UFN
Universidade Franciscana
Somos universidade. Somos você.

# Curso de Jogos Digitais

## Disciplina de Computação Gráfica

## Primitivas Geométricas - Círculos

Aula 05 - Parte I

Professor: André Flores dos Santos



**Santa Maria – RS 2025**

# SUMÁRIO

# Círculos

- Círculos e arcos são fundamentais em Computação Gráfica, uma vez que permitem a geração de curvas e superfícies;

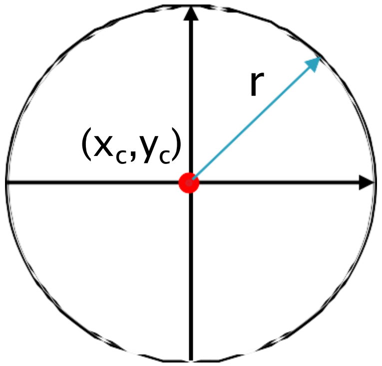
- O traçado de um círculo é definido por um conjunto de pontos que estão a uma mesma distância de um ponto:

  - Tal distância é o raio e o ponto é o centro do círculo;

- Matematicamente, a equação do círculo é dada por:

$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

# Círculos




- É útil também isolarmos a equação em termos de x e y:

$$x = x_c \pm \sqrt{r^2 - (y - y_c)^2}$$
$$y = y_c \pm \sqrt{r^2 - (x - x_c)^2}$$

- O "±" significa que, por exemplo, na 2ª expressão, para cada valor de x, são obtidos dois valores de y;

  Isso ocorre porque um círculo é simétrico em relação ao seu centro, portanto, existem dois pontos simétricos em relação ao eixo x (ou y) para cada x (ou y) específico.

# Círculo na origem

- Para um círculo cujo ponto central é na **origem** ($x_c = 0$ e $y_c = 0$), a equação fica:

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

- Em Comp. Gráfica é interessante trabalharmos com **círculos na origem**, pelo fato de sua equação ser mais simples, e então aplicarmos transformações ao mesmo para modificar sua posição.

