MODUL IV

KOMPUTER GRAFIK 2D
LINGKARAN DAN ELLIPS, TRANSFORMASI 2D

D3 TEKNIK INFORMATIKA
JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA
POLITEKNIK NEGERI BANDUNG

IRFANSYAH 089 | KOMPUTER GRAFIK | SEPTEMBER, 16 2023

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.

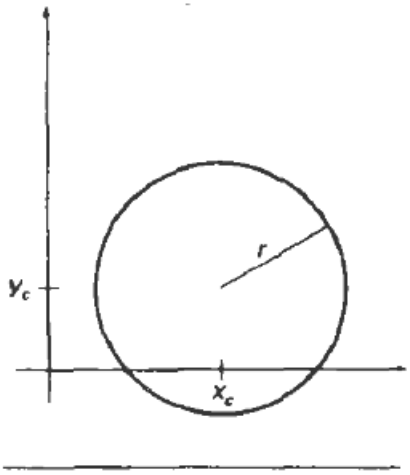


Figure 3-12
Circle with center coordinates (x_c, y_c) and radius r.

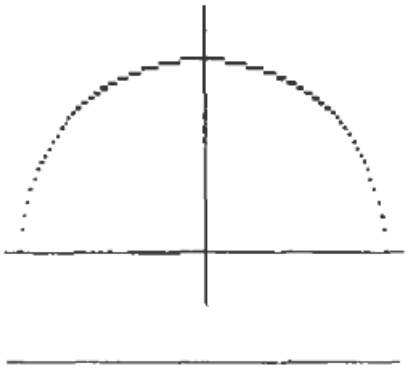


Figure 3-13
Positive half of a circle plotted with Eq. 3-25 and with $(x_c, y_c) = (0, 0)$.

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

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

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

Step size :

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

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

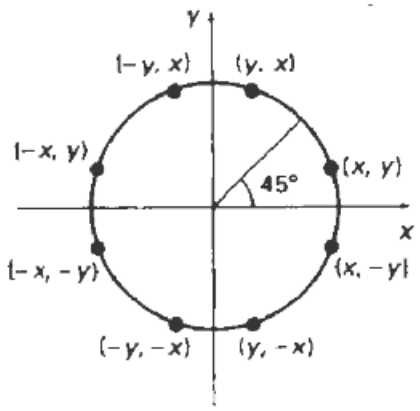


Figure 3-14
Symmetry of a circle. Calculation of a circle point (x, y) in one octant yields the circle points shown for the other seven octants.

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 $\mathrm{y_c}$ 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.

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

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

$$f\_circle(x, y) = \begin{cases} < 0, & if\ (x, y)\ is\ inside\ the\ circle\ boundary \\ = 0, & if\ (x, y)\ is\ on\ the\ circle\ boundary \\ > 0, & if\ (x, y)\ is\ outside\ the\ circle\ boundary \end{cases}$$

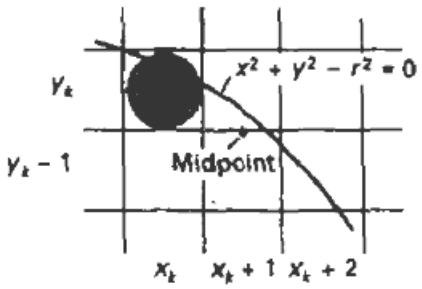


Figure 3-15
Midpoint between candidate pixels at sampling position x_k+1 along a circular path.

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

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

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

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

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

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

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

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

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

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

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 pada py5

Contoh Pemanggilan Lingkaran menggunakan Points (kumpulan titik)
py5.stroke(0,randint(0,255),0,255) py5.points( )

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.

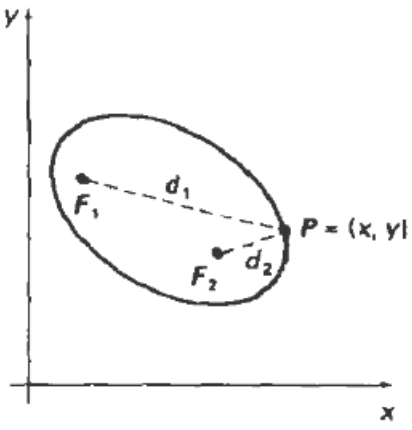


Figure 3-17
Ellipse generated about foci F_1 and F_2.

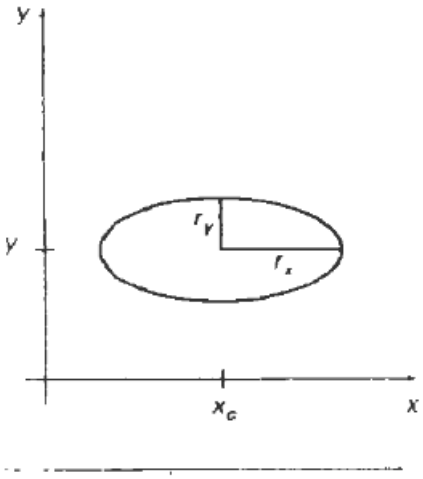


Figure 3-18
Ellipse centered at (x_c, y_c) with semimajor axis r_x and semiminor axis r_y.

