MODUL III

KOMPUTER GRAFIK 2D
BENTUK DASAR LANJUTAN DAN ATRIBUT GARIS

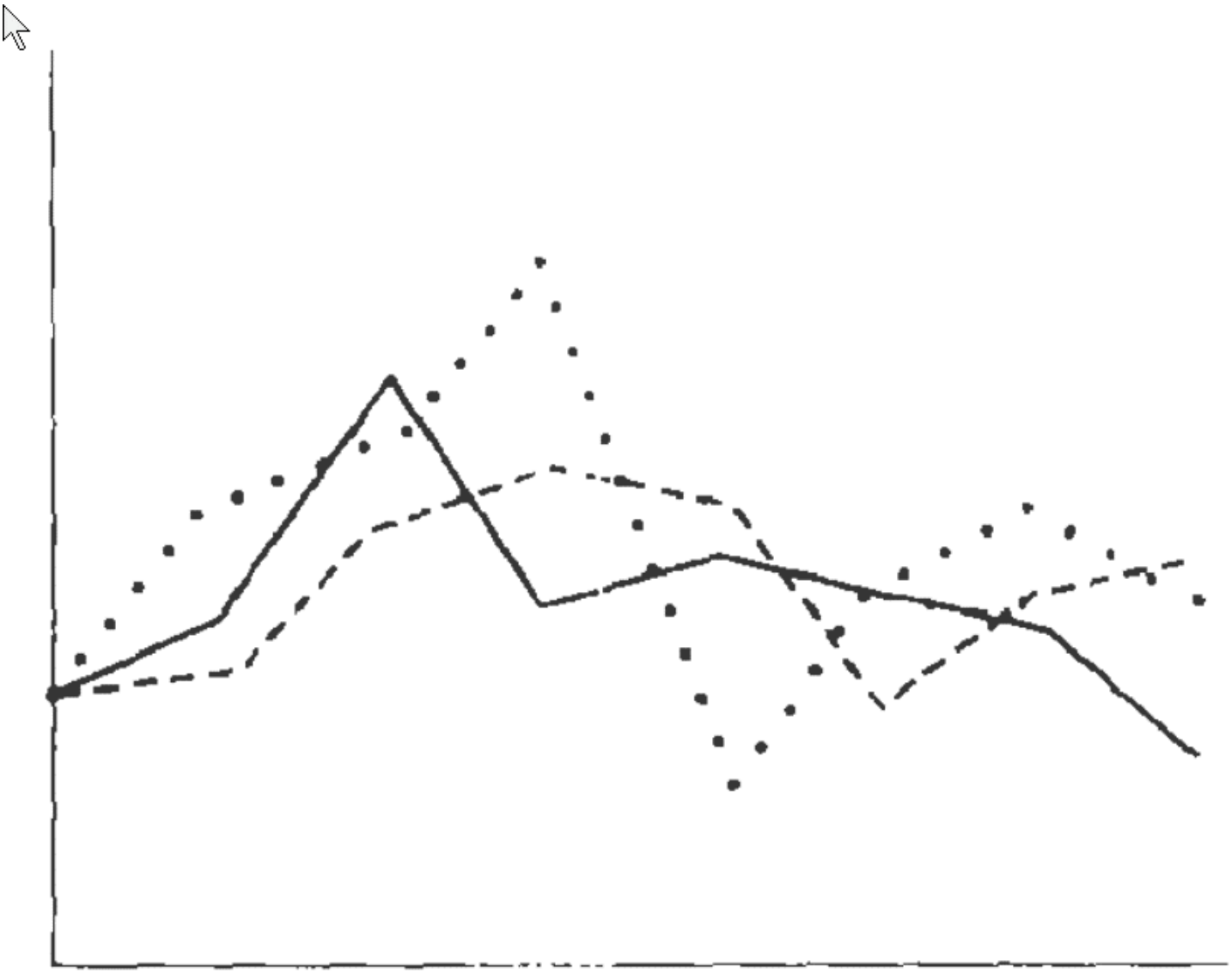
D3 TEKNIK INFORMATIKA
JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA
POLITEKNIK NEGERI BANDUNG

IRFANSYAH 089 | KOMPUTER GRAFIK | SEPTEMBER, 2 2024

CONTENTS

ATRIBUTE GARIS

RAGAM ATTRIBUTE GARIS



TIC TAC TOE GAME

https://playtictactoe.org/

Permainan Tic-Tac-Toe atau catur jawa atau XOX merupakan permainan classic yang digunakan untuk belajar pemrograman game. Selain Tic-Tac-Toe ada minesweeper, digger, snake, dan pong. Aturan main Tic-Tac-Toe sangat sederhana, 2 pemain berusaha menyelesaikan kondisi kemenangan yaitu ketika terdapat tiga tanda yang sama pada posisi vertikal, horizontal atau diagonal secara berurutan. Sebaliknya permainan akan draw jika 2 pemain tidak dapat menghasilkan kondisi tersebut.

Pemain 1 menulis X dan Pemain 2 menulis O pada papan 3x3

Pada praktikum sebelumnya kita telah membuat bentuk dasar lingkaran, garis dan kali dan akan memanfaatkan kode tersebut untuk membuat permainan ini.

Fitur-fitur Permainan Tic Tac Toe
1. Papan Grid 3x3 2. Pemain 1 bisa menuliskan X di Papan & Pemain 2 bisa menuliskan O di Papan 3. Kolom yang sudah dituliskan tidak boleh dituliskan kembali 4. Kondisi draw tidak ada X atau O yang sesuai dengan kondisi (Vertikal, Horizontal, Diagonal) 5. Kondisi menang ada X atau O yang sesuai dengan kondisi (Vertikal, Horizontal, Diagonal)

TASK PRAKTIKUM

TASK 1-2: PEMBELAJARAN OOP PYTHON DENGAN PROCESSING

1. Amati implementasi code untuk membuat tictactoe, Tictactoe, Asteroid, Stick Man, Kendaraan
2. Setelah diamati, modifikasi kode untuk memahami maksud dari OOP dan tuliskan temuan yang didapatkan.

