x + (diagonal/2), titik_awal.y + (diagonal/2), titik_awal.x, titik_awal.y + diagonal))
    return res

```

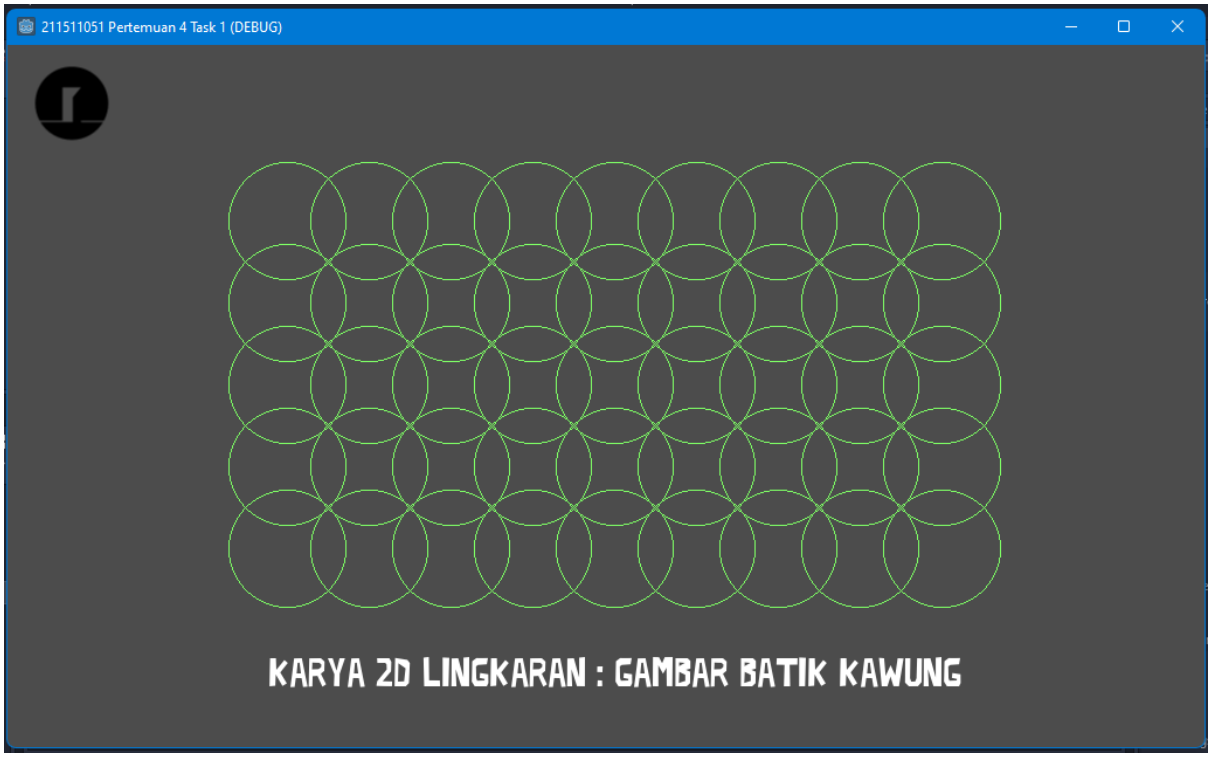
```
54 func layanglayang (titik_awal: Vector2, diagonal_a, diagonal_b):
55     var res = PoolVector2Array()
56     point_dash(titik_awal.x, titik_awal.y, titik_awal.x, titik_awal.y + diagonal_a)
57     point_dash(titik_awal.x - (diagonal_b/2), titik_awal.y + (diagonal_a/3), titik_awal.x + (diagonal_b/2), titik_awal.y + (diagonal_a/3))
58     res.append_array(bold(titik_awal.x, titik_awal.y, titik_awal.x  - (diagonal_b/2), titik_awal.y + (diagonal_a/3)))
59     res.append_array(bold(titik_awal.x, titik_awal.y, titik_awal.x  + (diagonal_b/2), titik_awal.y + (diagonal_a/3)))
60     res.append_array(bold(titik_awal.x - (diagonal_b/2), titik_awal.y + (diagonal_a/3), titik_awal.x, titik_awal.y + diagonal_a))
61     res.append_array(bold(titik_awal.x + (diagonal_b/2), titik_awal.y + (diagonal_a/3), titik_awal.x, titik_awal.y + diagonal_a))
62     return res
63
64 func circleMidPoint(xCenter, yCenter, radius, color):
