ASSIGNMENT 7

COMPUTER GRAPHICS

AUTHORS:

LUCÍA SÁNCHEZ-MONTES GÓMEZ

TERESA DEL CARMEN CHECA MARABOTTO

Exercise 1 [7 points]

Let the camera opening angle be $\frac{3}{4}\pi$ radians and the window be 15×15 pixels large. Which pixels does the midpoint algorithm (without anti-aliasing) set for the line from $p_1 = (1, 1, 4)$ to $p_2 = (2, 1, 1)$, given in global coordinates? And which pixels are set if we replace p_2 by $p_2' = (3, 1, -2)$? Is that correct, and if not, then which pixels should be set instead?

EXERCISE 1

Camera angle $\rightarrow \frac{3}{4}\pi$ rad

Window $\rightarrow 15 \times 15$ pixels

midpoint algorithm for the line from $p_1 = (1,1,4)$ to $p_2 = (2,1,1)$:

In order to solve this exercise we need to understand how to convert the global coordinates to the pixel view.

First of all we will change the point p_1 and p_2 from 3D to 2D:

$$p_1 = (1,1,4) \longrightarrow (1/4, 1/4, 1) \rightarrow (1/4, 1/4)$$

$$p_2 = (2,1,1) \longrightarrow (2,1,1) \longrightarrow (2,1)$$

Now, our points are $p_1' = (1/4, 1/4)$ and $p_2' = (2,1)$.

Then, since we know that the formulas to obtain the pixels values are:

$$x = X + x_1 \cdot s + s/2$$

$$y = Y + y_1 \cdot s + s/2$$

We will solve that formula.

We know that:

$$s = \frac{2 \cdot \tan(0.5 \cdot fov)}{size} = \frac{2 \cdot \tan(0.5 \cdot \frac{3}{4}\pi)}{15} = 0.3219$$

Also that:

$$X = -\frac{s * size}{2} = \frac{-0.3219 \times 15}{2} = -2.415$$

$$Y = \frac{s * size}{2} = \frac{0.3219 * 15}{2} = 2.415$$

For the first point $p_1' = (1/4, 1/4)$

$$x = X + x_1 \cdot s + s/2 ;$$

$$\frac{x - X - s/2}{s} = x_1 ; \quad x_1 = \frac{1/4 + 2.415 - \frac{0.3219}{2}}{0.3219} = 7.779$$

$$y = Y \pm y_1 \cdot s - s/2$$

$$\frac{y - Y + \frac{s}{2}}{-s} = y_1 ; \quad y_1 = \frac{1/4 - 2.415 + \frac{0.3219}{2}}{-0.3219} = 6.226$$

We obtain: $\begin{cases} x_1 = 7.779 \approx 8 \\ x_2 = 6.226 \approx 6 \end{cases}$

For the secon point $p_2' = (2, 1)$

$$x = X + x_2 \cdot s + s/2 ;$$

$$x_2 = \frac{x - X - s/2}{s} = \frac{2 + 2.415 - \frac{0.3219}{2}}{0.3219} = 13.215$$

$$y = Y - y_2 \cdot s - s/2$$

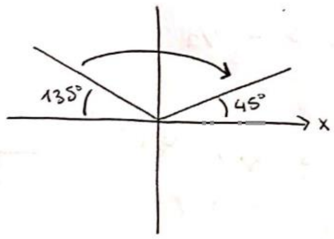
$$y_2 = \frac{y - Y + \frac{s}{2}}{-s} = \frac{1 - 2.415 + \frac{0.3219}{2}}{-0.3219} = 3.896$$

We obtain: $\begin{cases} x_2 = 13'215 \approx 13 \\ y_2 = 3'896 \approx 4 \end{cases}$

Now, we need to check the slope:

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)} = \frac{4-6}{13-8} = -0.4$$

m isn't between 0 and 1, because our angle is 135° (3/4π rad), so we need to change it to 45° which means, change the x componen to negative:



So now, we have:

$x_1 = -8$ $\quad$ $x_2 = -13$

$y_1 = 6$ $\quad$ $y_2 = 4$

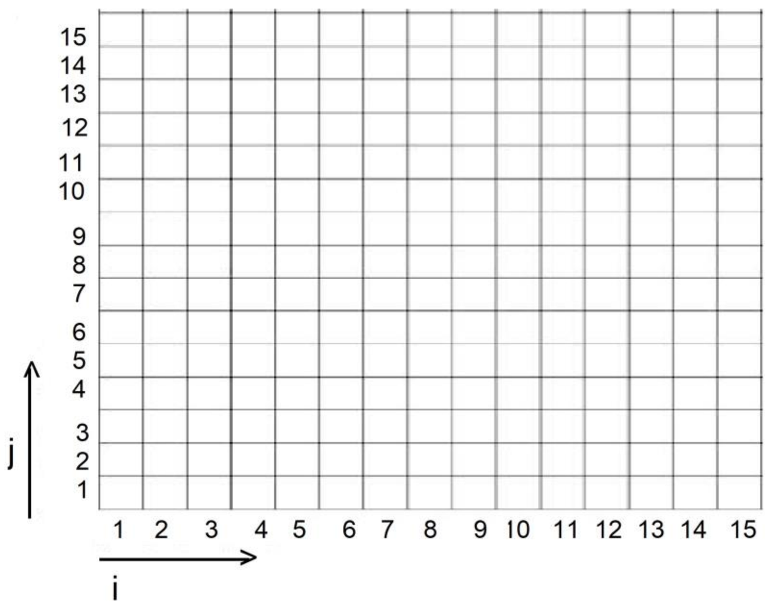
And the slope is:

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)} = \frac{4-6}{-13+8} = 0.4$$

now m is between 0 and 1, but the condition $x_1 < x_2$ do not pass (because $-8 \nless -13$).

To solve that, we need to assume that our initial point for the pixels is flipped.

So the new representation will be:



So now we can compute the midpoint algorithm.

We need to follow the next algorithm:

$x = x_1$; $y = y_1$

$dx = x2 - x1$; $dy = y2 - y1$

$f = -2 \cdot dy + dx$

for $i=0$ to dx do

setpixel (x, y)

$x+1$

if $(f<0)$

$y+=1$

$f+=2 \cdot dx$

$f -= 2 \cdot dy$

We know that:

$x = x_1 = -8$

$y = y_1 = 6$

$dx = -13 - (-8) = -5$

$dy = 4 - 6 = -2$

$f = -2 \cdot (-2) - 5 = -1$

As the pseudocode said, we need to compute the next algorithm 6 times (for $i=0$ to -5).

1) setpixel$(-8,6)$

$x+1$; $x = -7$

if $(-1 < 0)$

$y+1 = 7$

$f = -1 + 2(-5) = -11$

$f = -11 - 2(-2) = -7$

2) setpixel (-7, 7)

x+=1 => x = -6

if (-7 < 0)

y = 8

f = -17

f = -17 + 4 = -13

3) setpixel (-6, 8)

x = -5

if (-13 < 0)

y = 9

f = -23

f = -19

4) setpixel (-5, 9)

x = -4

if(-19 < 0)

y = 10

f = -29

f = -25

5) setpixel (-4, 10)

x = -3

if (-25 < 0)

y = 11

f = -35

f = -31

6) setpixel (-3, 11)

x = -2

if (-31 < 0)

y = 12

f = - 41

f = -38

So we obtain the next set of points:

(-8, 6) (-7, 7) (-6, 8)