Lesson Learnt (Hasil Karya, dan Komentar)

Pada praktikum task 0 main.py
Saya berhasil memperbaiki error, ketika posisi blok sudah diisi namun akan diisi lagi akan ke tumpuk. Pada penyelesaiannya saya membuat label dengan pendefinisian (posisi, 0)
Pada massing – massing posisi saya definisikan label nya 0, ketika diisikan label nya menjadi 1. Kemudian ketika, blok akan diisi simbol x atau o saya membuat kondisi dulu jika label == 0 maka boleh diisi, jika label == 1 blok tidak boleh diisi.

```
def draw():
    py5.background(191)
    (top_left,0)
    (top_middle,0)
    (top_right,0)
    (middle_left,0)
    (middle_middle,0)
    (middle_right,0)
    (bottom_left,0)
    (bottom_middle,0)
    (bottom_right,0)
```

Pendefinisian

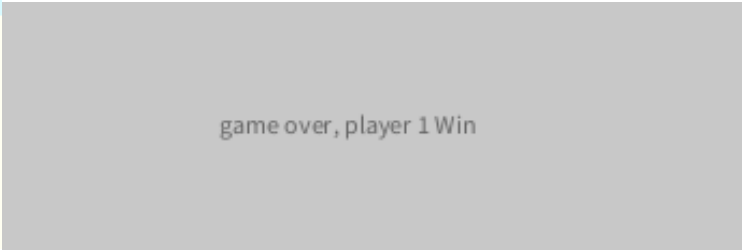
```
if((py5.mouse_x > 0 and py5.mouse_x < r) and (py5.mouse_y > 0 and py5.mouse_y < r )):
    if top_left == 0:
        top_left = turn
        switch_turns()
elif((py5.mouse_x > r and py5.mouse_x < r*2) and (py5.mouse_y > 0 and py5.mouse_y < r )):
    if top_middle == 0:
        top_middle = turn
        switch_turns()
elif((py5.mouse_x > r*2 and py5.mouse_x < r*3) and (py5.mouse_y > 0 and py5.mouse_y < r )):
    if top_right == 0:
        top_right = turn
        switch_turns()
```

Potongan Code pengecekan label sebelum dilakukan pengisian simbol dan switch pemain

Pada mainv2.py
Saya berhasil membuat kondisi menang ketika blok yang berurutan adalah diagonal, dengan membuat kondisi ketika grid [0,0] [1,1] [2,2] simbolnya sama. Dan membuat kondisi ketika grid [0,2] [1,1] [2,0] simbolnya sama
Saya juga berhasil membuat kondisi draw dengan cara mengecek seluruh posisi grid dengan nested loop 3 perulangan Jika hingga iterasi terakhir terisi simbol semua, maka class akan mereturn 1

```
def find_draw():
    aa = 0
    for x in range(3):
        for y in range(3):
            if grid[x][y] != 0:
                aa = 1;
            else:
                aa = 0;
                break;
        if aa == 0:
            break;
    if aa == 1:
        return aa
```

```python
def find_winners():

    for y in range(3):
        if (grid[y][0] == grid[y][1] == grid[y][2] != 0):
            return grid[y][0]
        elif (grid[0][y] == grid[1][y] == grid[2][y] != 0):
            return grid[0][y]
    if (grid[0][0] == grid[1][1] == grid[2][2] != 0):
            return grid[y][y]
    elif (grid[0][2] == grid[1][1] == grid[2][0] != 0):
            return grid[1][1]
```



Pada mainv3.py sebelumnya ketika dijalankan hanya bisa menangani kondisi menang horizontal dan tidak bisa menangani draw. Untuk kondisi menang dan draw saya code nya sama dengan mainv2.py. Pengecekan menang dan draw nya dilakukan sesudah player memilih simbol dan sesudah ai memilih simbol

Pada Task 2 saya membuat

- class Mobil digunakan untuk mendefinisikan dan menyimpan informasi dasar tentang nama mobil
- Class DaftarMobil digunakan untuk menyimpan dan mengelola daftar mobil. Ini berfungsi sebagai penyimpanan yang menyimpan semua mobil beserta informasi terkait.
- Kemudian saya membuat objek dengan membuat nama terlebih dahulu dengan memanggil class mobil
- kemudian menambahkannya ke daftar mobil ke class daftarmobil, nantinya objek yang sudah dibentuk di class dapat dipanggil lagi setelah melalui proses manipulasi di class

TASK 4: MEMBUAT FUNGSI BENTUK DASAR

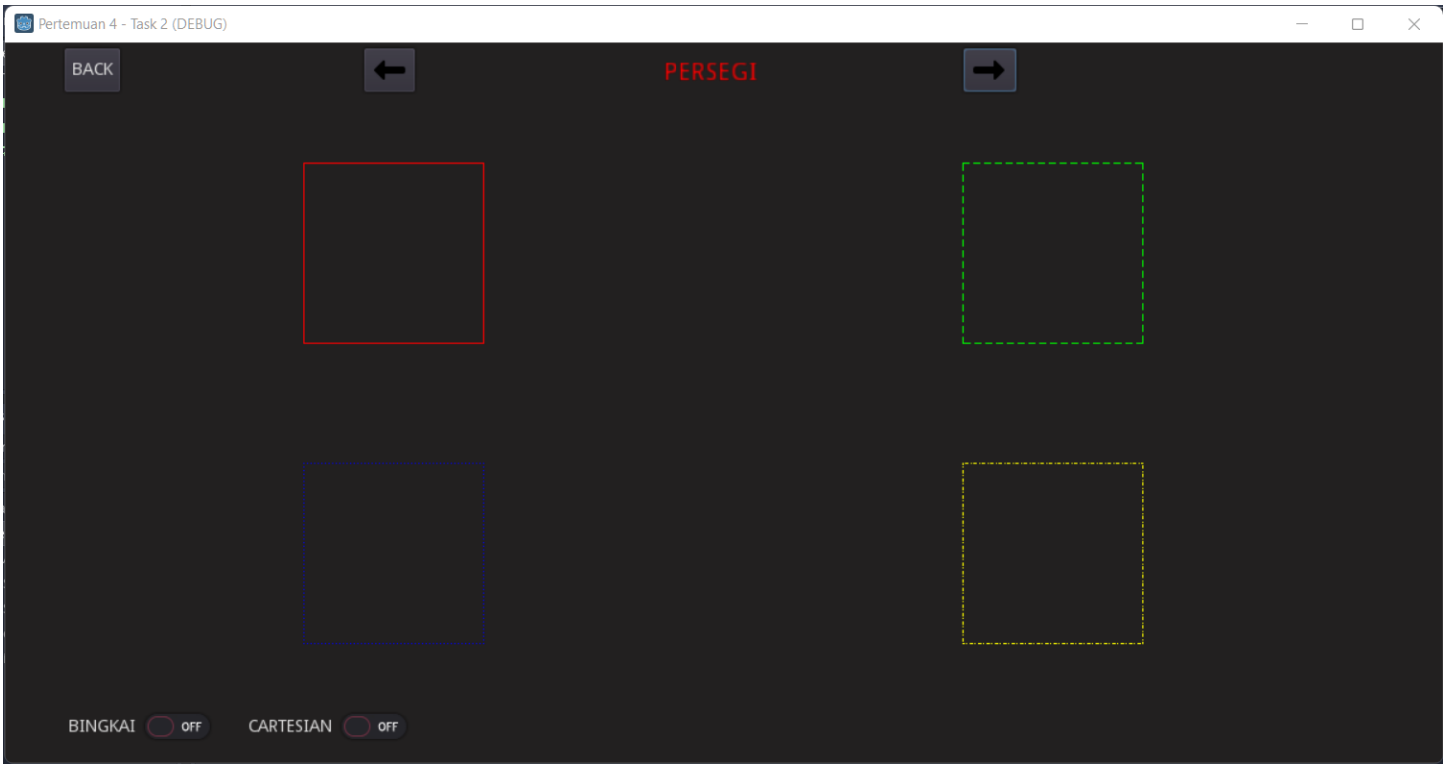
1. Buatlah Class Bentuk Dasar menggunakan algoritma generalisasi line bersenham sbb: Persegi, Persegi Panjang, Segitiga Siku-Siku, dan Trapesium Siku-Siku
2. Buatlah Ragam Attribute Garis untuk setiap Bentuk Dasar. Hints (jadikan parameter bukan hardcoding, manipulasi dilakukan setelah koordinat garis terbentuk atau manipulasi hasil array of koordinat)
3. Posisikan Bentuk Dasar menjadi 4 Quadran.