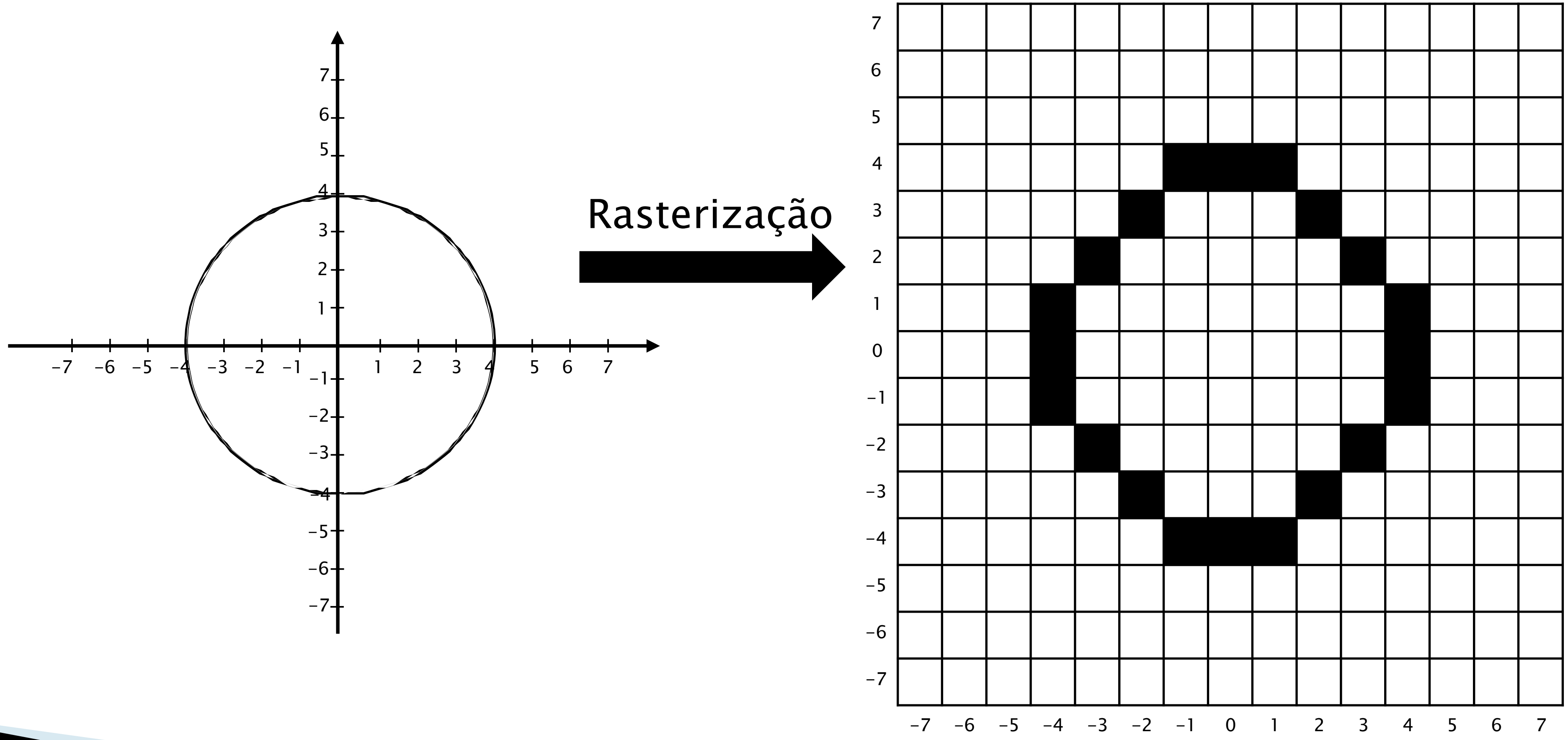
- Em termos de x e y, a equação fica:

$$x = \pm\sqrt{r^2 - y^2}$$

$$y = \pm\sqrt{r^2 - x^2}$$

# Rasterização de círculos

- Problema:

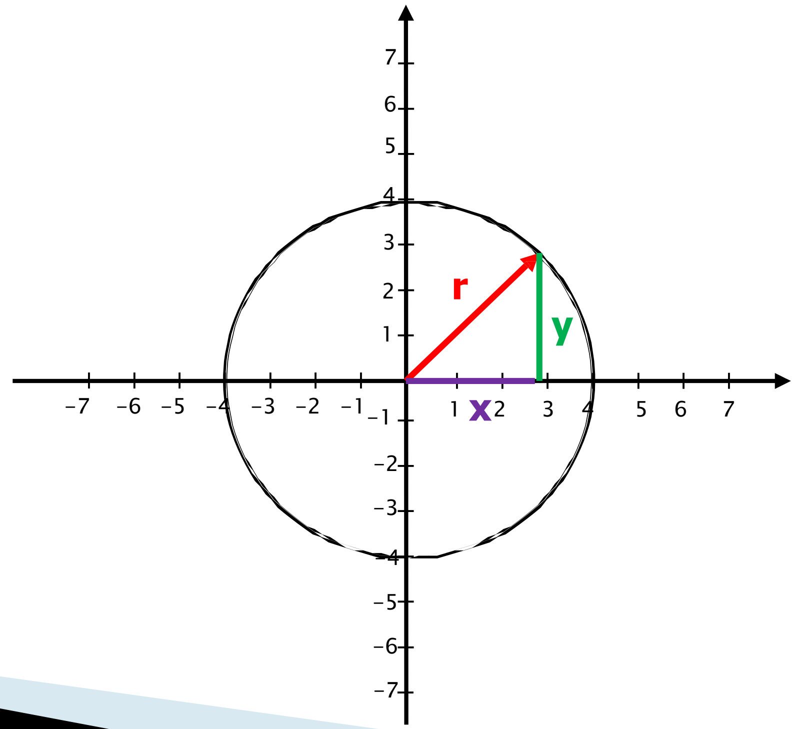


# Solução 1

- Como vimos, a equação do círculo é:

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

- As variáveis da equação refletem o que é apresentado na figura abaixo:

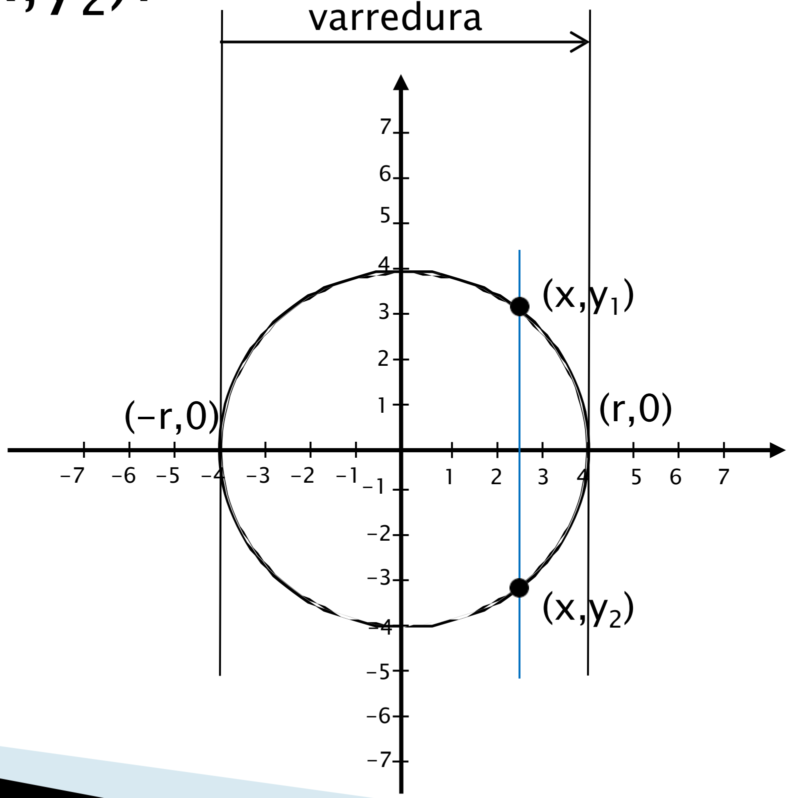


# Solução 1

- Esse tipo de equação é denominado **equação implícita**, pois podemos substituir nela os valores para **x**, **y** e **r** para verificar se a mesma permanece verdadeira.
    - Caso positivo, isso significa que o ponto (x,y) pertence ao círculo.
    - Ex: Em um circunferência de raio = 4, o ponto (4,0) pertence ao circunferência?
        - $x^2 + y^2 = r^2$
        - $4^2 + 0^2 = 4^2$
        - $\mathbf{16 = 16}$
        - Portanto, o ponto (4,0) pertence a circunferência.

# Solução 1

- Na figura abaixo, podemos perceber que para cada **x**, que varia de **−r** até **r**, temos dois valores possíveis para **y**, que são os pontos $(x,y_1)$ e $(x,y_2)$:



# Solução 1

- Portanto, a solução mais simples seria fazer uma varredura para cada **x** de **$-r$** até **r**, para descobrirmos os valores de **$y_1$** e **$y_2$**. Desta forma, ao final da varredura, nós poderemos desenhar um círculo;

- Sendo assim, precisamos determinar os valores de **$y_1$** e **$y_2$** para cada **x**;