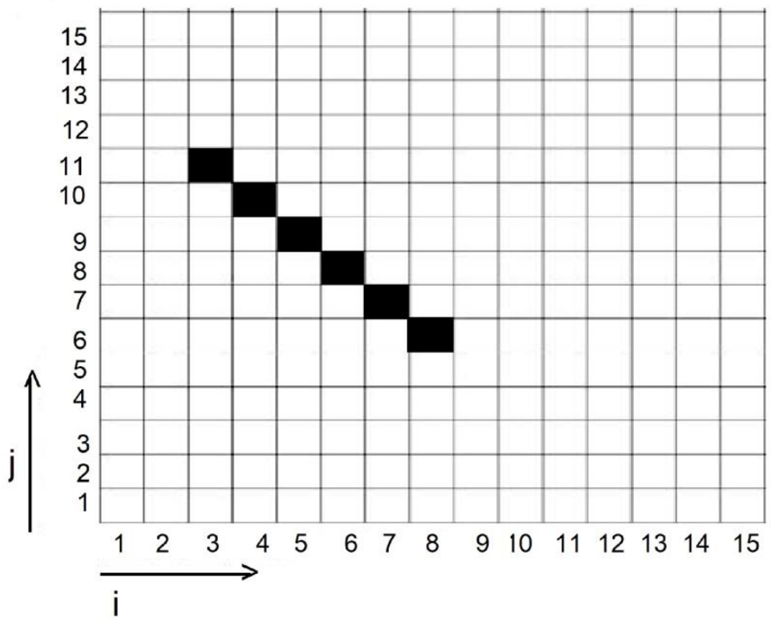
(-5, 9) (-4, 10) (-3, 11)

Now we need to change it again from 45° to 135°;

so we will obtain:

(8, 6) (7, 7) (6, 8)

(5, 9) (4, 10) (3, 11)



Now, we will do the same for the point $p_2' = (3,1,-2)$

First, we will change the point p_2' from 3D to 2D:

$$p_2' = (3,1,-2) \sim (-3/2, -1/2)$$

Since we know that s, X, and Y are the same (they do not change) we just need to calculate the rest; so:

$$x_2' = \frac{-3/2 + 2.415 - \frac{0.3219}{2}}{0.3219} = 2.34 \sim 2$$

$$y_2' = \frac{-1/2 - 2.415 + \frac{0.3219}{2}}{-0.3219} = 8.55 \sim 9$$

So we get:

$$p_1 \begin{cases} x_1 = 7.779 \sim 8 \\ y_1 = 6.226 \sim 6 \end{cases} \qquad p_2' \begin{cases} x_2' = 2.34 \sim 2 \\ y_2' = 8.55 \sim 9 \end{cases}$$

Now we can check the slope:

$$m = \frac{(y_2' - y_1)}{(x_2' - x_1)} = \frac{9 - 6}{2 - 8} = -0.5$$

m is not between 0 and 1, so we will change again to 45° because now we are in 135°.

So, we will flip the points:

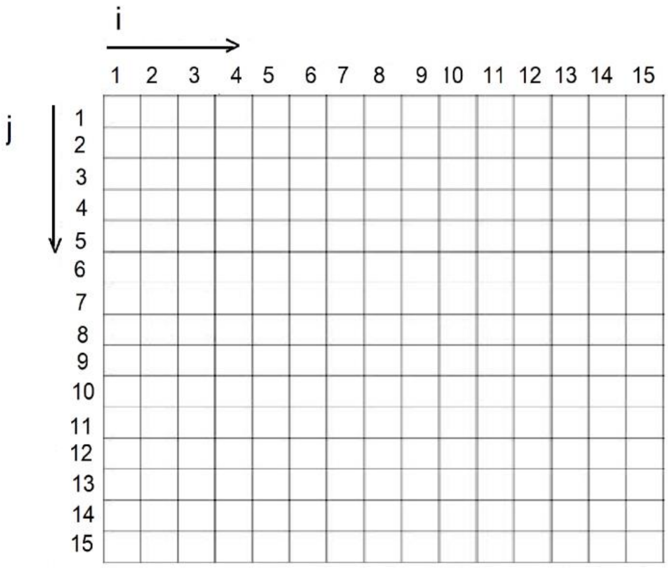
$p_1 = (-8, 6)$ and $p_2' = (-2, 9)$

And now:

$$m = \frac{(9-6)}{-2+8} = 0.5$$

m is between 0 and 1, so we can continue.

