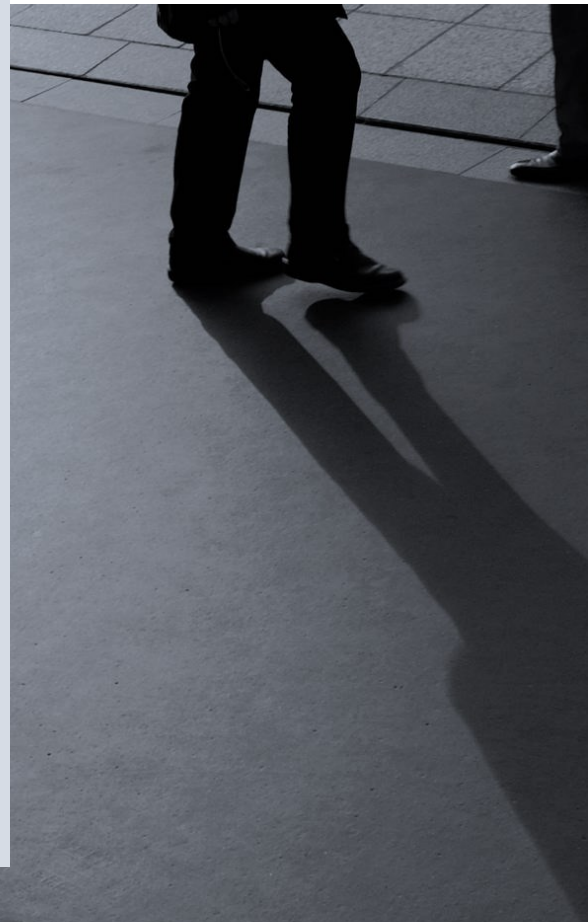


MODUL II

KOMPUTER GRAFIK 2D
PENGENALAN PYTHON DAN PROCESSING

D3 TEKNIK INFORMATIKA
JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA
POLITEKNIK NEGERI BANDUNG



IRFANSYAH 089 | KOMPUTER GRAFIK | AGUSTUS, 26 2024

CONTENTS

PRIMITIF 2D 1

PERSAMAAN GARIS 1

LINE DDA 2

LINE BERSENHAM 3

GENERALISASI LINE BERSENHAM..... 6

TASK PRAKTIKUM 8

PENGUMPULAN..... 15

PRIMITIF 2D

Apa itu gambar? Gambar adalah kumpulan dari objek yang kompleks, seperti gambar pemandangan yang terdiri dari gunung, pohon, sawah, matahari dan awan yang diposisikan dalam sebuah sistem koordinat. Objek kompleks tersusun dari bentuk-bentuk primitif seperti titik, garis, lingkaran, ellips dan bentuk dasar turunan lainnya.

Dalam komputer grafik, struktur geometri dasar disebut dengan output primitives. Titik dan Garis merupakan primitif paling sederhana dalam komputer grafik. Pada Py5 Processing, terdapat dua fungsi untuk membuat titik yaitu :

point	https://py5coding.org/reference/sketch_point.html menggambar titik, pada sistem koordinat dengan dimensi 1 pixel <pre>point(x: float, # x-coordinate of the point y: float, # y-coordinate of the point /,) -> None</pre>
points	https://py5coding.org/reference/sketch_points.html menggambar kumpulan titik, pada sistem koordinat dengan dimensi 1 pixel <pre>points(coordinates: npt.NDArray[np.floating], # 2D array of point coordinates with 2 or 3 columns for 2D or 3D points, respectively /,) -> None</pre> Pada points, Kumpulan titik merupakan Numpy Array

Dari primitif titik, kita dapat menggambar garis di layer komputer menggunakan algoritma yang berasal dari persamaan matematika garis.

PERSAMAAN GARIS

Persamaan garis yang umum diketahui yaitu The cartesian slope intercept equation.

$$y = mx + b$$

Jika diketahui dua endpoints dari line segment (x_1, y_1) dan (x_2, y_2) maka nilai slope m dan intercept b dapat diketahui

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

$$b = y_1 - m \cdot x_1$$

Untuk setiap nilai x pada interval dx , nilai y pada interval dy dapat diketahui dan juga sebaliknya.

$$dy = m \cdot dx$$

$$dx = \frac{dy}{m}$$

Hubungan slopes / magnitudes dengan dx dan dy

$ m < 1$	Mendekati Horizontal
-----------	----------------------



$ m > 1$	Mendekati Vertical
$ m = 1$	Diagonal

Pada sistem raster (layar), titik di plot menjadi garis dan ukuran step arah horizontal dan vertikal dibatasi dengan jarak antara pixel (integer). Maka ada proses sampling penempatan posisi garis secara diskrit dan menentukan pixel terdekat untuk setiap iterasi. Proses ini disebut dengan scan-conversion, terdapat dua algoritma pembentukan garis yaitu LINE DDA dan Line Bersenham.

LINE DDA

The digital differential analyzer (DDA) adalah algoritma line scan-conversion berdasarkan dari perhitungan antara dy atau dx pada persamaan garis. Proses sampling pada satu unit koordinat (x atau y) dan menentukan nilai integer terdekat yang sesuai dengan jalur garis dari koordinat lain.

Jika garis memiliki $m \leq 1$ maka sampling akan diambil pada koordinat x atau $dx = 1$ maka untuk mendapatkan nilai y berikutnya adalah:

$$y_{k+1} = y_k + m$$

Subscript k merupakan nilai integer yang dimulai dari 1, mulai dari titik pertama, bertambah 1 sampai titik terakhir. Karena nilai slope bernilai bilangan real antara 0 sampai 1 maka nilai y akan di bulatkan hingga nilai integer terdekat.