65     var x = 0
66     var y = radius
67     var p = 1 - radius
68
69     circlePlotPoints(xCenter, yCenter, x, y, color)
70
71     while (x < y):
72         x += 1
73         if (p < 0):
74             p += 2*x + 1
75         else:
76             y -= 1
77             p += 2*(x-y) + 1
78         circlePlotPoints(xCenter, yCenter, x, y, color)
```

```
80 func circlePlotPoints(xCenter, yCenter, x, y, color):
81     put_pixel(xCenter + x, yCenter + y, color) #1
82     put_pixel(xCenter - x, yCenter + y, color) #2
83     put_pixel(xCenter + x, yCenter - y, color) #3
84     put_pixel(xCenter - x, yCenter - y, color) #4
85     put_pixel(xCenter + y, yCenter + x, color) #5
86     put_pixel(xCenter - y, yCenter + x, color) #6
87     put_pixel(xCenter + y, yCenter - x, color) #7
88     put_pixel(xCenter - y, yCenter - x, color) #8
89
90 func ellipseMidPoint(xCenter, yCenter, Rx, Ry, color):
91     var Rx2 = Rx*Rx
92     var Ry2 = Ry*Ry
93     var twoRx2 = 2*Rx2
94     var twoRy2 = 2*Ry2
95     var p
96     var x = 0
97     var y = Ry
98     var px = 0
99     var py = twoRx2*y