Now $x_1 < x_2$ (because $-8 < -2$). So, we will assume the next pixel representation:



Now we will apply the algorithm:

$x = -8$

$y = 6$

$dx = x_2 - x_1 = -2+8 = +6$

$dy = y_2 - y_1 = 9-6 = 3$

We will apply the algorithm 4 times (for i=0 to 3):

$f = -2dy + dx = -2 \cdot 3 + 6 = 0$

1) setpixel $(-8, 6)$

$x = -7$

$0 \nless 0$

entonces $y = 6$

$f = 0 - 2 \cdot 3 = -6$

2) setpixel $(-7, 6)$

$x = -6$

$-6 < 0$

$y = 7$

$f = -6 + 2 \cdot 6 = -6 + 12 = +6$

$f = 6 - 2 \cdot 3 = 0$

3) setpixel $(-6, 7)$

$x = -5$

$0 \nless 0$

so $y = 7$

$f = 0 - 2 \cdot 3 = -6$

4) setpixel $(-5, 7)$

$x = -4$

$-6 < 0$

so $y = 8$

$f = -6 + 12 = 6$

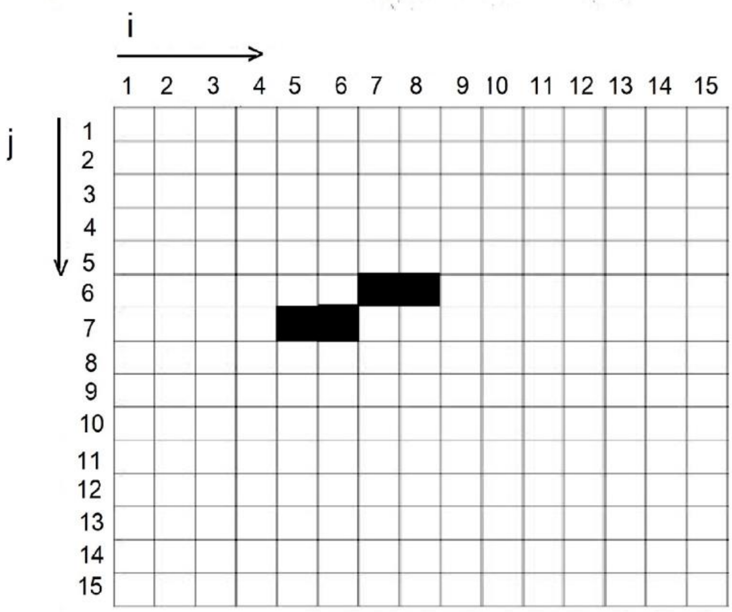
$f = 6 - 6 = 0$

So we obtained the next set of points:

(-8, 6) (-7, 6) (-6, 7) (-5, 7)

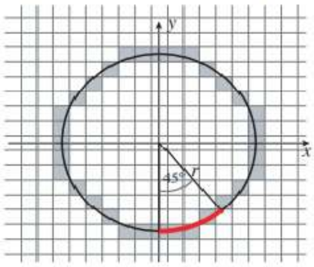
Now we need to change it again from 45° to 135°, so we will obtain:

(8, 6) (7, 6) (6, 7) (5, 7)



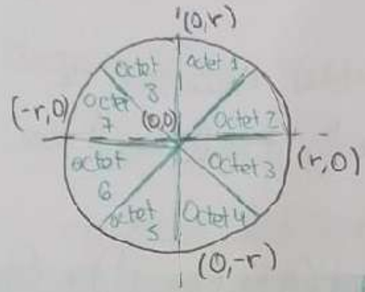
Exercise 2 [8 points]