Sedangkan jika garis memiliki nilai $m \geq 1$ maka sampling akan diambil pada koordinat y atau $dy = 1$ maka untuk mendapatkan nilai x berikutnya adalah:

$$x_{k+1} = x_k + \frac{1}{m}$$

Asumsi sebaliknya jika jalur garis dari RTL maka $dy = -1$ dan $dx = -1$

Implementasi Line DDA pada py5

```
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]]

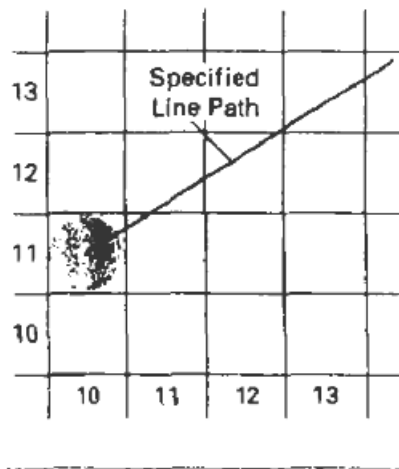
    for i in range(length+1):
        res.append([round(x), round(y)])
        x = x+dx
        y = y+dy
    return np.array(res)
```

Contoh Pemanggilan Line DDA menggunakan Points (kumpulan titik)

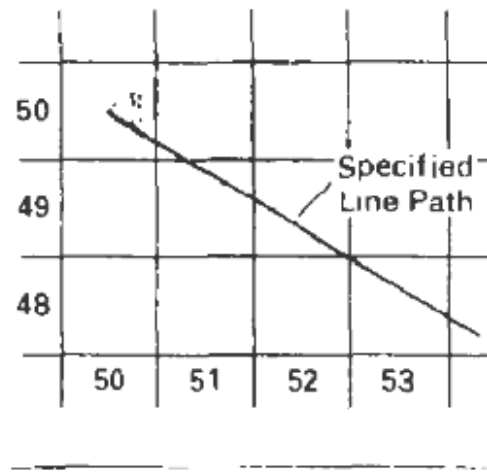
```
py5.stroke(0,randint(0,255),0,255)
py5.points(
  line_dda(
    randint(0,py5.width),randint(0,py5.height),
    randint(0,py5.width),randint(0,py5.height)
  )
)
```

LINE BERSENHAM

Line Bersenham adalah algoritma scan-conversion garis yang hanya membutuhkan kalkulasi nilai integer secara incremental dan dapat diadaptasi untuk membangun lingkaran dan bentuk kurva yang lain.



Gambar 1 Kasus I



Gambar 2 Kasus II

Kasus I slope ?, pilih x selanjutnya (11,11) atau (11, 12) ?

Kasus 2 slope ?, pilih x selanjutnya (51,49) atau (51,50) ?

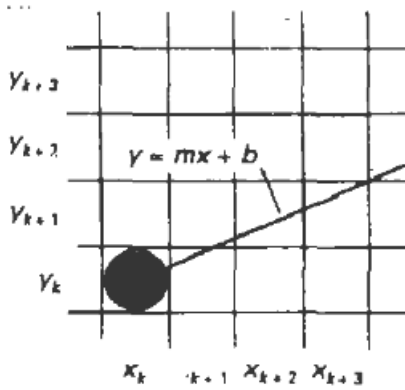


Figure 3-7
Section of the screen grid showing a pixel in column x_k on scan line y_k that is to be plotted along the path of a line segment with slope $0 < m < 1$.

Gambar 3 Ilustrasi Alur Garis

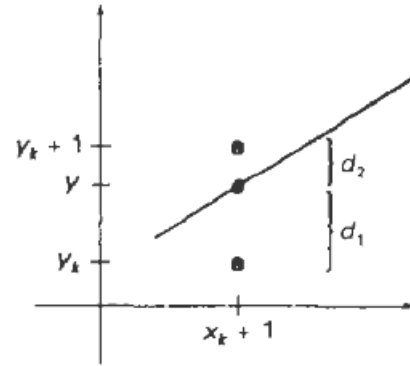


Figure 3-8
Distances between pixel positions and the line y coordinate at sampling position $x_k + 1$.

Gambar 4 Ilustrasi Jarak d_1 dan d_2

Perhatikan Gambar 3, merupakan studikusus ilustrasi garis yang akan dibangun menggunakan algoritma line bersenham, dengan slope $0 < m < 1$

1. Koordinat Posisi awal adalah (x_k, y_k)
2. Seperti line dda karena slope < 1 maka arah akan ke horizontal untuk mencari x selanjutnya antara $(x_k + 1, y_k)$ atau $(x_k + 1, y_k + 1)$
3. Perhatikan gambar 4 ilustrasi jarak d_1 dan d_2 untuk memilih point mana yang merupakan point selanjutnya jarak terkecil dari koordinat y dari mathematical line.

$$y = m(x_k + 1) + b$$

4. Perhitungan d_1 dan d_2

$$\begin{aligned} d_1 &= y - y_k \\ &= m(x_k + 1) + b - y_k \end{aligned}$$

$$\begin{aligned} d_2 &= y_k + 1 - y \\ &= y_k + 1 - m(x_k + 1) - b \end{aligned}$$

5. Delta dari dua jarak separasi d_1 dan d_2 adalah

$$d_1 - d_2 = 2m(x_k + 1) - 2y_k + 2b - 1$$

6. Untuk mendapatkan decision parameter / parameter yang akan menentukan titik integer mana yang dipilih dengan melakukan substitusi ke persamaan jarak separasi slope atau $m = \frac{dy}{dx}$

7. Definisi p_k

$$\begin{aligned} p_k &= dx(d_1 - d_2) \\ &= 2dy \cdot x_k - 2dx \cdot y_k + c \end{aligned}$$

8. Tanda dari decision parameter p_k sama dengan tanda dari delta jarak $d_1 - d_2$, Karena pada contoh $dx > 0$ dan parameter c adalah constant dan independent dari posisi pixel maka akan dieliminasi seiring dengan iterasi kalkulasi dari p_k .
9. Jika $d_1 - d_2 < 0$ maka y_k lebih dekat maka lower pixel yang dipilih sebaliknya lebih dekat dengan $y_k + 1$ upper pixel yang dipilih.