100
101
102     ellipsePlotPoints(xCenter, yCenter, x, y, color)
```

```
    p = round(Ry2 - (Rx2*Ry) + (0.25 * Rx2))
    while (px < py):
        x += 1
        px += twoRy2
        if (p < 0):
            p += Ry2 + px
        else:
            y -= 1
            py -= twoRx2
            p += Ry2 + px - py
        ellipsePlotPoints(xCenter, yCenter, x, y, color)

    # Region 2 #
    p = round(Ry2*(x+0.5)*(x+0.5) + Rx2*(y-1)*(y-1) - Rx2*Ry2)
    while (y > 0):
        y -= 1
        py -= twoRx2
        if (p > 0):
            p += Rx2 - py
        else :
            x += 1
            px += twoRy2
            p += Rx2 - py + px
        ellipsePlotPoints(xCenter, yCenter, x, y, color)

func ellipsePlotPoints(xCenter, yCenter, x, y, color):
    put_pixel(xCenter + x, yCenter + y, color)
    put_pixel(xCenter - x, yCenter + y, color)
    put_pixel(xCenter + x, yCenter - y, color)
    put_pixel(xCenter - x, yCenter - y, color)

```

B. Task 2

Berikut adalah hasil output dari pertanyaan Task 2



Karya 2D Lingkaran yang saya buat adalah berbentuk batik kawung