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.

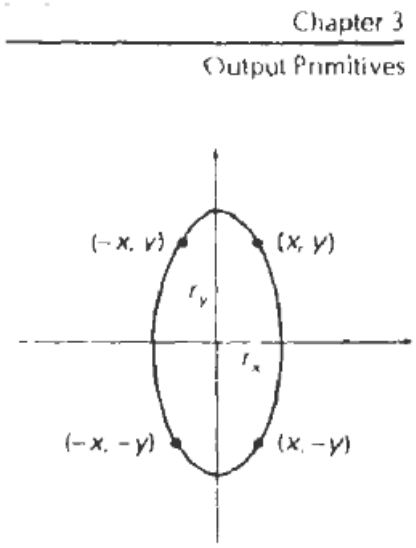


Figure 3-19
Symmetry of an ellipse Calculation of a point (x, y) in one quadrant yields the ellipse points shown for the other three quadrants.

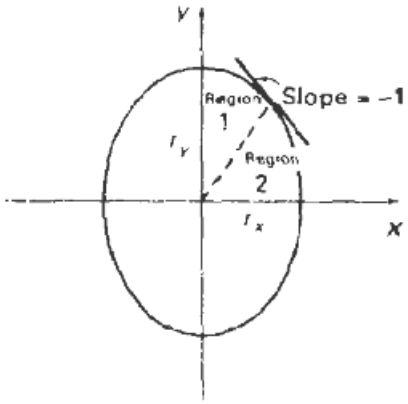


Figure 3-20
Ellipse processing regions. Over region 1, the magnitude of the ellipse slope is less than 1; over region 2, the magnitude of the slope is greater than 1.

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:

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

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

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

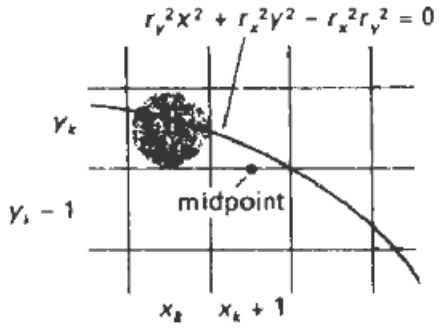


Figure 3-21
Midpoint between candidate pixels at sampling position $x_k + 1$ along an elliptical path.

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

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

$$y_{k+1} = \begin{cases} jika\ p1_k < 0, y_k \\ jika\ p1_k > 0, y_k - 1 \end{cases}$$

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

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

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:

$$incerement\ = \begin{cases} 2r_y^2 x_{k+1} + r_y^2, jika\ p1_k < 0 \\ 2r_y^2 x_{k+1} + r_y^2 - 2r_x^2 y_{k+1}, jika\ p1_k > 0 \end{cases}$$

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

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

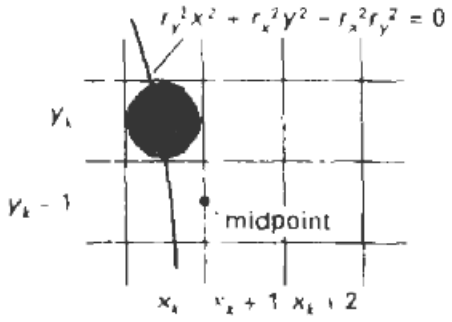


Figure 3-22
Midpoint between candidate pixels at sampling position $y_k - 1$ along an elliptical path.

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:

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

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

$$\mathrm{x_{k+1}} = \begin{cases} jika\ p2_k > 0, x_k \\ jika\ p2_k \leq\ 0, x_{k+1} \end{cases}$$

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

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

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 py5

Contoh Pemanggilan Ellips menggunakan Points (kumpulan titik)
py5.stroke(0,randint(0,255),0,255) py5.points( )

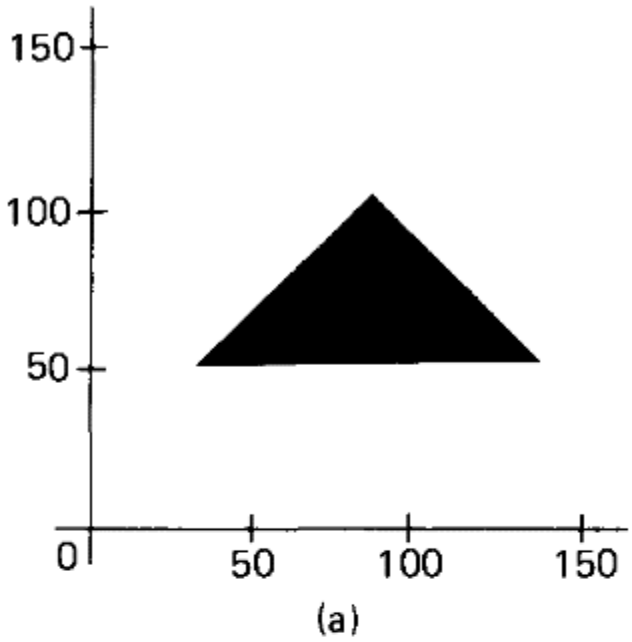
TRANSLASI 2D

A translation is a straight-line movement of an object from one position to another. We translate a point from coordinate position (x, y) to a new position (x', y') by adding translation distances, Tx and Ty, to the original coordinates:

$$x' = x + Tx, \qquad y' = y + Ty \tag{5--1}$$

Polygons are translated by adding the specified translation distances to the coordinates of each line endpoint in the object. Figure 5—1 illustrates movement of a polygon to a new position as determined by the translation vector (20, 50). Objects drawn with curves are translated by changing the defining coordinates of the object. To change the position of a circle or ellipse, we translate the center coordinates and redraw the figure in the new location.

Translation distances can be specified as any real numbers (positive, negative, or zero). .



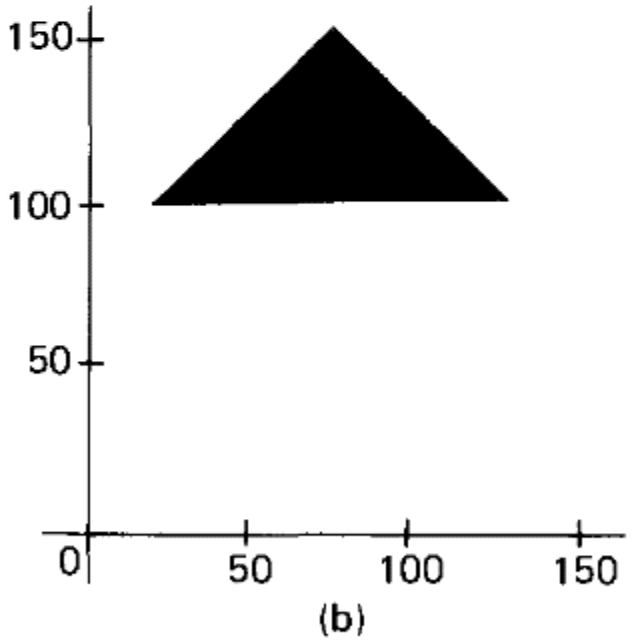


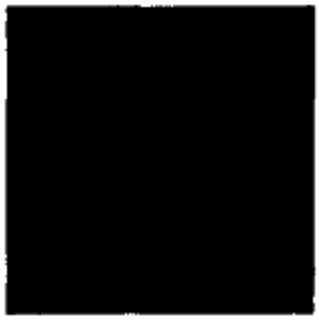
FIGURE 5–1
Translation of an object from position (a) to position (b) with translation distances (−20, 50).

SCALING 2D

A transformation to alter the size of an object is called scaling. This operation can be carried out for polygons by multiplying the coordinate values (x, y) for each boundary vertex by scaling factors Sx and Sy to produce transformed coordinates.

$$x' = x \cdot Sx, \qquad y' = y \cdot Sy \tag{5-2}$$

Scaling factor Sx scales objects in the x direction, while Sy scales in the y direction. Any positive numeric values can be assigned to the scaling factors Sx and Sy. Values less than 1 reduce the size of objects; values greater than 1 produce anenlargement. Specifying a value of 1 for both Sx and Sy leaves the size of objects unchanged. When Sx and Sy are assigned the same value, a uniform scaling is produced, which maintains relative proportions of the scaled object.



(a)

(b)

FIGURE 5–3
Turning a square (a) into a rectangle (b) by setting $S_x = 2$ and $S_y = 1$.

ROTATE 2D

Transformation of object points along circular paths is called rotation. We specify this type of transformation with a rotation angle, which determines the amount of rotation for each vertex of a polygon.

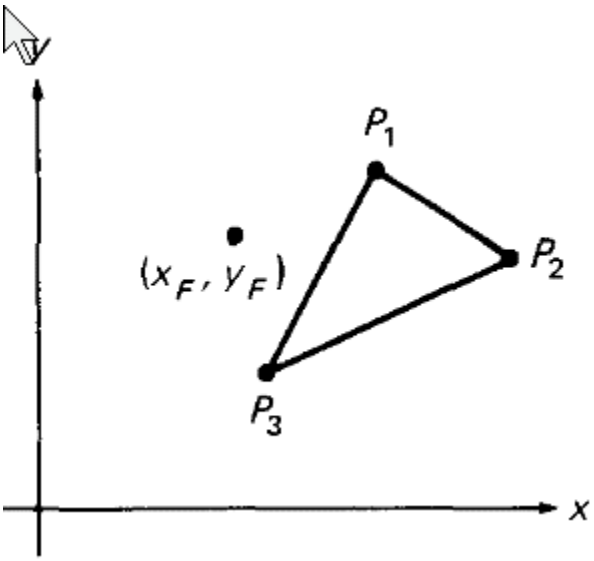


Figure 5-6 illustrates displacement of a point from position (x, y) to position (x', y'), as determined by a specified rotation angle 0 relative to the coordinate origin. In this figure, angle is the original angular position of the point from the horizontal. We can determine the transformation equations for rotation of the point from the relationships between the sides of the right triangles shown and the