10. Perubahan koordinat pada jalur garis terjadi antara arah x atau y selanjutnya pemilihan decision parameter selanjutnya bergantung pada $k + 1$ steps melalui incremental kalkulasi integer.

$$p_{k+1} = 2dy \cdot x_{k+1} + 1 - 2dx \cdot x_k + 1 + c$$

11. Mengurangi p_{k+1} dan p_k menghasilkan dan $x_{k+1} = x_k$

$$p_{k+1} - p_k = 2dy(x_{k+1} - x_k) - 2dx(y_{k+1} - y_k)$$

$$p_{k+1} = p_k + 2dy - 2dx(y_{k+1} - y_k)$$

12. Dimana $y_{k+1} - y_k$ memiliki nilai antar 0 atau 1 tergantung pada tanda dari decision parameter p_k

13. Nilai p_0

$$p_0 = 2dy - dx$$

Bresenham's Line-Drawing Algorithm for $|m| < 1$

1. Input the two line endpoints and store the left endpoint in (x_0, y_0) .
2. Load (x_0, y_0) into the frame buffer; that is, plot the first point.
3. Calculate constants Δx , Δy , $2\Delta y$, and $2\Delta y - 2\Delta x$, and obtain the starting value for the decision parameter as

$$p_0 = 2\Delta y - \Delta x$$

4. At each x_k along the line, starting at $k = 0$, perform the following test:
If $p_k < 0$, the next point to plot is $(x_k + 1, y_k)$ and

$$p_{k+1} = p_k + 2\Delta y$$

Otherwise, the next point to plot is $(x_k + 1, y_k + 1)$ and

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$

5. Repeat step 4 Δx times.

Implementasi Line Bersenham pada py5

```
def line_bresenham(xa, ya, xb, yb):
    dx, dy = abs(xa-xb), abs(ya-yb)
    p = 2 * dy - dx
    twoDy, twoDyDx = 2 * dy, 2 * (dy - dx)
    x,y,xEnd = 0,0,0

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

```

res = [[x,y]]

while(x < xEnd):
    x+=1
    if (p<0):
        p+=twoDy
    else:
        y+=1
        p+=twoDyDx
    res.append([x,y])
return np.array(res)

```

Contoh Pemanggilan Line DDA menggunakan Points (kumpulan titik)

```

py5.stroke(0,randint(0,255),0,255)
py5.points(
    line_bresenham(
        randint(0,py5.width),randint(0,py5.height),
        randint(0,py5.width),randint(0,py5.height)
    )
)

```

GENERALISASI LINE BERSENHAM

Bresenham's algorithm is generalized to lines with arbitrary slope by considering the symmetry between the various octants and quadrants of the xy plane. For a line with positive slope greater than 1, we interchange the roles of the x and y directions. That is, we step along the y direction in unit steps and calculate successive x values nearest the line path. Also, we could revise the program to plot pixels starting from either endpoint. If the initial position for a line with positive slope is the right endpoint, both x and y decrease as we step from right to left. To ensure that the same pixels are plotted regardless of the starting endpoint, we always choose the upper (or the lower) of the two candidate pixels whenever the two vertical separations from the line path are equal ($d_1 = d_2$). For negative slopes, the procedures are similar, except that now one coordinate decreases as the other increases. Finally, special cases can be handled separately: Horizontal lines ($\Delta y = 0$), vertical lines ($\Delta x = 0$), and diagonal lines with $|\Delta x| = |\Delta y|$ each can be loaded directly into the frame buffer without processing them through the line-plotting algorithm.

Computer Graphics, C Version, Donald Hearn

Implementasi Generalisasi Line Bersenham pada py5
https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm

```

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

```



```

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

```

Line_low

```

def line_low(xa,ya,xb,yb):
    res = [[xa,ya]]
    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

    for x in range(xa,xb):
        res.append([x,y])
        if(p > 0):
            y += yi
            p += twodydx
        else:
            p += twody

    return res

```

Line_high

```

def line_high(xa,ya,xb,yb):
    res = [[xa,ya]]

    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

    for y in range(ya,yb):
        res.append([x,y])
        if(p > 0):
            x += xi
            p += twodxdy
        else:
            p += twodx