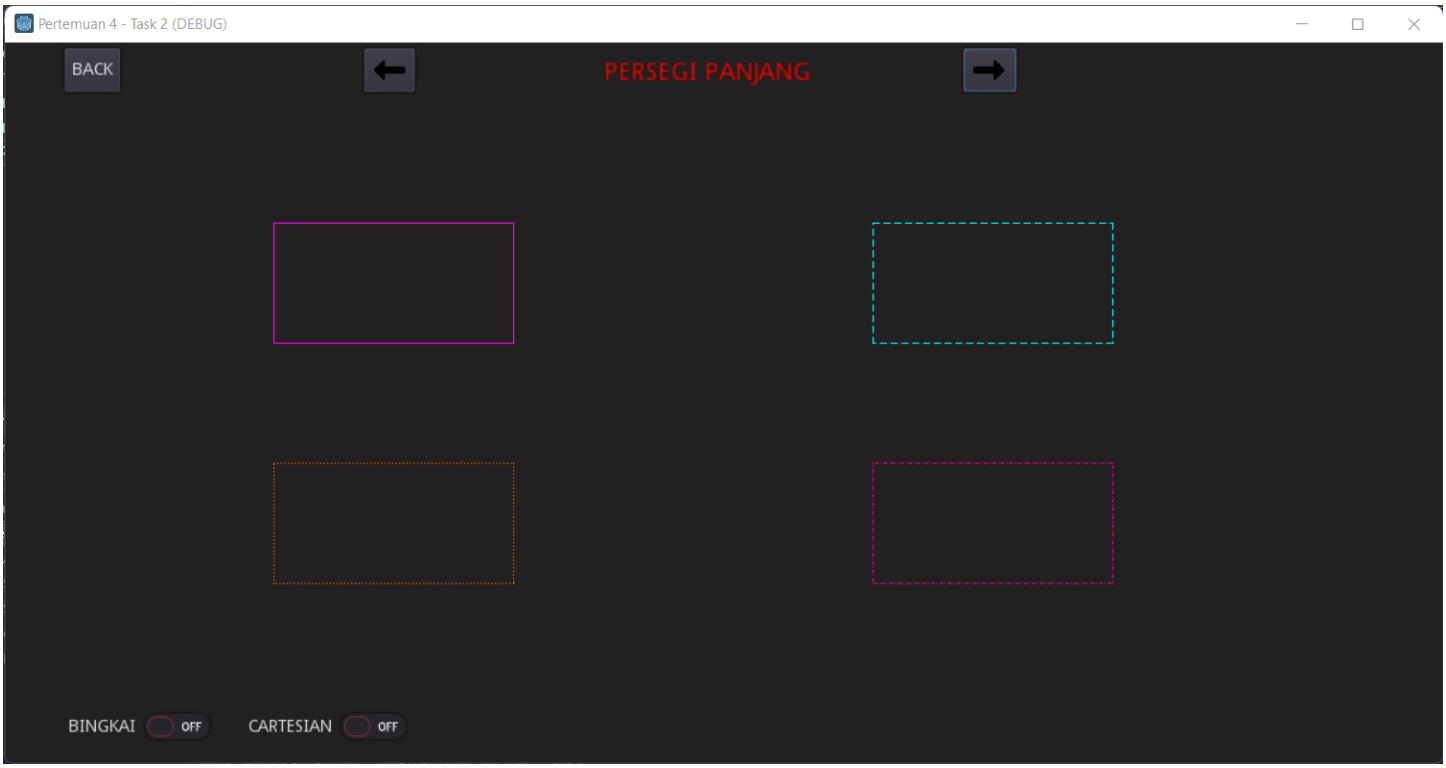
Untuk mengubah posisi dapat menggunakan fungsi convert to cartesian berikut