Consider the problem of rasterizing a circle. Derive a version of Bresenham's algorithm for this task, and sketch a pseudo-code for it. Assume that your circle is already defined using pixel indices. To simplify your task, assume that the pixel indices can be negative, and the center of the circle is located at $(0,0)$. The radius of the circle is r. Note that in your derivation, you only have to consider 1/8 of the circle marked in red (see picture on the right). The rest of the pixels can be computed using symmetries. More precisely, if you color pixel (x,y), you should also color: $(-x,y)$, $(x,-y)$, $(-x,-y)$, (y,x), $(-y,x)$, $(y,-x)$, $(-y,-x)$. By deriving the algorithm for this small piece of the circle, you also can assume that you have to color only one pixel per column. This is similar to the assumption about 45° slope in the original line drawing algorithm. The derivation of the algorithm for the circle follows the same approach as for the line and uses a mid-point idea together with an implicit circle equation, which should have different sign depending on whether you are inside or outside of the circle.



Exercise 2

To compute this algorithm we need to know that we are in the coordinates (x_n, y_n) and what we want to get in order to represent the circle is the next pixel that we are going to color, this pixel will be the (x_{n+1}, y_{n+1})



To compute the pixels in a easier way we are goint to divide the circle in 8 octets. Thanks to symmetry we will obtain all the pixels by only compluting them in one octet

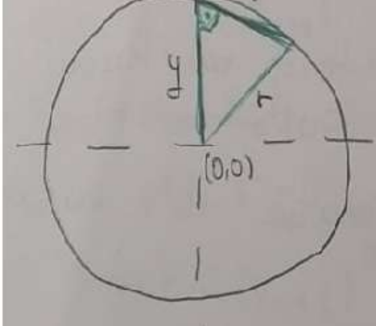
We will start in the ~~second~~ first octet and we will move in clockwise. This mean that we will have 2 options for our future pixel

Option 1: (x_{n+1}, y_n)

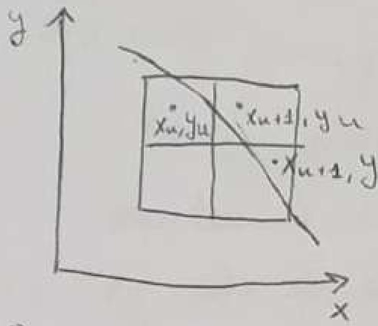
Option 2: (x_{n+1}, y_{n-1})

We know thanks to Pythagoras theorem that:

$$x^2 + y^2 = r^2$$



We asumme that center is in 0,0 to make calculus easier



$\rightarrow$ Option 1 $\Rightarrow$ We will call it pixel A

$\rightarrow$ Option 2 $\Rightarrow$ We will call it pixel B

So the point is to decide if the next pixel will be the option 1 or the option 2

- Note: The distance beetwen the center of the circle and any point of itsef will be the radius.

Distance of pixel A will be $\rightarrow$ $d(A) = (x_{n+1}^2) + (y_n)^2$

Dostance of pixel B will be $\rightarrow$ $d(B) = (x_{n+1})^2 + (y_n - 1)^2$

We are Seint to represent the Bresenham circle drawing algorithm to choose what will be the next pixel

$F(A) = d(A)^2 - r^2$

$F(B) = d(B)^2 - r^2$

Since we are in the first octant we know that $F(A) > 0$ because it's outsude the circle and $F(B) < 0$ because it's inside the circle

We are going to create the decision parameter D_n

$$D_n = F(A) + F(B)$$

- If $D_n < 0$. Means that the next pixel will be A
- If $D_n > 0$. Means that the next pixel will be B

We are going to compute the decision parameter:

$$D_n = (x_n+1)^2 + (x_n)^2 - r^2 + (x_{n+1})^2 + (y_n-1)^2 - r^2 \Rightarrow$$

$$\Rightarrow D_n = 2\cdot(X_n+1)^2 + (y_n)^2 + (y_n-1)^2 - 2r^2$$

The decision parameter of the next pixel will be:

First:

$$D_{n+1} = 2\cdot(X_{n+1}+1)^2 + (Y_{n+1})^2 + (Y_{n+1}-1)^2 - 2r^2 \Rightarrow$$

$$\Rightarrow D_{n+1} = 2\cdot(X_n+1+1)^2 + (Y_{n+1})^2 + (Y_{n+1}-1)^2 - 2r^2 \Rightarrow$$

$$\Rightarrow D_{n+1} = 2(X_n+2)^2 + (Y_{n+1})^2 + (Y_{n+1}-1)^2 - 2r^2$$

Second:

$$D_{n+1} - D_n \Rightarrow$$

$$\Rightarrow 2(X_n+2)^2 + (Y_{n+1})^2 + (Y_{n+1}-1)^2 - 2r^2 - [2\cdot(X_n+1)^2 + (y_n)^2 + (Y_n-1)^2 - 2r^2] \Rightarrow$$

$$\Rightarrow 2(X_n)^2 + 8X_n + 8 + (Y_{n+1})^2 + (Y_{n+1})^2 - 2Y_{n+1} + 1 - 2r^2 - 2(X_n)^2 - 4X_n - 2 -$$
$$- (Y_n)^2 - (Y_n)^2 - 2Y_n - 1 + 2r^2 \Rightarrow$$

$$\Rightarrow 4x_n + 2(Y_{n+1})^2 - 2Y_{n+1} - 2(Y_n)^2 + 2Y_n + 6 \Rightarrow$$

$$\Rightarrow D_{n+1} = D_n + 4x_n + 2(Y_{n+1})^2 - 2Y_{n+1} - 2(Y_n)^2 + 2Y_n + 6$$

The following decision parameter will change in function of the value that we get in D_n

- If $D_n < 0$. The pixel will be A and $y_{n+1} = y_n$

So, $D_{n+1} = D_n + 4X_n + 2(y_n)^2 - 2y_n - 2(y_n)^2 + 2y_n + 6 \Rightarrow$

$\Rightarrow D_{n+1} = D_n + 4x_n + 6$

- If $D_n > 0$. The pixel will be B and $y_{n+1} = y_n - 1$

So, $D_{n+1} = D_n + 4X_n + 2(y_n - 1)^2 - 2(y_n - 1) - 2(y_n)^2 + 2y_n + 6 \Rightarrow$

$\Rightarrow D_{n+1} = D_n + 4X_n + 2(y_n)^2 + 2 - 4y_n - 2y_n + 2 - 2(y_n)^2 + 2y_n + 6 \Rightarrow$

$\Rightarrow D_{n+1} = D_n + 4X_n - 4y_n + 10 \Rightarrow$

$\Rightarrow D_{n+1} = D_n + 4\cdot(X_n - y_n) + 10$

~~The initial decision parameter will be c~~

To know the initial decision parameter we need to remember that we were in the first octant so the point $(0, y)$ will be in the circle. Since we know that is a circle and we have r, the starting point will be $(0, r)$

Using this formula $2(X_n + 1)^2 + (y_n)^2 + (y_n - 1)^2 - 2r^2$ we get the initial decision parameter. $\Rightarrow$