    return res

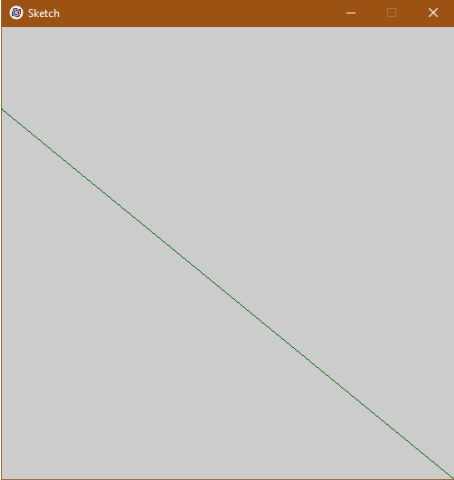
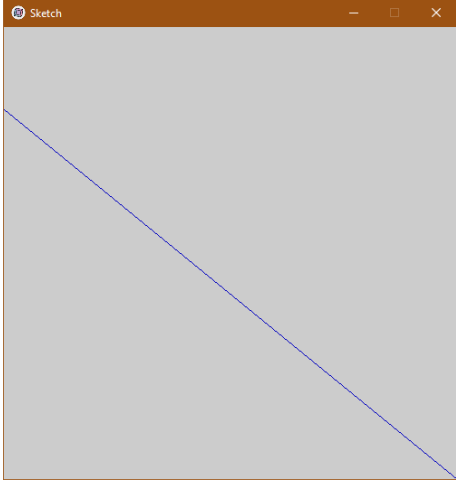
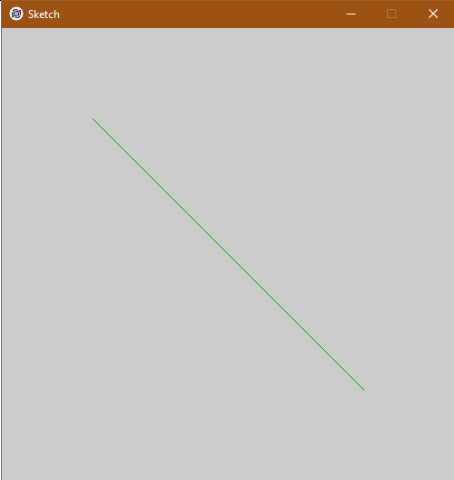
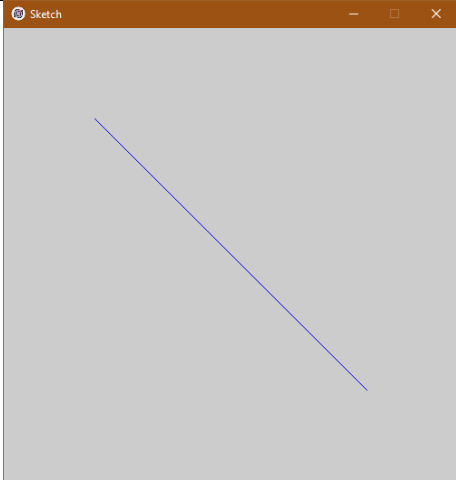
```

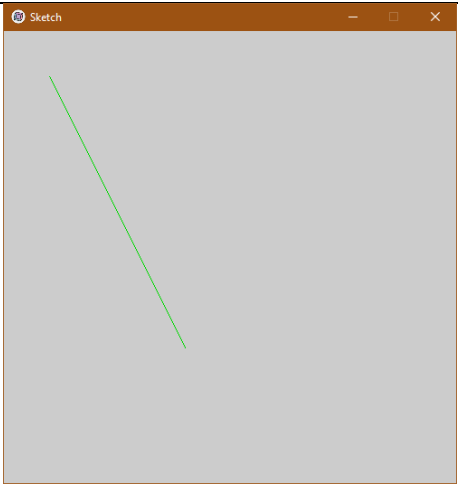
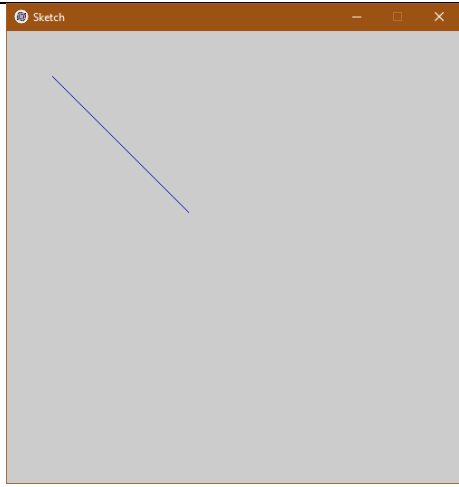
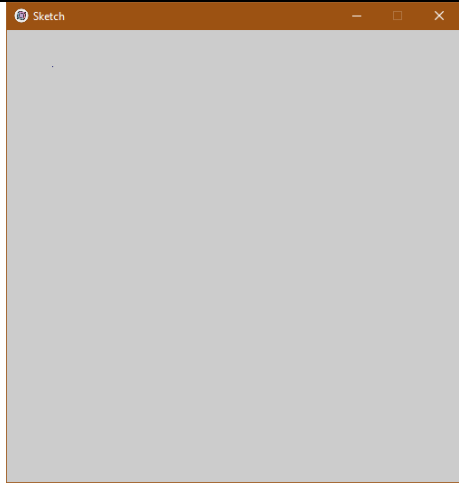
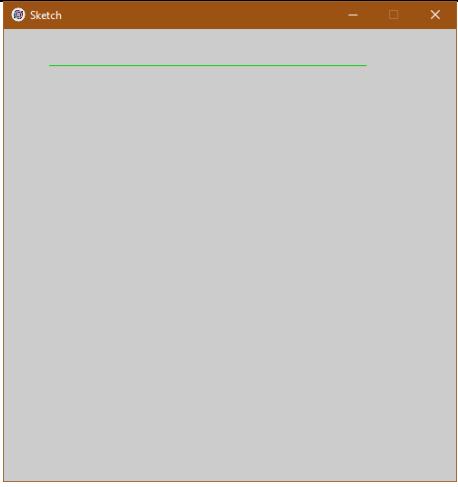
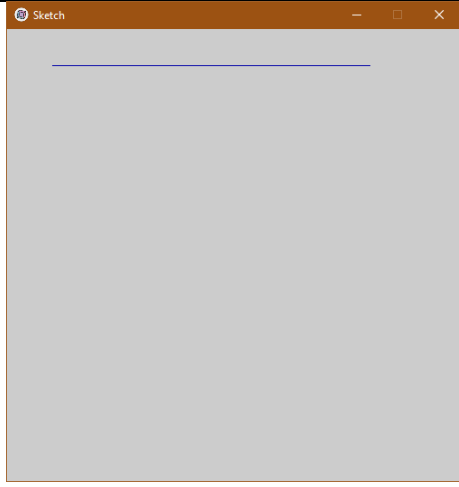
```
Contoh Pemanggilan Line DDA menggunakan Points (kumpulan titik)
py5.stroke(0,randint(0,255),0,255)
py5.points(
  line_bresenham(
    randint(0,py5.width),randint(0,py5.height),
    randint(0,py5.width),randint(0,py5.height)
  )
)
```

TASK PRAKTIKUM

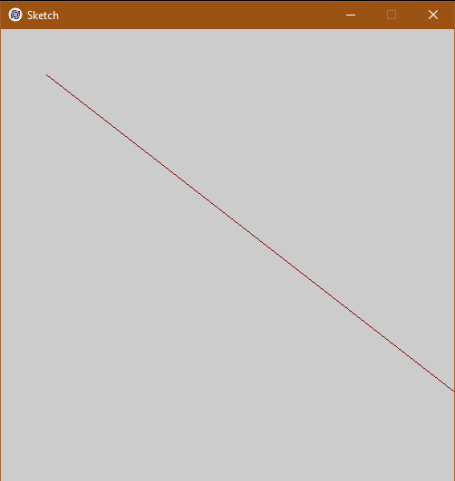
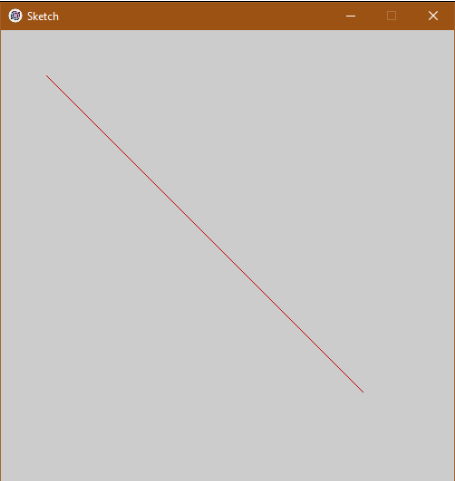
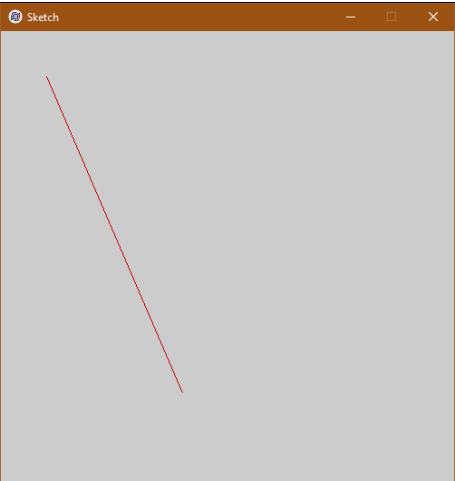
TASK 0-I: LINE DDA & LINE BERSENHAM