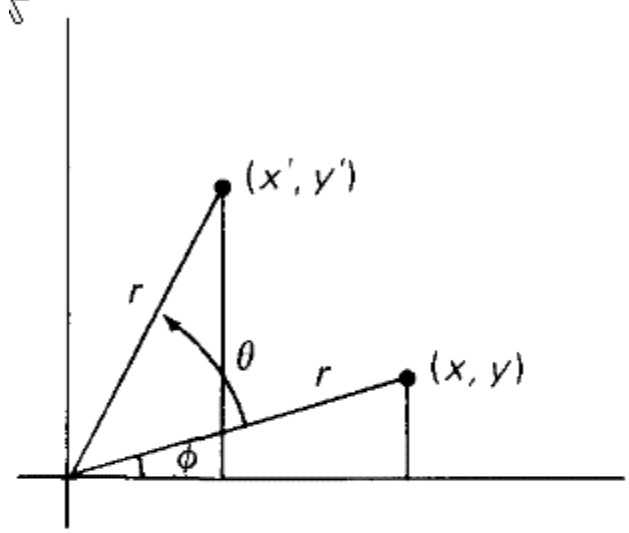


FIGURE 5–6
Rotation of a point from position *(x, y)* to position *(x', y')* through a rotation angle θ, specified relative to the coordinate origin. The original angular position of the point from the *x* axis is φ.

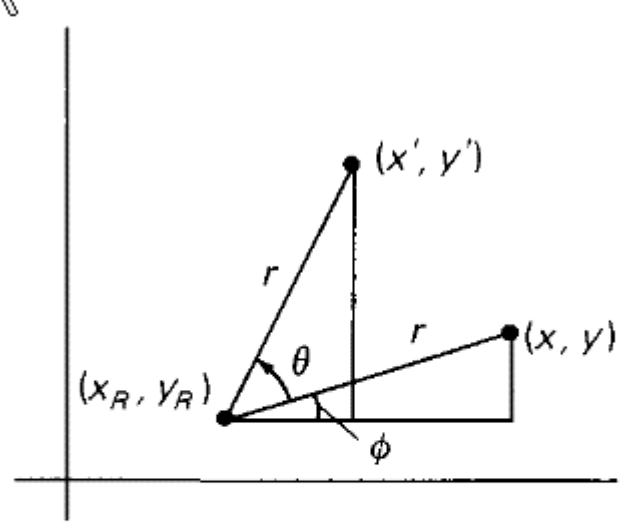


FIGURE 5–7
Rotation of a point from *(x, y)* to *(x', y')* through an angle θ, specified relative to a pivot point at (x_R, y_R).

MATRIX REPRESENTATIONS AND HOMOGENEOUS COORDINATES

There are many applications that make use of the basic transformations in various combinations. A picture, built up from a set of defined shapes, typically requires each shape to be scaled, rotated, and translated to fit into the proper picture position. This sequence of transformations could be carried out one step at a time. First, the coordinates defining the object could be scaled, then these scaled coordinates could be rotated, and finally the rotated coordinates could be translated to the required location. A more efficient approach is to calculate the final coordinates directly from the initial coordinates using matrix methods, with each of the basic transformations expressed in matrix form.

We can write transformation equations in a consistent matrix form by first expressing points as **homogeneous coordinates.** This means that we represent a two-dimensional coordinate position (x, y) as the triple $[x_h\ y_h\ w]$, where

$$x_h = x \cdot w, \qquad y_h = y \cdot w \tag{5–9}$$

$$[x'\ y'\ 1] = [x\ y\ 1]\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx & Ty & 1 \end{bmatrix} \tag{5–10}$$

We also introduce the abbreviated notation $T(Tx, Ty)$ for the 3 by 3 transformation matrix with translation distances Tx and Ty

$$T(Tx, Ty) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx & Ty & 1 \end{bmatrix} \tag{5–11}$$

Using this notation, we can write the matrix form for the translation equations more compactly as

$$P' = P \cdot T(Tx, Ty) \tag{5–12}$$

where $P' = [x'\ y'\ 1]$ and $P = [x\ y\ 1]$ are 1 by 3 matrices (three-element row vectors) in the matrix calculations.

Similarly, the scaling equations 5–2 are now written as

$$[x'\ y'\ 1] = [x\ y\ 1]\begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5–13}$$

with

$$S(Sx,\ Sy) = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5–15}$$

as the 3 by 3 transformation matrix for scaling with parameters Sx and Sy.