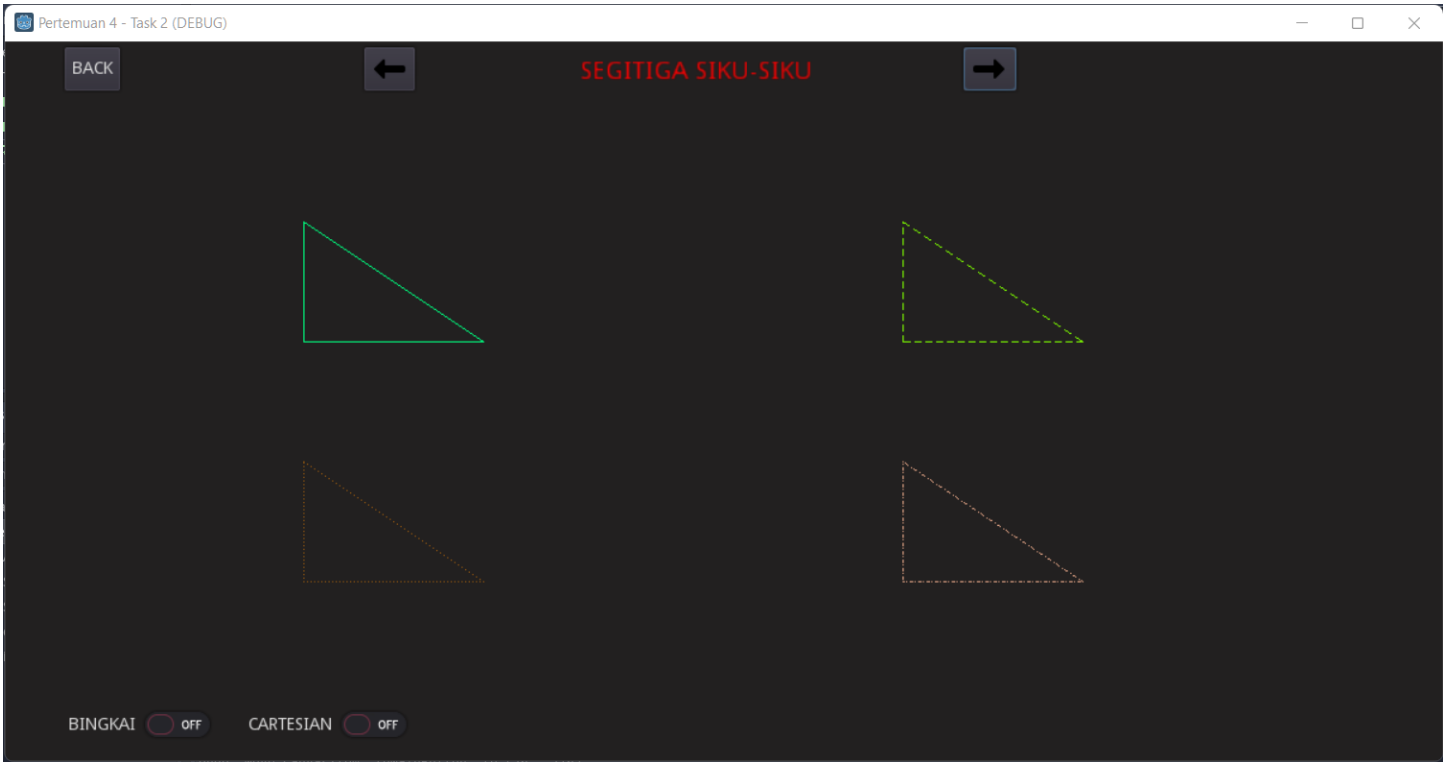
Utility.py

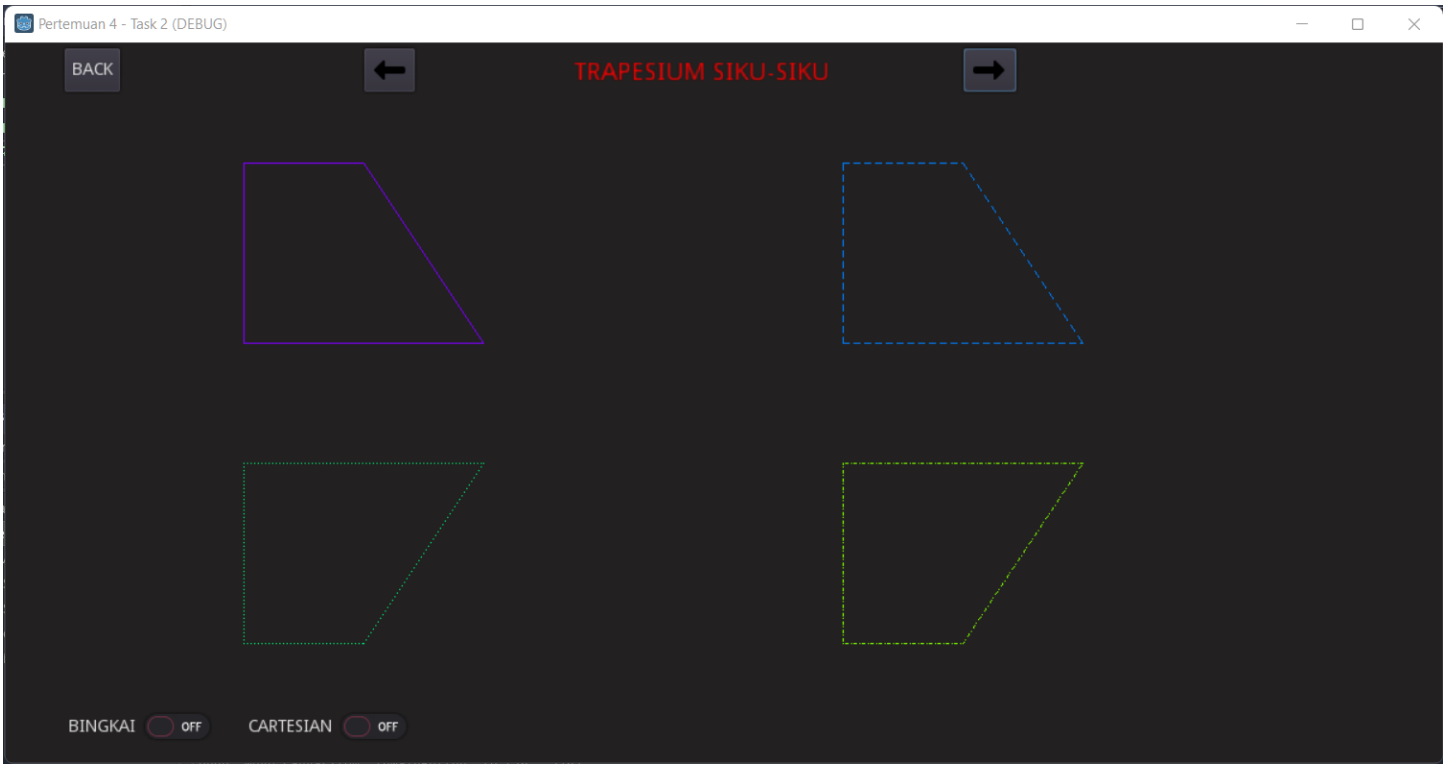
```python
import math

def convert_to_pixel(xa, ya, xb, yb, width, height, margin):
    return [margin+xa, height-margin-ya, margin+xb, height-margin-yb]
```

```
def convert_to_cartesian(xa, ya, xb, yb, width, height, margin):
    axis = math.ceil(width/2)
    ordinat = math.ceil(height/2)
    return [axis+xa, ordinat-ya, axis+xb, ordinat-yb]
```