- 1. Buka Folder [KG2024_2X_001_D3_2022]_Modul2
- 2. Amati dan jalankan script tersebut
- 3. Buatlah Garis dengan Skenario Berikut

Skenario	Hasil & Analisa	
	Line DDA	Line Bersenham
$0 < m < 1$	<div></div>	<div></div>
	Hasilnya tampak sama	
$m = 1$	<div></div>	<div></div>
	Hasilnya tampak sama	

$m > 1$	<div data-bbox="451 94 906 577">  </div> <div data-bbox="1003 94 1458 577">  </div> <p>Hasilnya tampak berbeda</p>
$dx = 0$	<div data-bbox="451 625 906 1108">  </div> <div data-bbox="1003 625 1458 1108">  </div> <p>Hasilnya tampak berbeda, line DDA dapat menampilkan garis vertikal sedangkan line bersenham tidak dapat menampilkan</p>
$dy = 0$	<div data-bbox="451 1207 906 1690">  </div> <div data-bbox="1003 1207 1458 1690">  </div> <p>Hasilnya tampak sama</p>

TASK 2: GENERALISASI LINE BERSENHAM

Skenario	Hasil & Analisa
$0 < m < 1$	<div data-bbox="410 218 862 695"></div> <p>Tampak sama dengan bersenham sebelumnya</p>
$m = 1$	<div data-bbox="410 764 862 1241"></div> <p>Tampak sama dengan bersenham sebelumnya</p>
$m > 1$	<div data-bbox="410 1310 862 1787"></div> <p>Tampak berbeda dengan bersenham sebelumnya, namun bersenham versi 2 ini tampak sama dengan line dda sebelumnya, bersenham ini dapat menangani sudut yang curam</p>

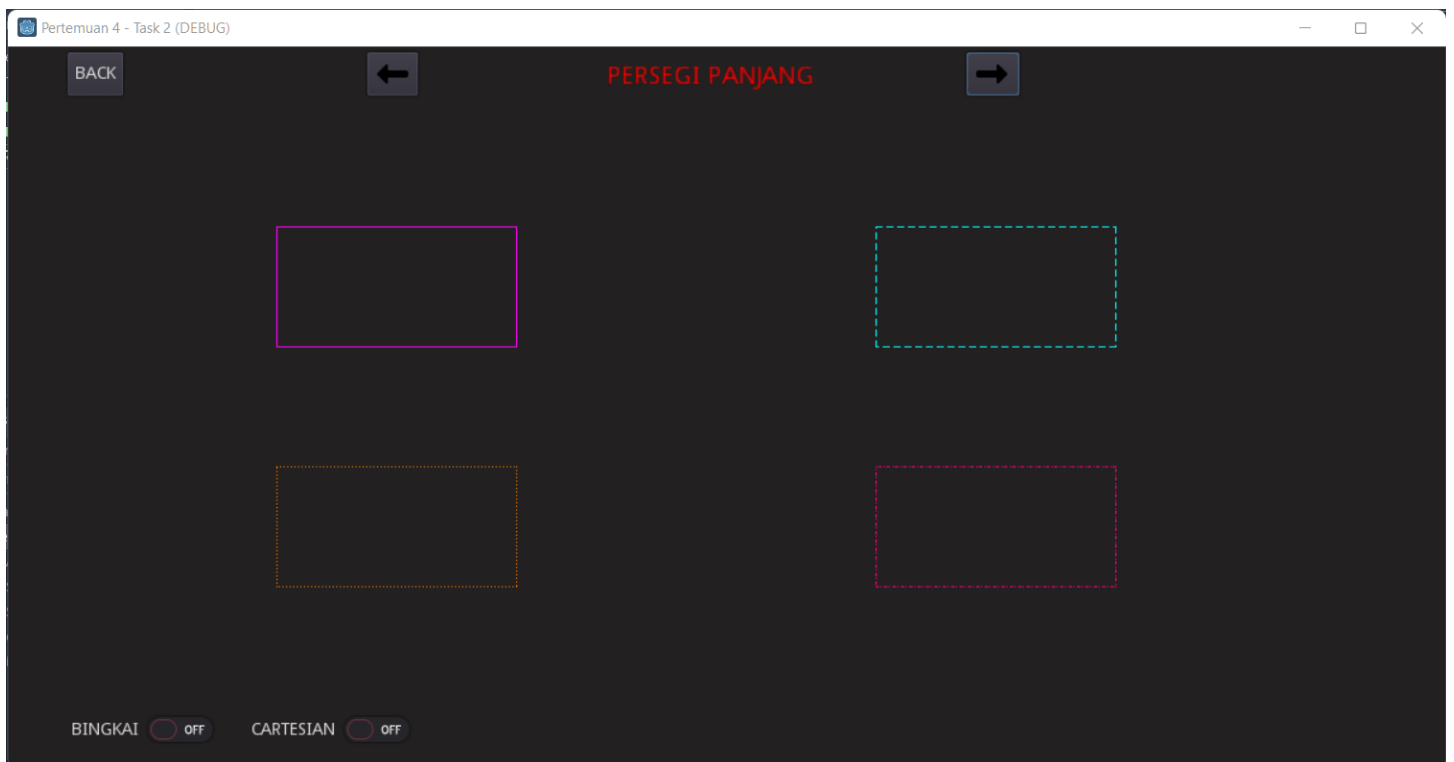
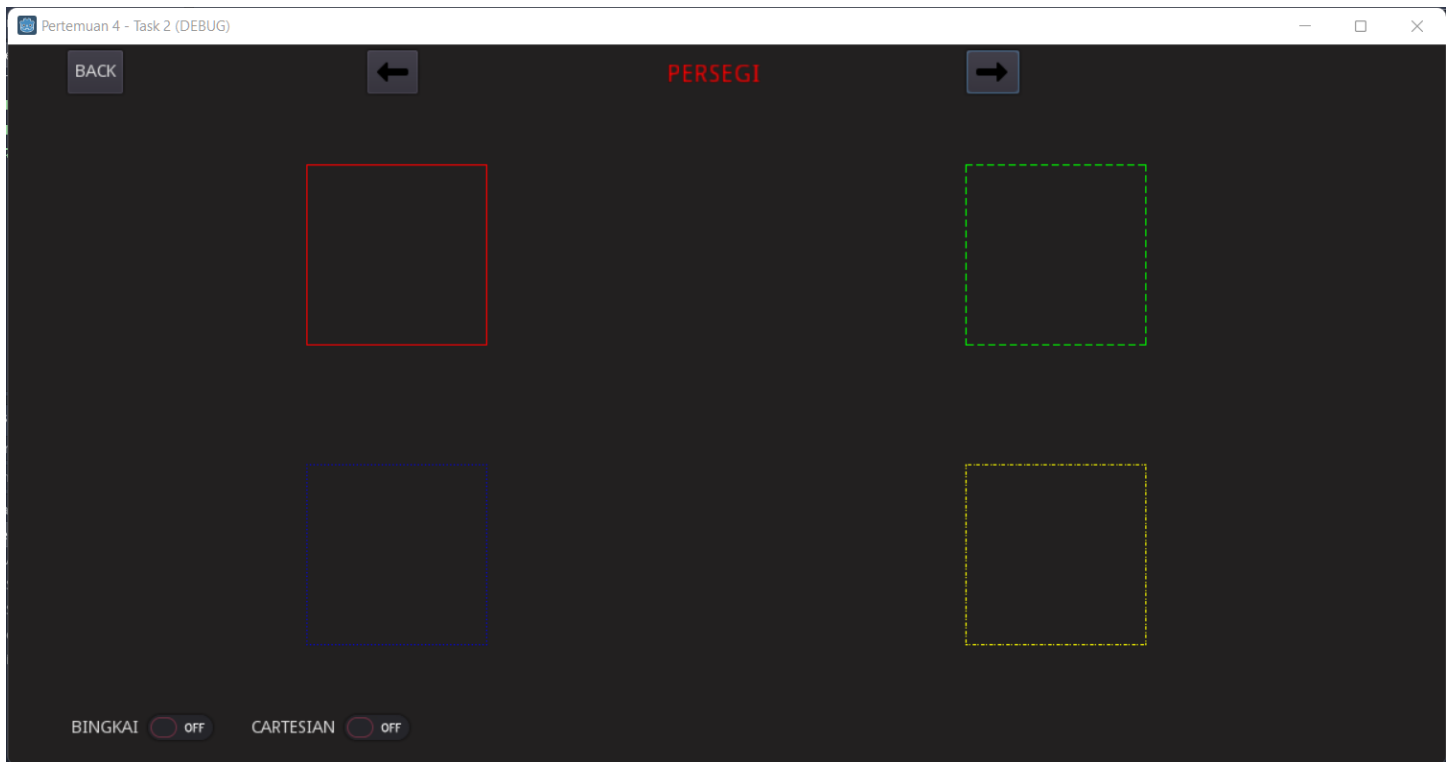
$dx = 0$	 <p>Tampak berbeda dengan bersenham sebelumnya, bersenham versi 2 ini dapat menampilkan garis lurus vertikal</p>
$dy = 0$	 <p>Tampak sama dengan bersenham sebelumnya</p>

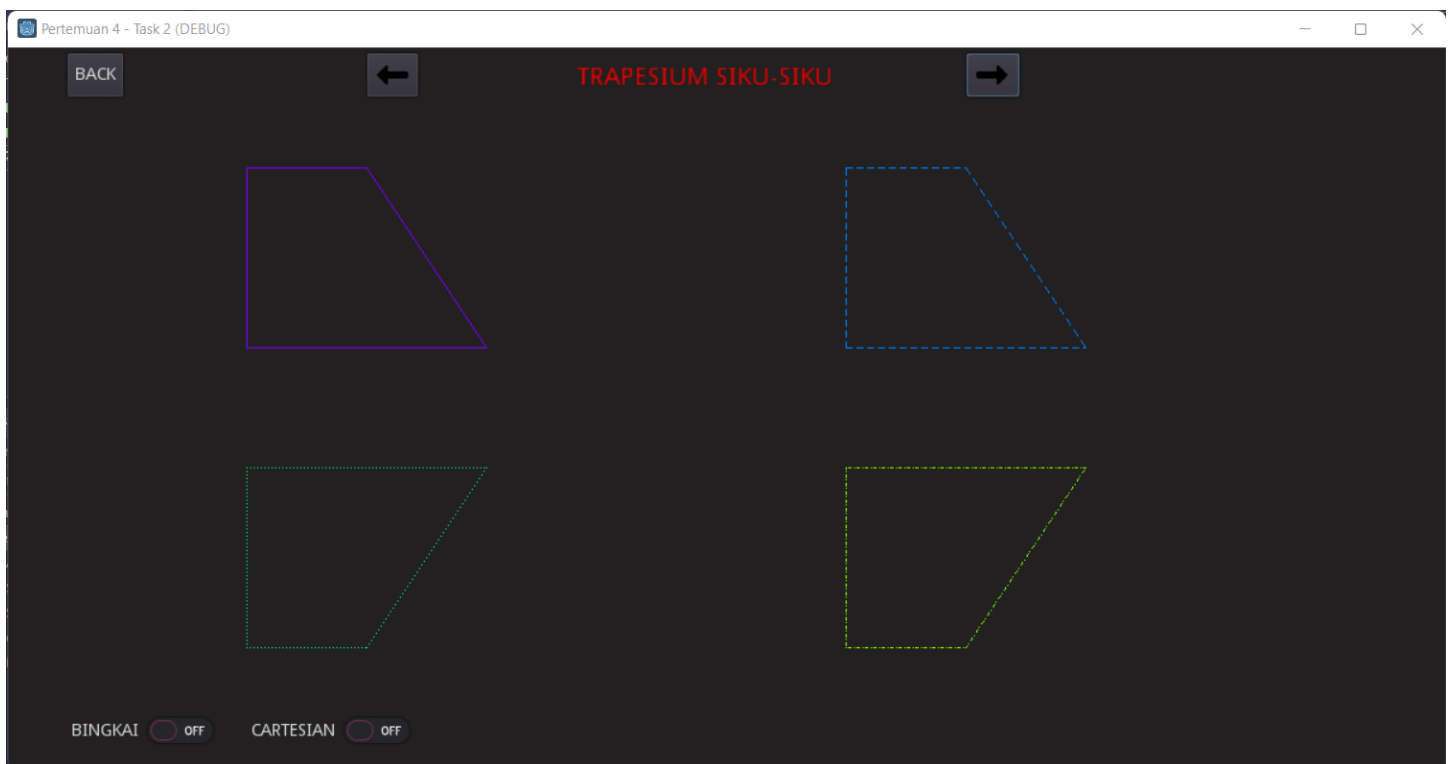
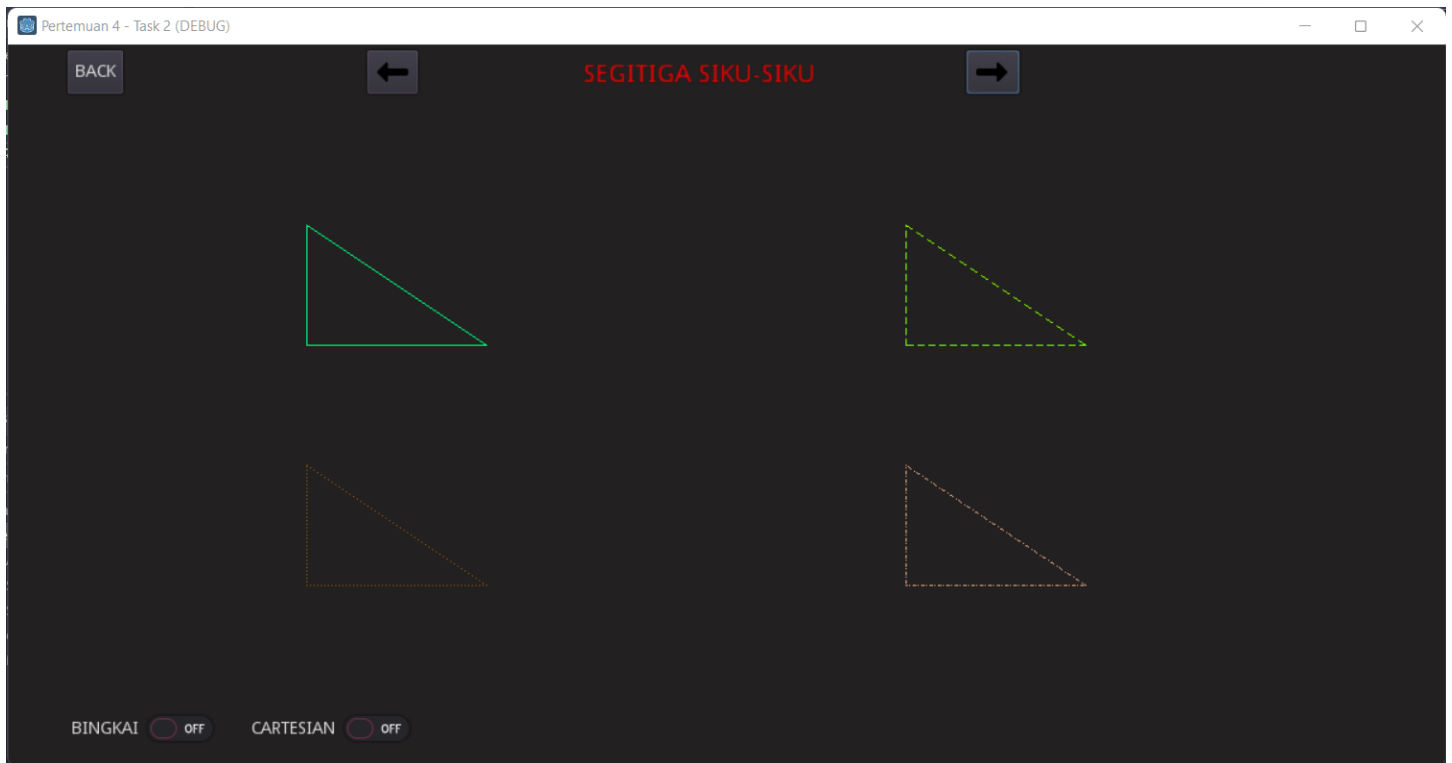