- Utilizando a equação implícita da circunferência, rearranjamos os termos da mesma de modo a isolar o y:

$$x^2 + y^2 = r^2$$
$$y^2 = r^2 - x^2$$
$$\boldsymbol{y = \pm\sqrt{r^2 - x^2}}$$

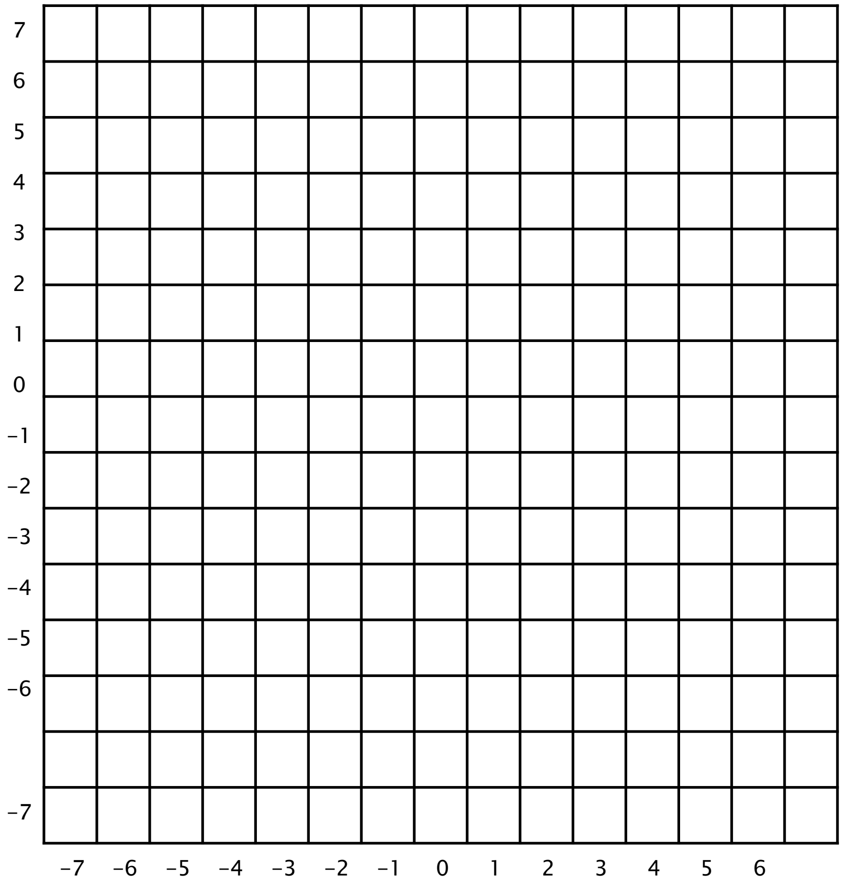
- A raiz quadrada, portanto, retorna um valor positivo e um valor negativo para **y**, ou seja, **$y_1$** e **$y_2$**.

# Solução 1

- Exemplo: rasterizar um círculo de raio = 4

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **x** | | | | | | | | | |
| $y_1$ | | | | | | | | | |
| $y_2$ | | | | | | | | | |

Vamos calcular, para cada **x** de **–4** até **4**, os valores de $\mathbf{y_1}$ e $\mathbf{y_2}$