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)

Basic.py

```
import primitif.line
import py5
import numpy as np

def round(x):
    return int(x+0.5)

def draw_margin(width, height, margin, c=[0,0,0,255]):
    py5.stroke(c[0], c[1], c[2], c[3])
    py5.points(primitif.line.line_dda(margin,margin,width-margin,margin))
    py5.points(primitif.line.line_dda(margin,height-margin,width-margin,height-margin))
    py5.points(primitif.line.line_bresenham(margin,margin,margin,height-margin,1))
    py5.points(primitif.line.line_bresenham(width-margin,margin,width-margin,height-margin,1))

def draw_grid(width, height, margin, c=[0,0,0,255]):
    # Sumbu Y
    xa = margin;
    ya = 2*margin;
    xb = width - xa
    yb = height - ya
    y_range = (height / margin)

    py5.stroke(c[0], c[1], c[2], c[3])
    for count in range(1, int(y_range)):
        py5.points(primitif.line.line_dda(xa,ya,xb,ya))
```

```
        ya = ya + margin

    # Sumbu X
    xa = 2*margin
    ya = margin
    xb = width - xa
    yb = height - ya
    x_range = (width / margin)
    for count in range(1, int(x_range)):
        py5.points(primitif.line.line_dda(xa,ya,xa,yb))
        xa = xa + margin

def draw_kartesian(width, height, margin, c=[0,0,0,255]):
    py5.stroke(c[0], c[1], c[2], c[3])
    py5.points(primitif.line.line_dda(width/2,margin,width/2,height-margin))
    py5.points(primitif.line.line_bresenham(margin,height/2,width-margin,height/2,1))

def persegi(xa, ya, panjang, pattern, c=[0,0,0,255]):
    py5.stroke(c[0], c[1], c[2], c[3])
    py5.points(primitif.line.line_bresenham(xa-panjang/2,ya-panjang/2,xa+panjang/2,ya-panjang/2,pattern))
    py5.points(primitif.line.line_bresenham(xa-panjang/2,ya+panjang/2,xa+panjang/2,ya+panjang/2,pattern))
    py5.points(primitif.line.line_bresenham(xa-panjang/2,ya-panjang/2,xa-panjang/2,ya+panjang/2,pattern))
    py5.points(primitif.line.line_bresenham(xa+panjang/2,ya-panjang/2, xa+panjang/2,ya+panjang/2,pattern))

def persegi_panjang(xa, ya, panjang, lebar, pattern, c=[0,0,0,255]):
    py5.stroke(c[0], c[1], c[2], c[3])
    py5.stroke(10)
    py5.points(primitif.line.line_bresenham(xa-panjang/2, ya-lebar/2, xa+panjang/2, ya-lebar/2,pattern))
    py5.points(primitif.line.line_bresenham(xa-panjang/2, ya-lebar/2, xa-panjang/2, ya+lebar/2,pattern))
    py5.points(primitif.line.line_bresenham(xa+panjang/2, ya-lebar/2, xa+panjang/2, ya+lebar/2,pattern))
    py5.points(primitif.line.line_bresenham(xa-panjang/2, ya+lebar/2, xa+panjang/2, ya+lebar/2,pattern))

def segitiga_siku(xa, ya, alas, tinggi, pattern, c=[0,0,0,255]):
    py5.points(primitif.line.line_bresenham(xa-alas/2, ya-tinggi/2, xa-alas/2, ya+tinggi/2,pattern))
    py5.points(primitif.line.line_bresenham(xa-alas/2, ya+tinggi/2, xa+alas/2, ya+tinggi/2,pattern))
    py5.points(primitif.line.line_bresenham(xa-alas/2, ya-tinggi/2, xa+alas/2, ya+tinggi/2,pattern))


def trapesium_siku(xa, ya, atas, bawah, tinggi, pattern, c=[0,0,0,255]):
    py5.stroke(c[0], c[1], c[2], c[3])
    py5.points(primitif.line.line_bresenham(xa-bawah/2, ya-tinggi/2, xa-bawah/2, ya+tinggi/2,pattern))
    py5.points(primitif.line.line_bresenham(xa-bawah/2, ya+tinggi/2, xa+bawah/2, ya+tinggi/2,pattern))
    py5.points(primitif.line.line_bresenham(xa-bawah/2, ya-tinggi/2, xa-(bawah/2)+atas, ya-tinggi/2,pattern))
    py5.points(primitif.line.line_bresenham(xa-(bawah/2)+atas, ya-tinggi/2, xa+bawah/2, ya+tinggi/2,pattern))

def kali(xa, ya, panjang, c=[255,0,0,255]):
    py5.stroke(c[0], c[1], c[2], c[3])
    py5.points(primitif.line.line_bresenham(xa,ya,xa+panjang,ya+panjang))
    py5.points(primitif.line.line_bresenham(xa,ya+panjang,xa+panjang,ya))

def circlePlotPoints(xc, yc, x, y):
    res = [
        [xc + x, yc + y],
        [xc - x, yc + y],
        [xc + x, yc - y],
        [xc - x, yc - y],
        [xc + y, yc + x],
        [xc - y, yc + x],
        [xc + y, yc - x],
        [xc - y, yc - x],
        ]
    return np.array(res)

def lingkaran(xc, yc, radius, c=[255,0,0,255]):
    x = 0
    y = radius
    p = 1 - radius
```

```
    py5.stroke(c[0], c[1], c[2], c[3])
    py5.points(circlePlotPoints(xc, yc, x, y))

    while(x < y):
        x+=1
        if (p < 0):
            p+= 2*x + 1
        else:
            y-=1
            p+= 2*(x-y) + 1
        py5.points(circlePlotPoints(xc, yc, x, y))

def ellipsePlotPoints(xc, yc, x, y):
    pass

def ellips(xc, yc, Rx, Ry, c=[255,0,0,255]):
    pass
```