TASK 3: MEMBUAT FUNGSI BENTUK DASAR

1. Buatlah Fungsi-fungsi Bentuk Dasar menggunakan algoritma generalisasi line bersenham sbb: Persegi, Persegi Panjang, Segitiga Siku-Siku, dan Trapesium Siku-Siku
2. Posisikan Bentuk Dasar menjadi 4 Quadran.

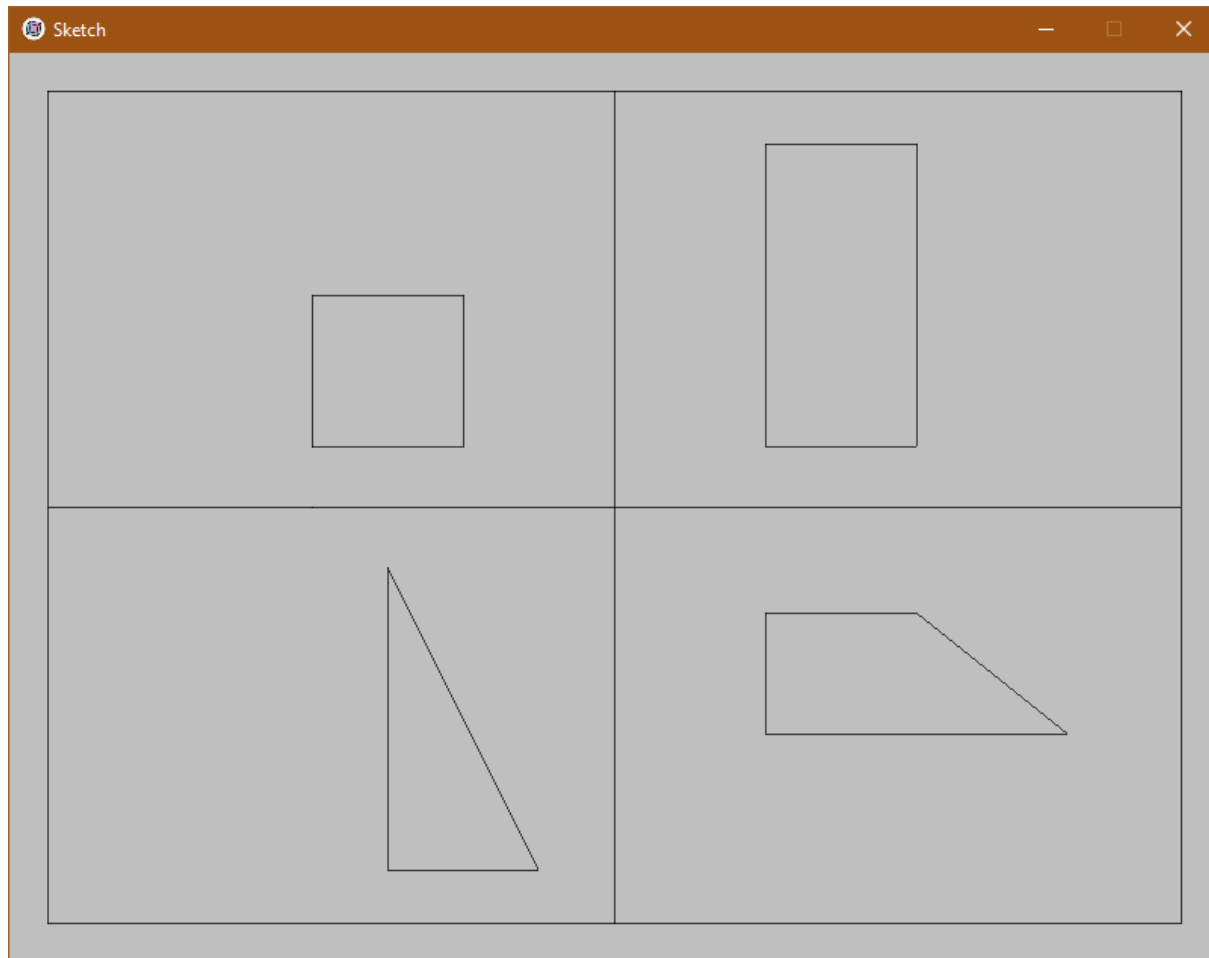
Untuk mengubah posisi dapat menggunakan fungsi convert to cartesian berikut

Utility.py
<pre>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]</pre>





Lesson Learnt (Print Screen Hasil Karya, dan Komentar)



Pada pengamatan penggambaran di line dda dan line bersenham terdapat beberapa perbedaan, dimana line bersenham tidak dapat menangani dan menggambarkan garis yang sudutnya lancip atau garis vertikal. Pada generalisasi line bersenham algoritmanya sudah disesuaikan dan dapat menangani sudut yang lancip dan bahkan garis vertikal

Pembuatan bentuk persegi, persegi panjang, segitiga dan trapesium sebagai penerapan dari praktikum ini, yang menggunakan line bersenham versi 2 atau yang algoritmanya sudah disesuaikan.

Dimana untuk format bentuk dasar disimpan sebagai function di file lain dan bisa digunakan kapan pun, ukuran dan posisinya bisa disesuaikan.

TASK 4: MEMBUAT KARYA 2D TIC TAC TOE

Buatlah board tic tac toe 3x3 dengan syarat sbb

1. Pemain 1 berupa X dan Pemain 2 Berupa Persegi
2. Letakan Pemain dengan Kondisi Pemain 1 Menang atau Letakan Pemain dengan Kondisi Pemain 2 Menang atau Letakan Pemain dengan Kondisi Tidak ada Pemenang

Lesson Learnt (Print Screen Hasil Karya, dan Komentar)

PENGUMPULAN

Ikuti Format yang diberikan di Google Classroom.