~~$D_n = 2(X_n + 1)^2 +$~~

$D_0 = 2(0+1)^2 + r^2 + (r-1)^2 - 2r^2 \Rightarrow$

$\Rightarrow D_0 = 2 + r^2 + 1 - 2r - 2r^2 \Rightarrow$

$\Rightarrow D_0 = 3 - 2r$

Summary

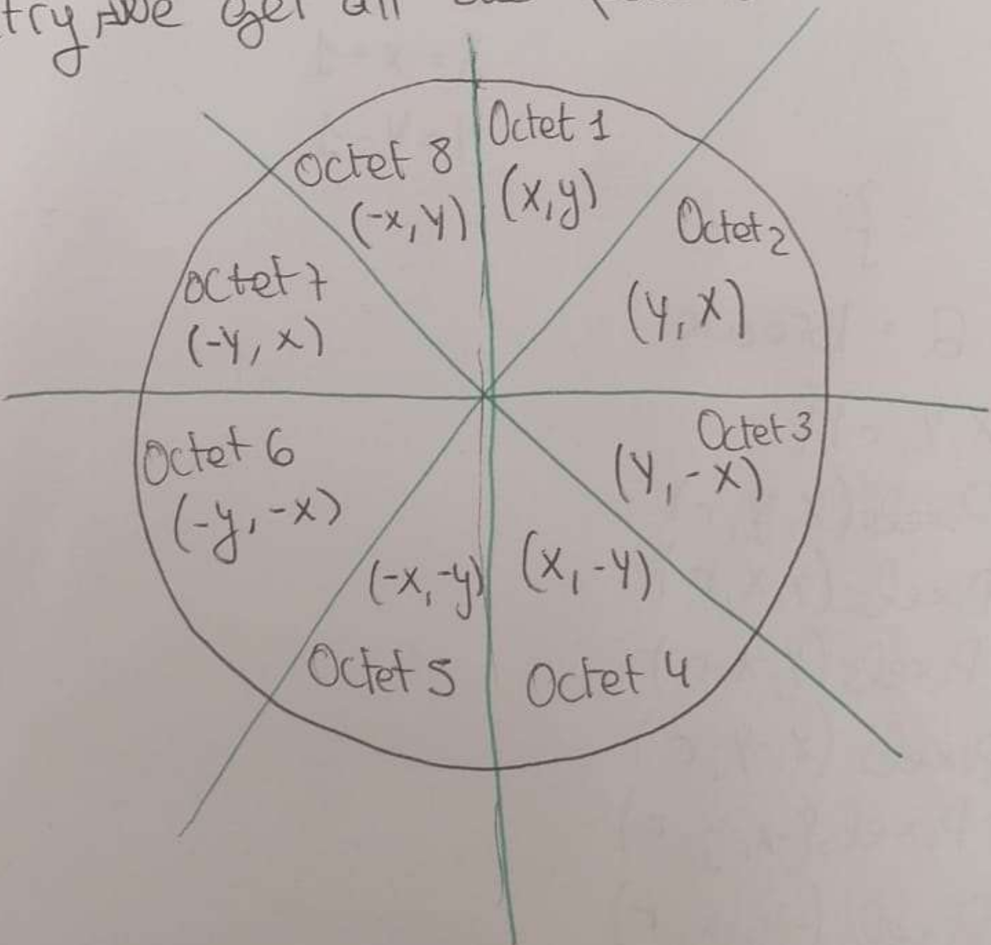
Initial decision parameter → $P_0 = 3 - 2r$

2 cases:

→ $P_n < 0$ —
- $X_{n+1} = X_n + 1$
- $Y_{n+1} = Y_n$
- $P_{n+1} = P_n + 4X_{n+1} + 6$

→ $P_n > 0$ —
- $X_{n+1} = X_n + 1$
- $Y_{n+1} = Y_n - 1$
- $P_{n+1} = P_n + 4(X_{n+1} - Y_{n+1}) + 10$

By simmetry we get all the points



Pseudo-code

```
    GetPixels(x, y, r){
Step 1 = X0 = 0 , Y0 = r
step 2 = D0 = 3-2·r
step 3 = while ( x ≤ y ) {
 step 4 =              SetPixel (x, y)
 step 5.1 =            If D < 0 then
                           D = D + 4x + 6
                           x = x + 1
  step 5.2 =           If D > 0 then
                           D = D + (x - y) + 10
                           x = x + 1
                           y = y - 1
          }
  step 6 = break
Circle (x, y, r){
     GetPixels(x, y, r)        // Octet 1
     GetPixels(y, x, r)        // Octet 2
     GetPixels(y, -x, r)       // Octet 3
     GetPixels(x, -y, r)       // Octet 4
     GetPixels(-x, -y, r)      // Octet 5
     GetPixels(-y, -x, r)      // Octet 6
     GetPixels(-y, x, r)       // Octet 7
     GetPixels(-x, y, r)       // Octet 8
}
```