Equations 5–7, for rotation, are written in matrix form as

$$[x'\ y'\ 1] = [x\ y\ 1]\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5–16}$$

or as

$$P' = P \cdot R(\theta) \tag{5–17}$$

where

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5–18}$$

is the 3 by 3 transformation matrix for rotation with parameter θ.

Algoritma Translasi, Scaling dan Rotasi dengan Homogenous Coordinates.

```
#include <math.h>
#include "graphics.h"

typedef float Matrix3x3[3][3];
Matrix3x3 theMatrix;

void matrix3x3SetIdentity (Matrix3x3 m)
{
  int i,j;

  for (i=0; i<3; i++) for (j=0; j<3; j++) m[i][j] = (i == j);
}

/* Multiplies matrix a times b, putting result in b */
void matrix3x3PreMultiply (Matrix3x3 a, Matrix3x3 b)
{
  int r,c;
  Matrix3x3 tmp;

  for (r = 0; r < 3; r++)
    for (c = 0; c < 3; c++)
      tmp[r][c] =
        a[r][0]*b[0][c] + a[r][1]*b[1][c] + a[r][2]*b[2][c];

  for (r = 0; r < 3; r++)
    for (c = 0; c < 3; c++)
      b[r][c] = tmp[r][c];
}

void translate2 (int tx, int ty)
{
  Matrix3x3 m;

  matrix3x3SetIdentity (m);
  m[0][2] = tx;
  m[1][2] = ty;
  matrix3x3PreMultiply (m, theMatrix);
```

```
}

void scale2 (float sx, float sy, wcPt2 refpt)
{
  Matrix3x3 m;

  matrix3x3SetIdentity (m);
  m[0][0] = sx;
  m[0][2] = (1 - sx) * refpt.x;
  m[1][1] = sy;
  m[1][2] = (1 - sy) * refpt.y;
  matrix3x3PreMultiply (m, theMatrix);
}

void rotate2 (float a, wcPt2 refPt)
{
  Matrix3x3 m;

  matrix3x3SetIdentity (m);
  a = pToRadians (a);
  m[0][0] = cosf (a);
  m[0][1] = -sinf (a);
  m[0][2] = refPt.x * (1 - cosf (a)) + refPt.y * sinf (a);
  m[1][0] = sinf (a);
  m[1][1] = cosf (a);
  m[1][2] = refPt.y * (1 - cosf (a)) - refPt.x * sinf (a);
  matrix3x3PreMultiply (m, theMatrix);
}

void transformPoints2 (int npts, wcPt2 *pts)
{
  int k;
  float tmp;

  for (k = 0; k < npts; k++) {
    tmp = theMatrix[0][0] * pts[k].x + theMatrix[0][1] *
      pts[k].y + theMatrix[0][2];
    pts[k].y = theMatrix[1][0] * pts[k].x + theMatrix[1][1] *
      pts[k].y + theMatrix[1][2];
    pts[k].x = tmp;
  }
}
```