# Solução 1

- Exemplo: rasterizar um círculo de raio = 4

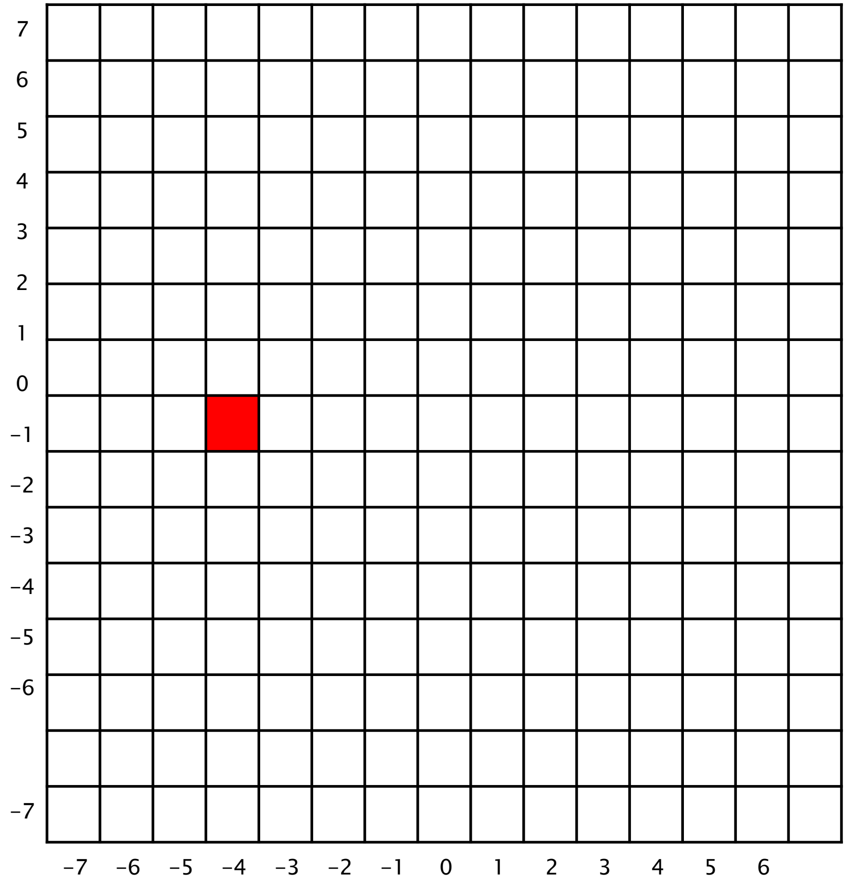
| x | -4 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $y_1$ | 0 | | | | | | | | |
| $y_2$ | 0 | | | | | | | | |

Para x=-4:

$$y = \sqrt{r^2 - x^2}$$
$$y = \sqrt{4^2 - (-4^2)}$$
$$y = \sqrt{16 - 16}$$
$$y = 0$$

→ $y_1$=0
→ $y_2$=0

Pintamos os pontos **(-4,0)** e **(-4,0)**



# Solução 1

- Exemplo: rasterizar um círculo de raio = 4

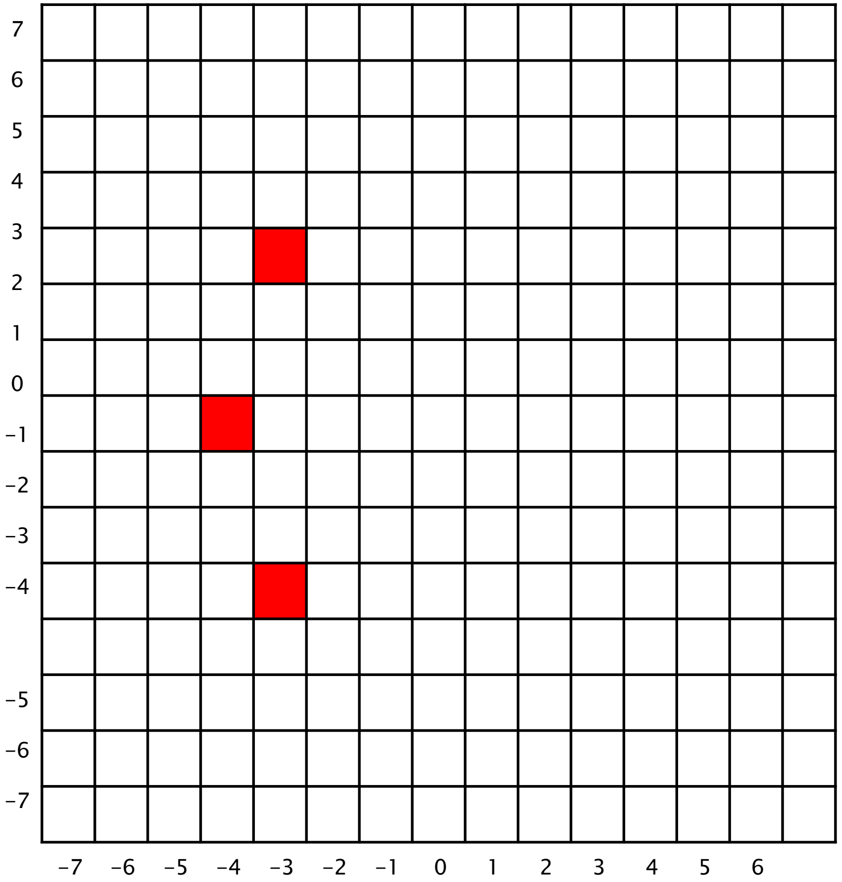
| **x** | **-4** | **-3** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $y_1$ | 0 | 3 | | | | | | | |
| $y_2$ | 0 | −3 | | | | | | | |