Line.py

```
import numpy as np
import math

def round(x):
    return int(x+0.5)

def line_dda(xa, ya, xb, yb):

    dx = abs(xb - xa)
    dy = abs(yb - ya)
    length = max(dx,dy)

    dx = (xb-xa)/length
    dy = (yb-ya)/length

    x = xa
    y = ya

    res = [[xa, ya]]

    for i in range(length+1):
        res.append([round(x), round(y)])
        x = x+dx
        y = y+dy

    return np.array(res)

class Pattern():
    SOLID = 1
    DASHED = 2
    DOTTED = 3
    DASH_DOT = 4

def line_bresenham(xa, ya, xb, yb, pattern):


    if abs(yb - ya) < abs(xb - xa):
        if xa > xb:
            return np.array(line_low(xb, yb, xa, ya, pattern))
        else:
            return np.array(line_low(xa, ya, xb, yb, pattern))
```

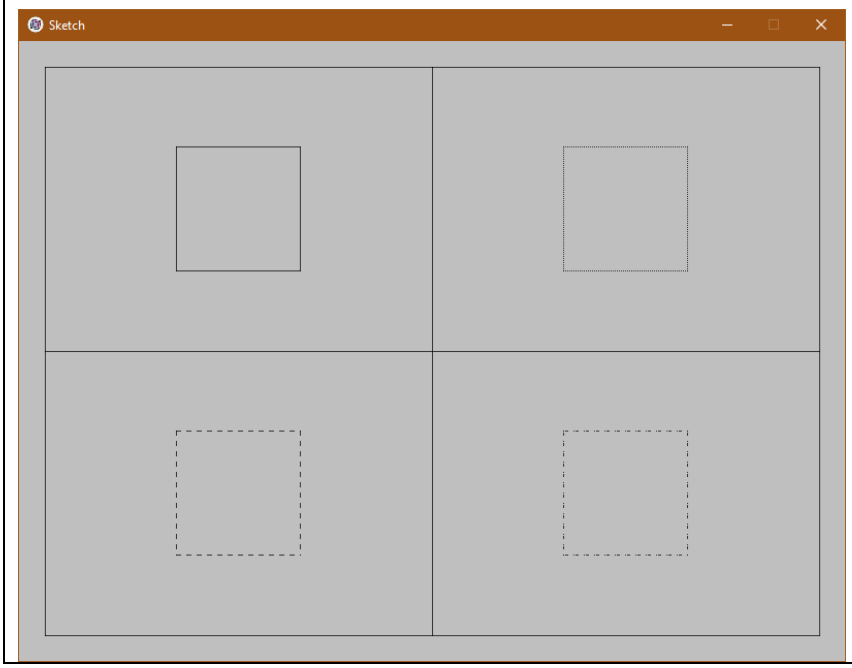
```
    else:
        if ya > yb:
            return np.array(line_high(xb, yb, xa, ya, pattern))
        else:
            return np.array(line_high(xa, ya, xb, yb, pattern))

def line_low(xa, ya, xb, yb, pattern):
    res = []
    dx = xb - xa
    dy = yb - ya
    yi = 1
    if dy < 0:
        yi = -1
        dy = -dy
    p = (2 * dy) - dx
    twody = 2 * dy
    twodydx = 2 * (dy - dx)
    y = ya
    pattern_counter = 0
    dash_length = 5
    gap_length = 5

    for x in range(int(xa), int(xb + 1)):
        if pattern == Pattern.SOLID:
            res.append([x, y])
        elif pattern == Pattern.DASHED:
            if pattern_counter < dash_length:
                res.append([x, y])
            pattern_counter = (pattern_counter + 1) % (dash_length + gap_length)
        elif pattern == Pattern.DOTTED:
            if pattern_counter == 0:
                res.append([x, y])
                pattern_counter = 1
            elif pattern_counter == 1:
                pattern_counter = 0
        elif pattern == Pattern.DASH_DOT:
            if pattern_counter < dash_length:
                res.append([x, y])
            elif pattern_counter < dash_length + gap_length:
                pass  # no dot here
            pattern_counter = (pattern_counter + 1) % (2 * (dash_length + gap_length))

        if p > 0:
            y += yi
            p += twodydx
        else:
            p += twody

    return res

def line_high(xa, ya, xb, yb, pattern):
    res = []
    dx = xb - xa
    dy = yb - ya
    xi = 1
    if dx < 0:
        xi = -1
        dx = -dx
    p = (2 * dx) - dy
    twodx = 2 * dx
```