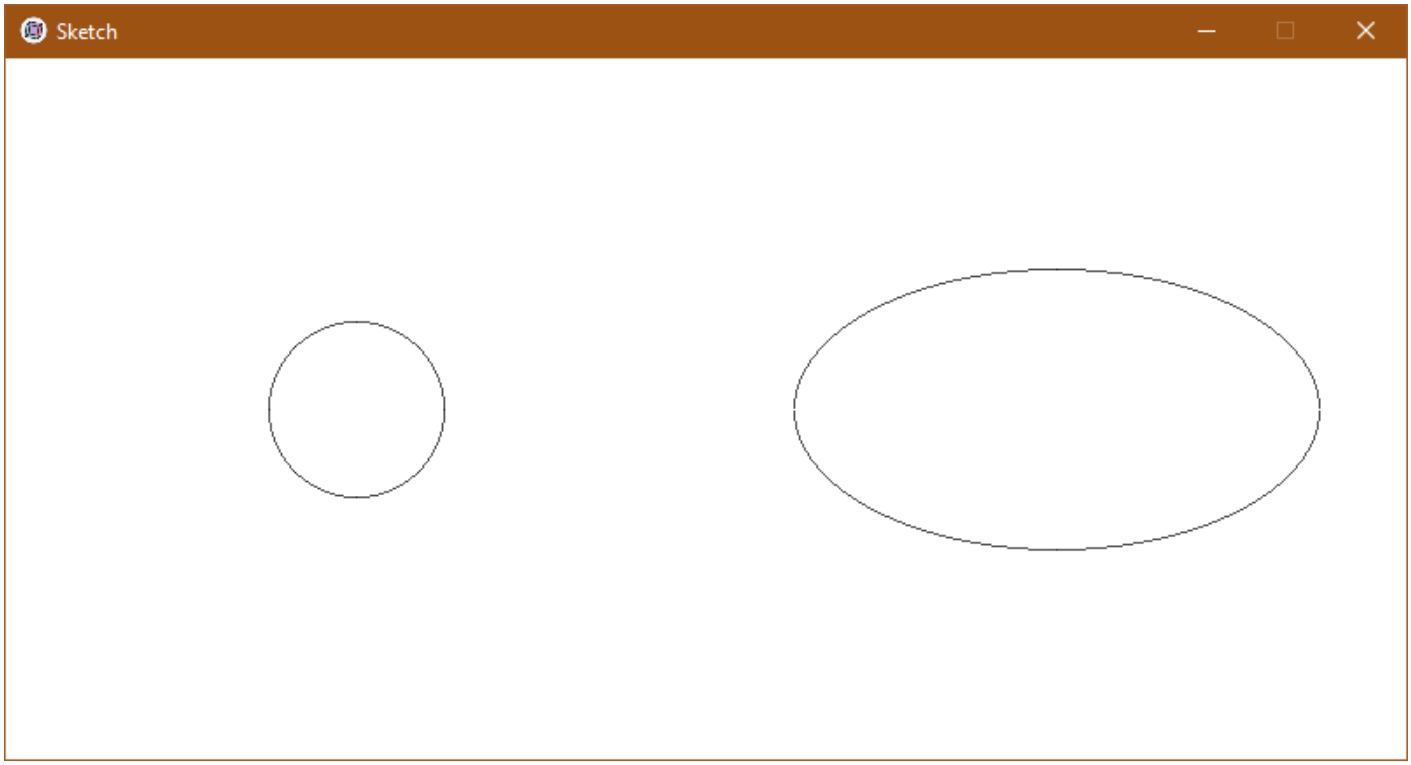
```
void main (int argc, char ** argv)
{
  wcPt2 pts[3] = { 50.0, 50.0, 150.0, 50.0, 100.0, 150.0};
  wcPt2 refPt = {100.0, 100.0};
  long windowID = openGraphics (*argv, 200, 350);

  setBackground (WHITE);
  setColor (BLUE);
  pFillArea (3, pts);
  matrix3x3SetIdentity (theMatrix);
  scale2 (0.5, 0.5, refPt);
  rotate2 (90.0, refPt);
  translate2 (0, 150);
  transformPoints2 (3, pts),
  pFillArea (3,pts);
  sleep (10);
  closeGraphics (windowID);
}
```

TASK PRAKTIKUM

TASK 1: KARYA 2D BUNGA KELOPAK

1. Lanjutkan kode minggu lalu [KG2024_2X_001_D3_2023]_Modul4
2. Amati Algoritma Midpoint untuk menggambar lingkaran dan ellips
3. Implementasi Algoritma tersebut dan buatlah fungsi lingkaran / circle dan ellips / ellipse pada primitif/basic.py

Lesson Learnt (Print Screen Hasil Karya, dan Komentar)



Pada pembuatan lingkaran, hanya diperlukan titik tengah dan radius yang mengelilingi titik tengah tersebut.

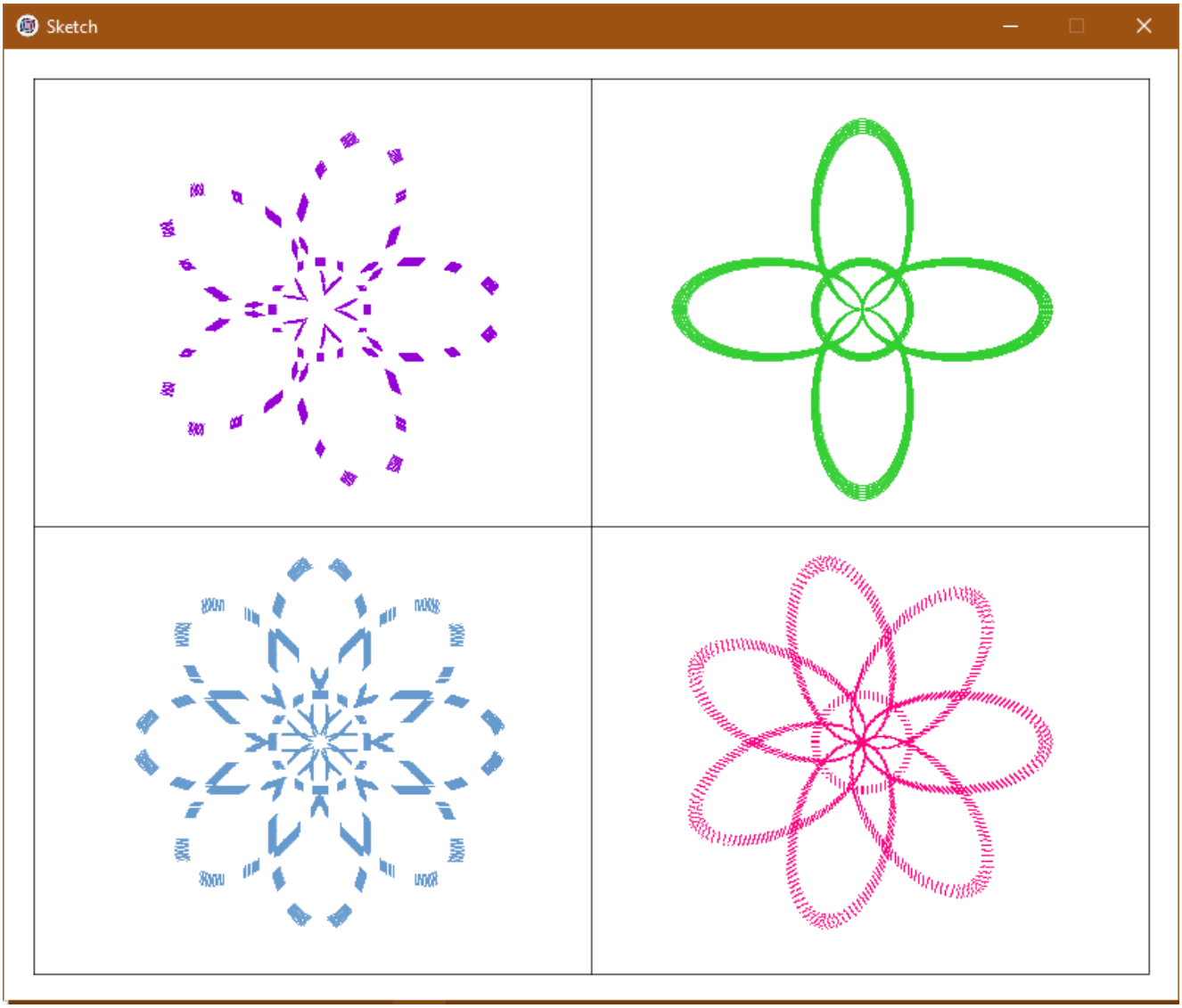
Lingkaran memiliki 8 segmen simetri, yang berarti cukup menggambarkan 1/8 nya dan sisanya dicerminkan dari gambar tsb.