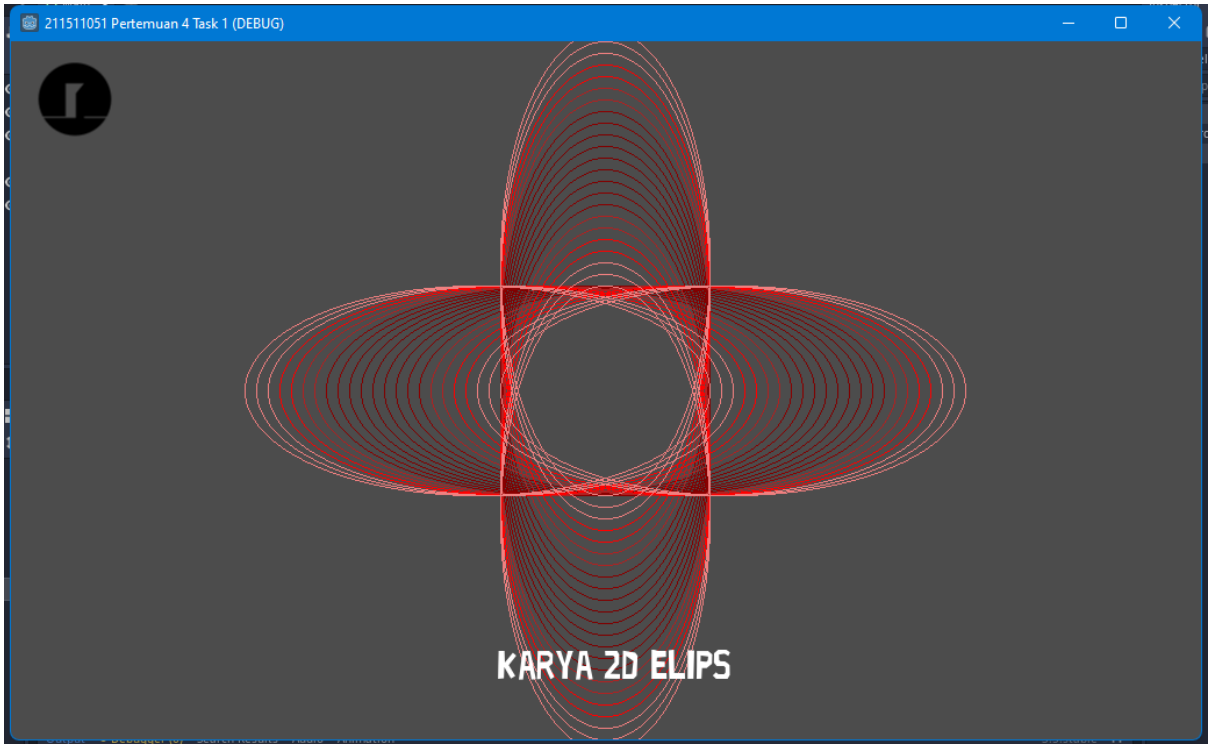
Dari library line dan shape yang telah dibuat sebelumnya, saya membuat function baru dengan nama lingkaranoverlapping yang mengextend function dari shape.gd. Berikut source codenya :

//call_overlapping.gd

```
extends 'res://SCRIPT/shape.gd'
func _ready():
    pass # Replace with function body.


func lingkaran_overlapping():
    var x = 170
    var y = 150
    var i = 0
    while (i<5):
        lingkaran_horizontal(x, y)
        y = y + 70
        i = i + 1
    #lingkaran_horizontal(x, y+70)
    #lingkaran_horizontal(x, y+140)
    #lingkaran_horizontal(x, y+210)
    #lingkaran_horizontal(x, y+280)
func lingkaran_horizontal(x, y):
    var i = 0

    while (i < 9) :
        x = x + 70
        circleMidPoint(x, y, 50, Color(0.5,1,0.4,1))
        i = i + 1


func _draw():
    lingkaran_overlapping()

```

C. Task 3

Berikut adalah hasil output dari pertanyaan Task 3



Karya 2D Elips yang saya buat adalah berupa bunga kelopak 4

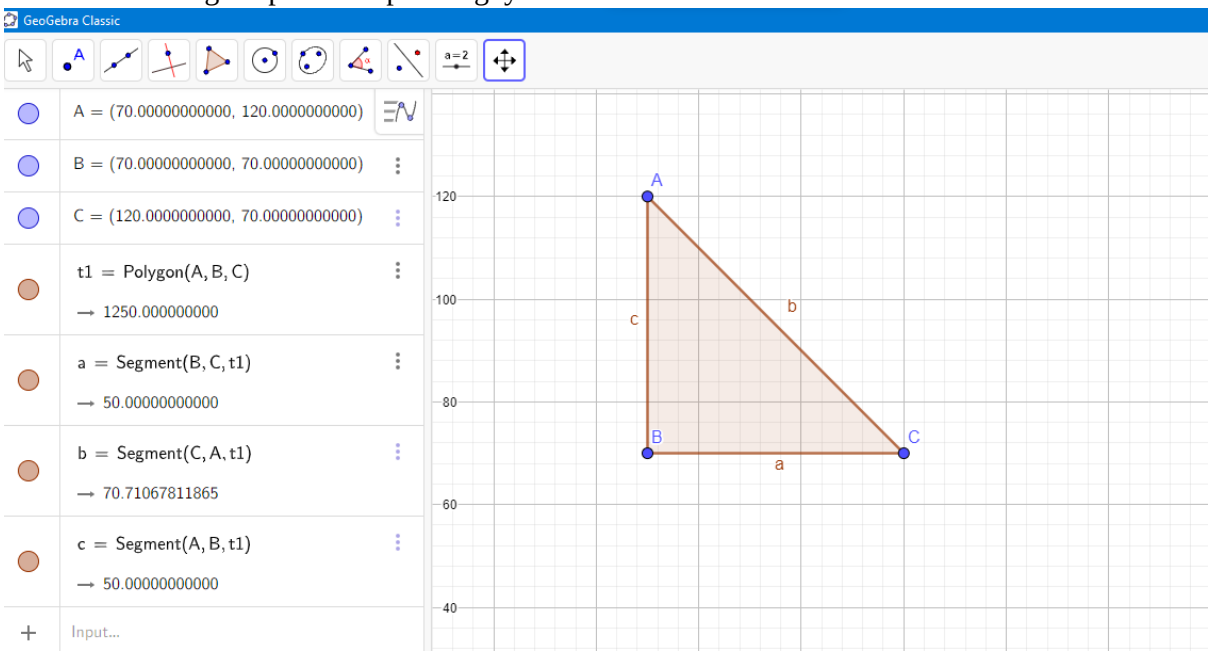