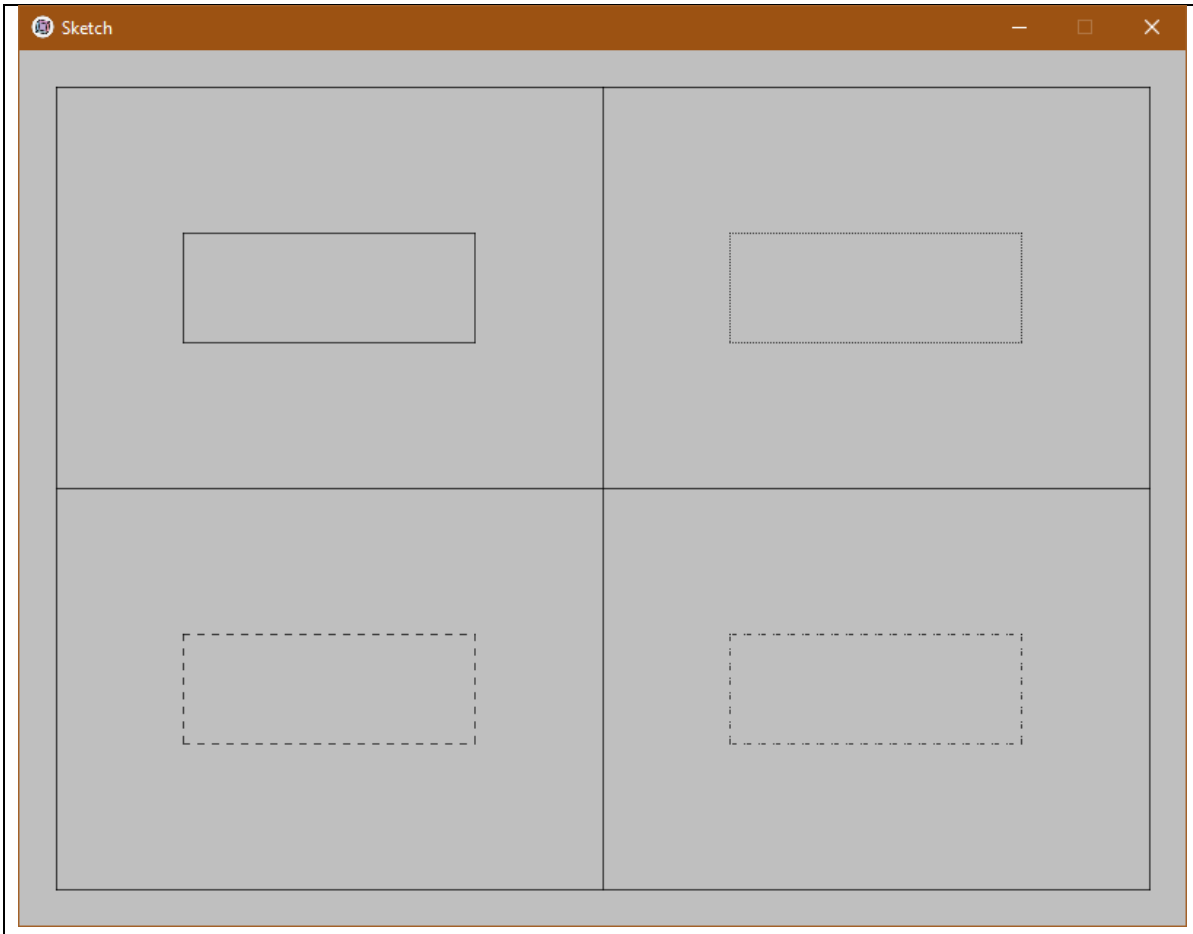
```
    twodxdy = 2 * (dx - dy)
    x = xa
    pattern_counter = 0
    dash_length = 5
    gap_length = 5

    for y in range(int(ya),int( yb + 1)):
        if pattern == Pattern.SOLID:
            res.append([x, y])
        elif pattern == Pattern.DASHED:
            if pattern_counter < dash_length:
                res.append([x, y])
            pattern_counter = (pattern_counter + 1) % (dash_length + gap_length)
        elif pattern == Pattern.DOTTED:
            if pattern_counter == 0:
                res.append([x, y])
                pattern_counter = 1
            elif pattern_counter == 1:
                pattern_counter = 0
        elif pattern == Pattern.DASH_DOT:
            if pattern_counter < dash_length:
                res.append([x, y])
            elif pattern_counter < dash_length + gap_length:
                pass  # no dot here
            pattern_counter = (pattern_counter + 1) % (2 * (dash_length + gap_length))

        if p > 0:
            x += xi
            p += twodxdy
        else:
            p += twodx

    return res
```