Sedangkan dalam pembuatan ellip diperlukan titik tengah dan kedua radius yang berbeda (satu sumbu horizontal dan satu vertikal). Ellips memiliki 4 segmen simetri.

TASK 2: MEMBUAT KARYA 2D SEDERHANA

1. Buatlah Kelopak Bunga dari Bentuk Dasar (pusat berupa lingkaran) dan kelopak berupa ellips
2. Posisikan Bentuk Dasar menjadi 4 Quadran. Quadran 1 bunga 4 kelopak, Quadran 2 bunga 5 kelopak, Quadran 3 bunga 7 kelopak, dan kuadran 4 bunga 8 kelopak.
3. Gunakan Konsep OOP untuk membuat karya tersebut dan tambahkan attribute garis untuk setiap quadran yang berbeda-beda.

Lesson Learnt (Print Screen Hasil Karya, dan Komentar)



Dalam pembuatan objek bunga ini, saya mengguakan konsep OOP, dimana terdapat sebuah class bernama Bunga.

Class bunga ini mempunyai parameter/ argumen titik tengah, radius_lingkaran, jumlah kelopak, color, dan pattern.

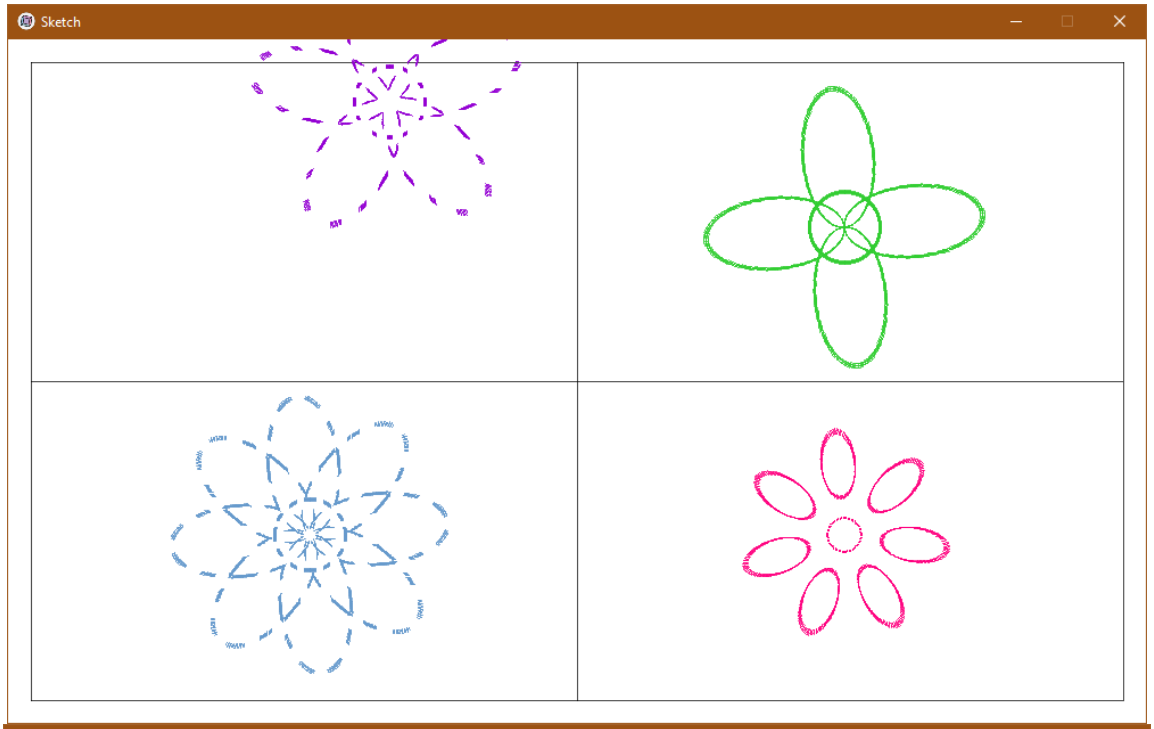
Di dalam class bunga sendiri terdapat fungsi **gambar_bunga** yang memanggil fungsi untuk **menggambar lingkaran** dan **menggambar kelopak.**