Dari library shape yang telah dibuat sebelumnya, saya membuat function baru dengan nama Elips2D yang mana isinya untuk membentuk karya elips seperti diatas ini. Berikut source codenya :

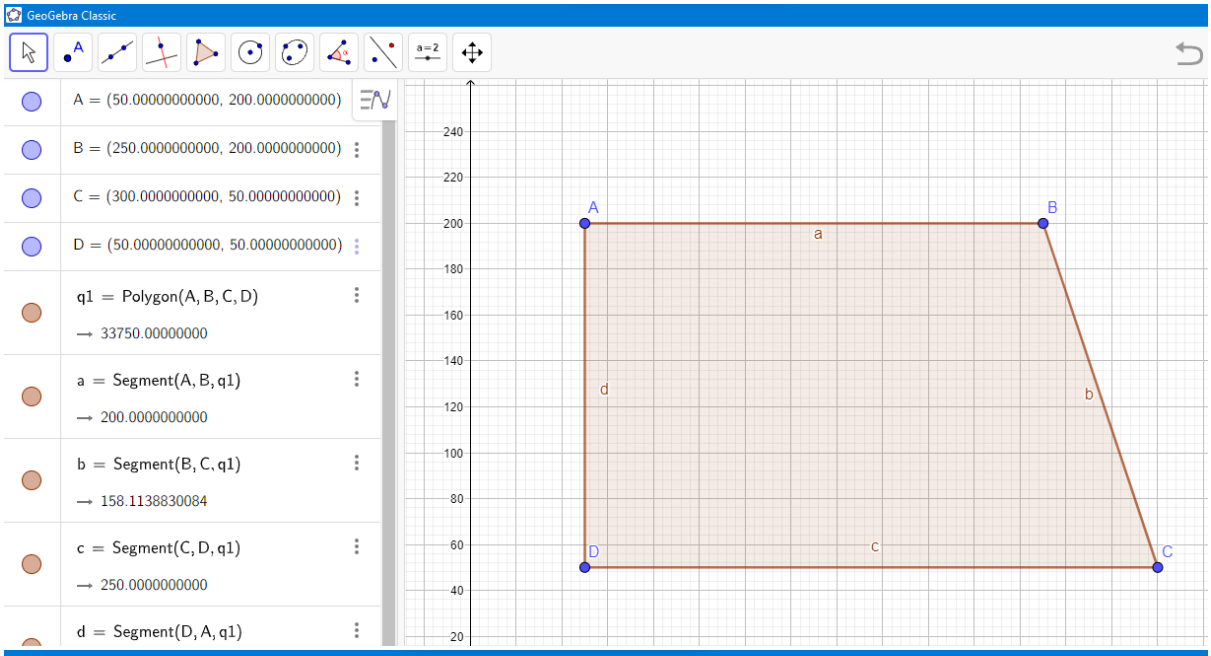
//call_elips

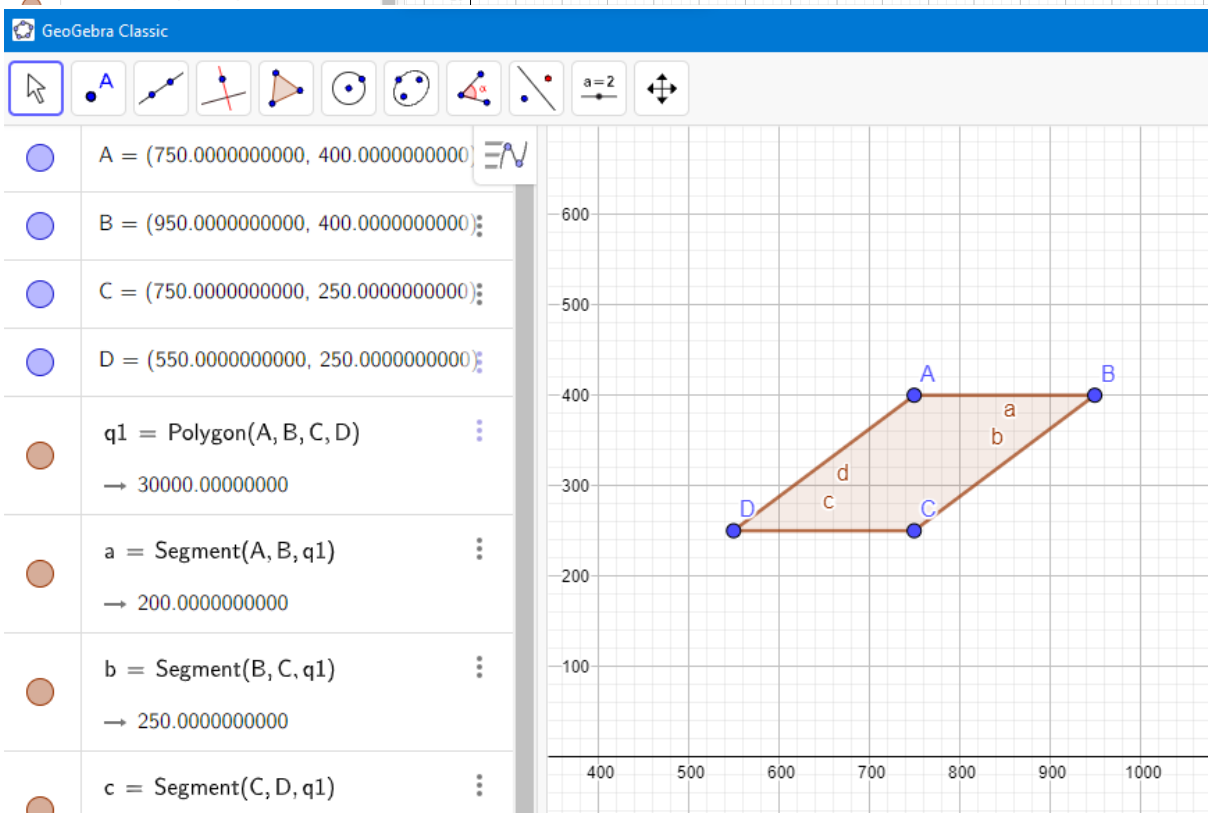
```
func elips2D():
    var i = 0
    var l = 10
    var color = Color(0.501961, 0, 0, 1)
    ellipseMidPoint(get_viewport().size.x/2, get_viewport().size.y/2, 200, 90, color)
    ellipseMidPoint(get_viewport().size.x/2, get_viewport().size.y/2, 90, 200, color)
    while (i <= 10):
        if (i == 2):
            color = Color( 0.545098, 0, 0, 1 )
        elif(i == 4):
            color = Color( 0.698039, 0.133333, 0.133333, 1 )
        elif(i == 6):
            color = Color( 1, 0, 0, 1 )
        elif(i == 8):
            color = Color( 0.941176, 0.501961, 0.501961, 1 )
        ellipseMidPoint(get_viewport().size.x/2+l, get_viewport().size.y/2, 200, 90, color)
        ellipseMidPoint(get_viewport().size.x/2-l, get_viewport().size.y/2, 200, 90, color)
        ellipseMidPoint(get_viewport().size.x/2, get_viewport().size.y/2-l, 90, 200, color)