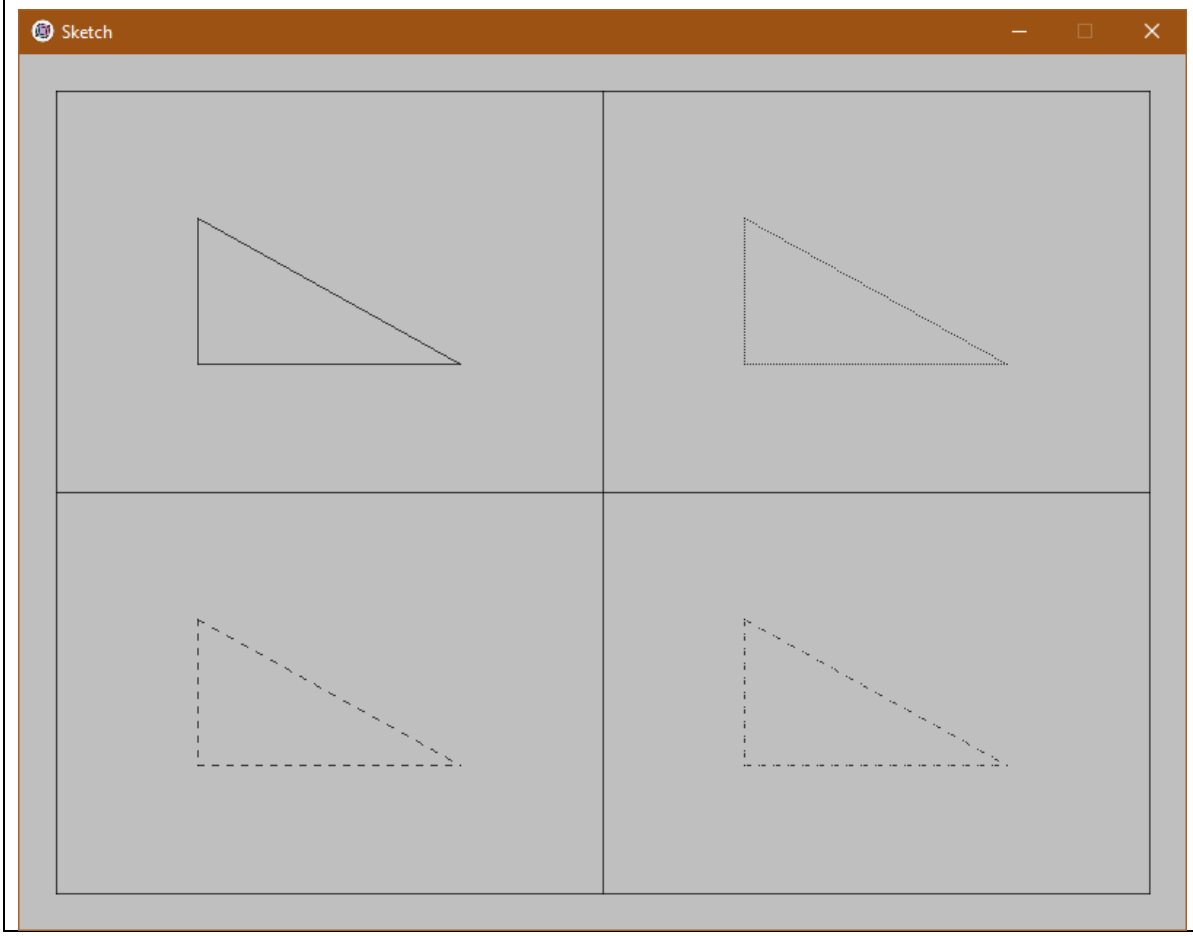
Main.py

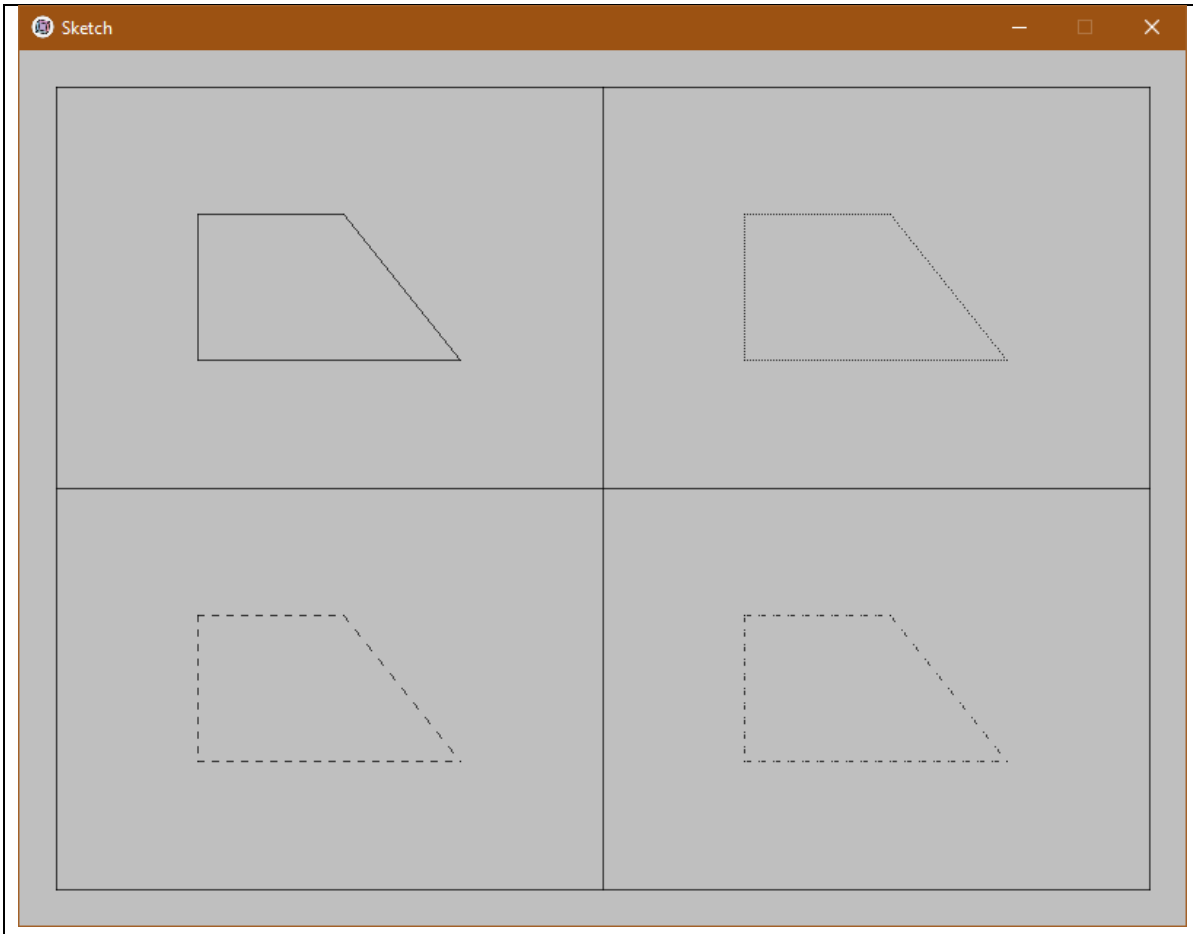
```
import py5
import primitif.line
import primitif.basic
import primitif.utility
import math
import config

margin = 25

def setup():
    py5.size(800, 600)
    py5.rect_mode(py5.CENTER)

def draw():
    py5.background(191)
    primitif.basic.draw_margin(py5.width, py5.height, margin, c=[0,0,0,255])
    primitif.basic.draw_kartesian(py5.width, py5.height, margin, c=[0,0,0,255])


    width_2 = (py5.width-2*margin)/4
    height_2 = (py5.height-2*margin)/4
    positionX = margin + width_2
    positionY = margin + height_2
```

```
    if config.anim <= config.times:
        y = 1
        for i in range(0, 4, 2):
            for x in range(0, 4, 2):
                primitif.basic.persegi(positionX + i*width_2, positionY + x*height_2, 120, y, c=[0,0,0,255])
                y = y+1

    elif config.anim <= 2*config.times:
        y = 1
        for i in range(0, 4, 2):
            for x in range(0, 4, 2):
                primitif.basic.persegi_panjang(positionX + i*width_2, positionY + x*height_2, 200, 75, y, c=[0,0,0,255])
                y = y+1

    elif config.anim <= 3*config.times:
        y = 1
        for i in range(0, 4, 2):
            for x in range(0, 4, 2):
                primitif.basic.segitiga_siku(positionX + i*width_2, positionY + x*height_2, 180, 100, y, c=[0,0,0,255])
                y = y+1

    elif config.anim <= 4*config.times:
        y = 1
        for i in range(0, 4, 2):
            for x in range(0, 4, 2):
                primitif.basic.trapesium_siku(positionX + i*width_2, positionY + x*height_2, 100, 180, 100, y, c=[0,0,0,255])
                y = y+1

    if config.anim > 4*config.times:
        config.anim = 0

    config.anim += 1

py5.run_sketch()
```




Sketch


Sketch



Untuk membuat bentuk-bentuk geometris dengan ukuran dan tipe garis yang dapat disesuaikan, kita perlu mendefinisikan sebuah kelas yang memungkinkan setiap objeknya memiliki atribut yang dinamis. Atribut-atribut ini dapat diubah sesuai dengan kebutuhan objek yang memanggilnya.

Dalam membuat 16 objek dengan bentuk dan tipe garis yang berbeda, saya menggunakan konsep looping untuk efisiensi, di mana terdapat nested loop untuk mengatur peletakan bangun datar menjadi empat bagian.

Untuk menentukan variasi tipe garis, sebelumnya saya mendefinisikan beberapa jenis garis dengan nilai 1 hingga 4 yang mewakili solid, dashed, dotted, dash_dot.

Di dalam loop, nilai tipe garis ini dapat berubah secara dinamis, dengan menambahkan sebuah variabel yang dimulai dari 1 dan diincrement pada setiap iterasi. Sehingga setiap objek yang dihasilkan memiliki karakteristik yang unik dalam hal bentuk dan tipe garis.

PENGUMPULAN

Ikuti Format yang diberikan di Google Classroom.