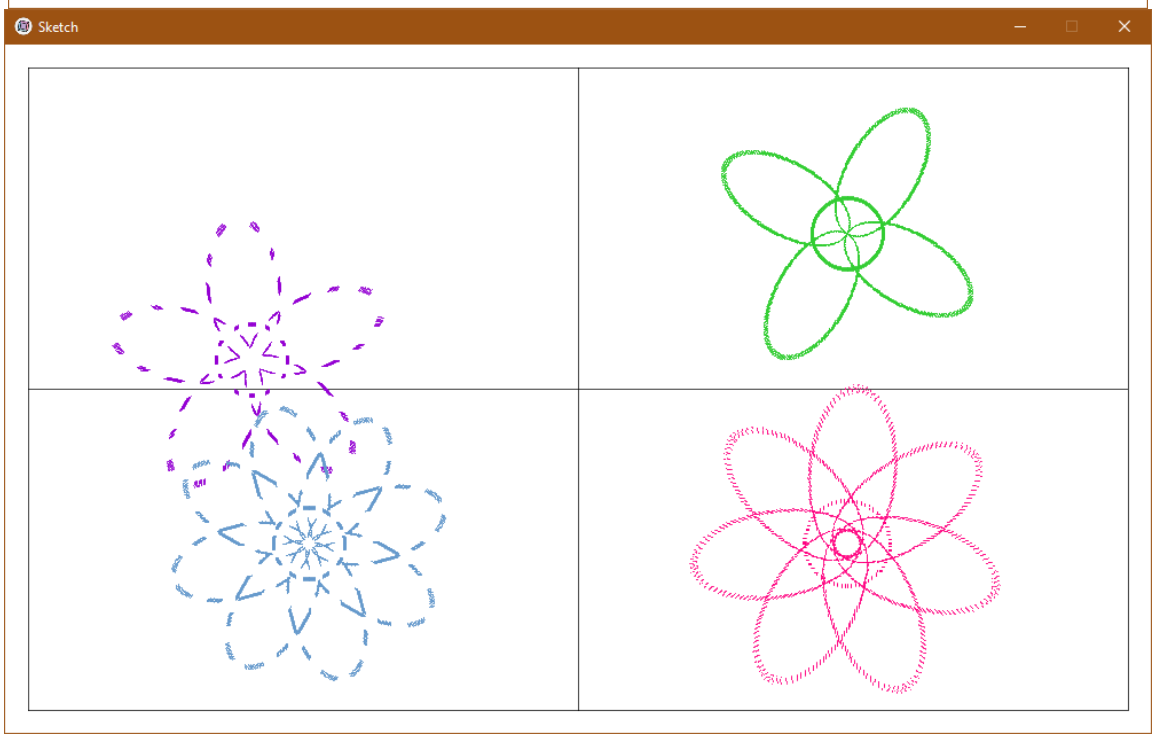
Pada implementasinya saya hanya perlu membuat objek dengan argumen. Kemudian memanggil objek tsb dengan method menggambar bunga

TASK 3 MEMBUAT FUNGSI-FUNGSI TRANSFORMASI 2D,

1. Bunga pada Task 2, dimanipulasi dengan Transformasi 2D
2. Buatlah animasi bunga tersebut Translasi, Scaling, Rotasi dan Komposit.
3. Gunakan fungsi transformasi yang sudah disediakan di code

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





Pada pembuatan transformasi ini saya harus mengetahui letak titik titik yang akan digambarkan, yang sebelumnya di gambarkan di function nya langsung, yaitu basic.py. dalam membuat transformasi titik titik tersebut harus ditransformasi dulu lalu digambarkan.
Jadi pada function basic.py saya tidak menggambarkan garisnya, tetapi mereturn value titik titik yang akan digambarkan. Sehingga titik titik ini dapat dilakukan transformasi.

PENGUMPULAN

Ikuti Format yang diberikan di Google Classroom.