Para x=−3:

$$\boldsymbol{y = \sqrt{r^2 - x^2}}$$
$$\boldsymbol{y = \sqrt{4^2 - (-3^2)}}$$
$$\boldsymbol{y = \sqrt{16 - 9}}$$
$$\boldsymbol{y = \sqrt{7} = 2.64}$$ → $y_1$=3

(arredondando) → $y_2$=−3

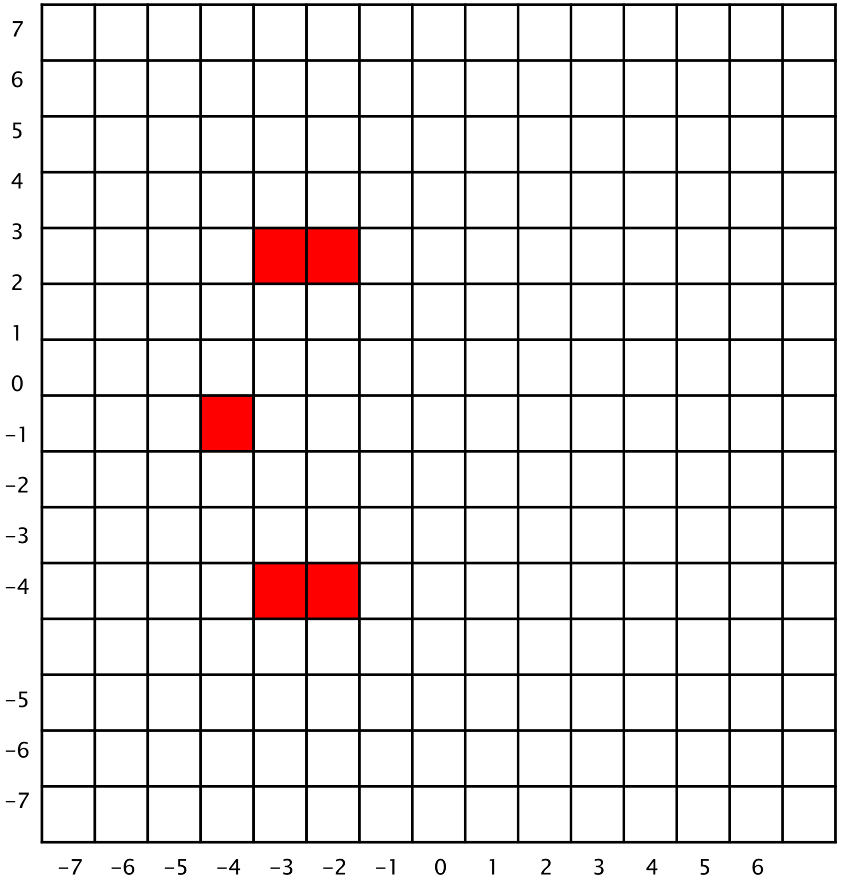
Pintamos os pontos **(-3,3)** e **(-3,-3)**



# Solução 1

- Exemplo: rasterizar um círculo de raio = 4

| x | -4 | -3 | -2 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $y_1$ | 0 | 3 | 3 | | | | | | |
| $y_2$ | 0 | −3 | −3 | | | | | | |