        ellipseMidPoint(get_viewport().size.x/2, get_viewport().size.y/2+l, 90, 200, color)
        l = l + 10
        i = i + 1

func _draw():
    elips2D()
```

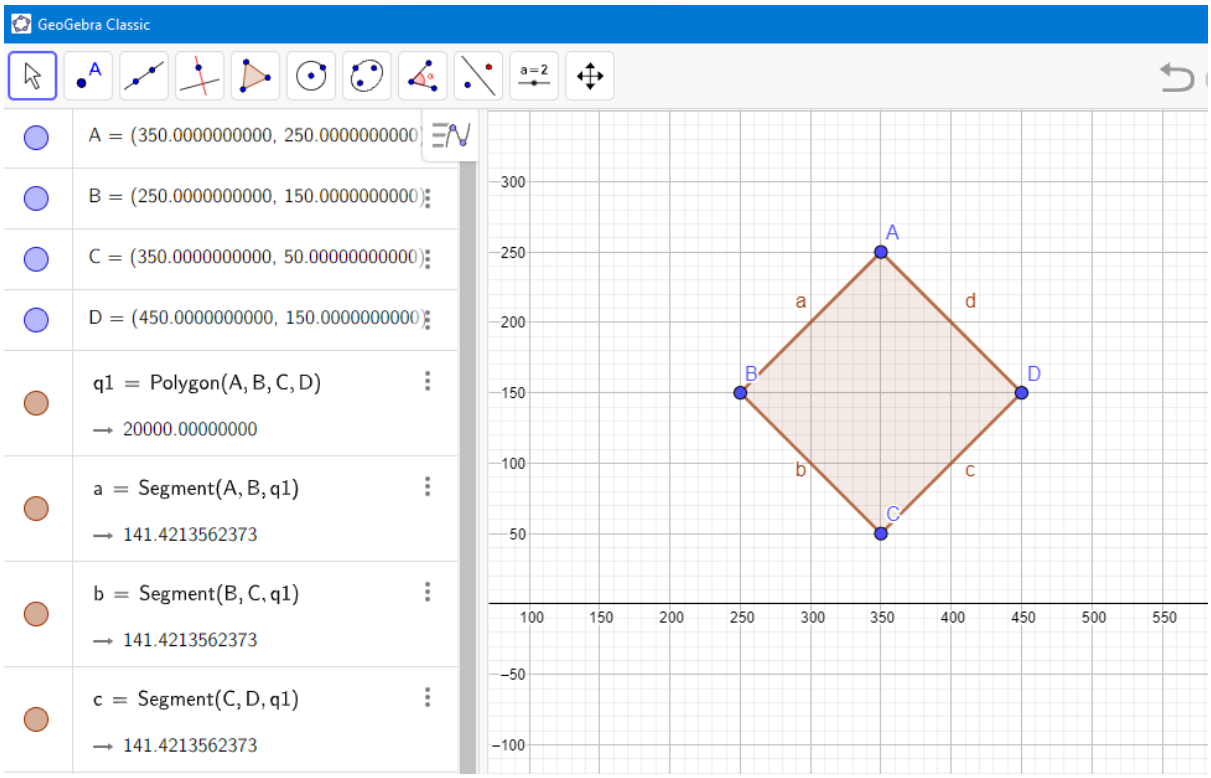
KOTRETAN

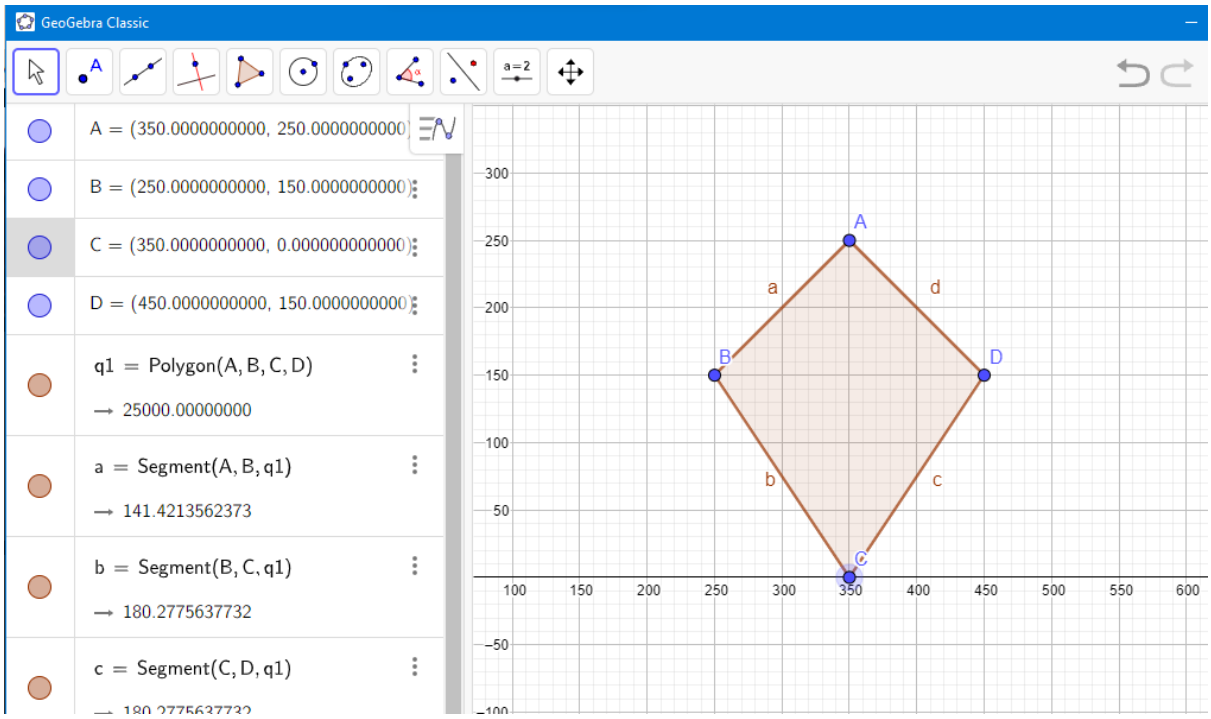
Kontretan selama pembuatan beberapa bidang dasar dengan mencoba di geogebra agar lebih gampang dalam membuat garis pada setiap bidangnya




GeoGebra Classic
A = (50.0000000000, 200.0000000000)
B = (250.0000000000, 200.0000000000)
C = (300.0000000000, 50.0000000000)
D = (50.0000000000, 50.0000000000)
q1 = Polygon(A, B, C, D)
→ 33750.00000000
a = Segment(A, B, q1)
→ 200.0000000000
b = Segment(B, C, q1)
→ 158.1138830084
c = Segment(C, D, q1)
→ 250.0000000000
d = Segment(D, A, q1)
A
B
C
D
a
b
c
d


GeoGebra Classic
A = (750.0000000000, 400.0000000000)
B = (950.0000000000, 400.0000000000)
C = (750.0000000000, 250.0000000000)
D = (550.0000000000, 250.0000000000)
q1 = Polygon(A, B, C, D)
→ 30000.00000000
a = Segment(A, B, q1)
→ 200.0000000000
b = Segment(B, C, q1)
→ 250.0000000000
c = Segment(C, D, q1)
A
B
C
D
a
b
c
d





LESSON LEARN

Setelah saya mengerjakan tugas ini, materi yang diajarkan pada mata kuliah teori menjadi lebih jelas lagi. Karena di tugas sebelumnya ditugaskan mengenai manipulasi garis, jadi function yang telah digunakan bisa digunakan di dalam tugas pertemuan minggu 4 ini. Yang menjadi kendala adalah mengenai logikanya

yang sering error jadi hasilnya tidak sesuai dengan keinginan. Selain itu juga kesalahan yang sering terjadi adalah kesalahan penempatan looping yang terjadi berulang kali sehingga ketika dijalankan akan not responding aplikasinya.

CURHAT DOSEN

Untuk tugas godot ini terasa lebih terbayang karena sebelumnya telah mengerjakan task mengenai manipulasi garis. Yang paling sulit itu menurut saya adalah bagaimana mengkonversi logika yang kita punya ke dalam godot script. Mungkin hal yang membuat mudah itu ketika sudah ada contoh dari logika program bahasa lain, contohnya yang elips dan lingkaran ini. Selama mengkonversi kita jadi sambil mikir tentang logikanya dan jadi lebih paham lagi ttg perilaku gdscript kalau dibedakan dengan bahasa pemrograman lain.