Para x=−2:

$$y = \sqrt{r^2 - x^2}$$
$$y = \sqrt{4^2 - (-2^2)}$$
$$y = \sqrt{16 - 4}$$
$$y = \sqrt{12} = 3.46$$ → $y_1$=3

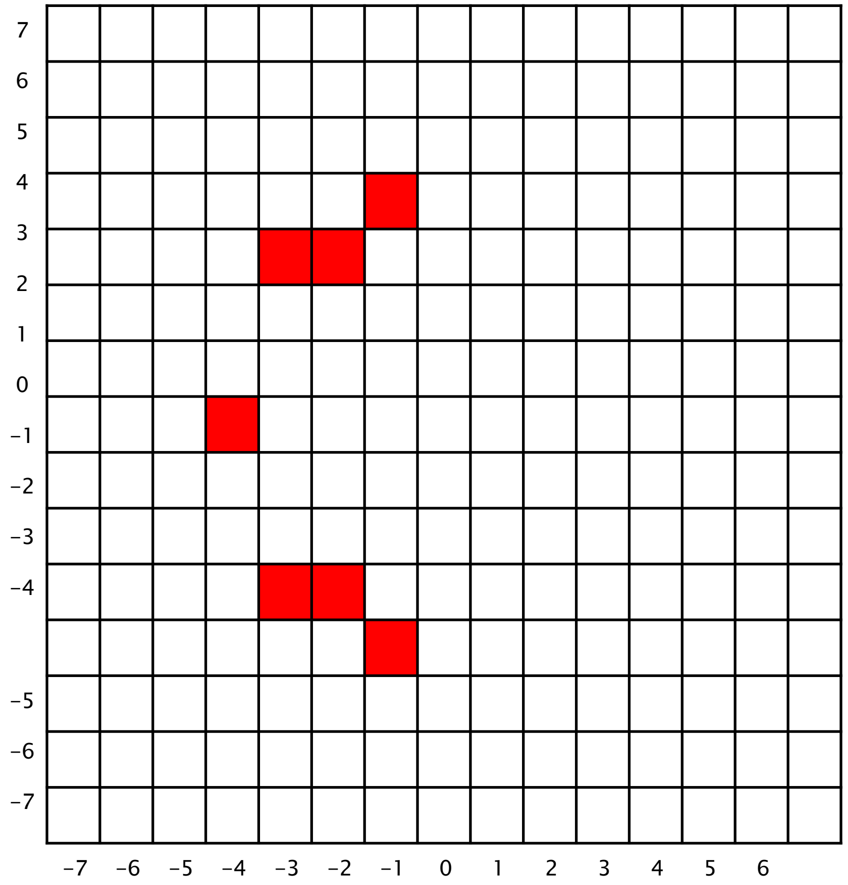
(arredondando) → $y_2$=−3

Pintamos os pontos **(-2,3)** e **(-2,-3)**

# Solução 1

- Exemplo: rasterizar um círculo de raio = 4

| x | -4 | -3 | -2 | -1 | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $y_1$ | 0 | 3 | 3 | 4 | | | | | |
| $y_2$ | 0 | −3 | −3 | −4 | | | | | |




Para x=−1:

$$\boldsymbol{y = \sqrt{r^2 - x^2}}$$

$$\boldsymbol{y = \sqrt{4^2 - (-1^2)}}$$

$$\boldsymbol{y = \sqrt{16 - 1}}$$

$$\boldsymbol{y = \sqrt{15} = 3.87} \quad \rightarrow y_1=4$$

(arredondando) $\rightarrow y_2=-4$

Pintamos os pontos **(-1,4)** e **(-1,-4)**

# Solução 1

- Exemplo: rasterizar um círculo de raio = 4

| x | -4 | -3 | -2 | -1 | 0 | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $y_1$ | 0 | 3 | 3 | 4 | 4 | | | | |
| $y_2$ | 0 | −3 | −3 | −4 | −4 | | | | |

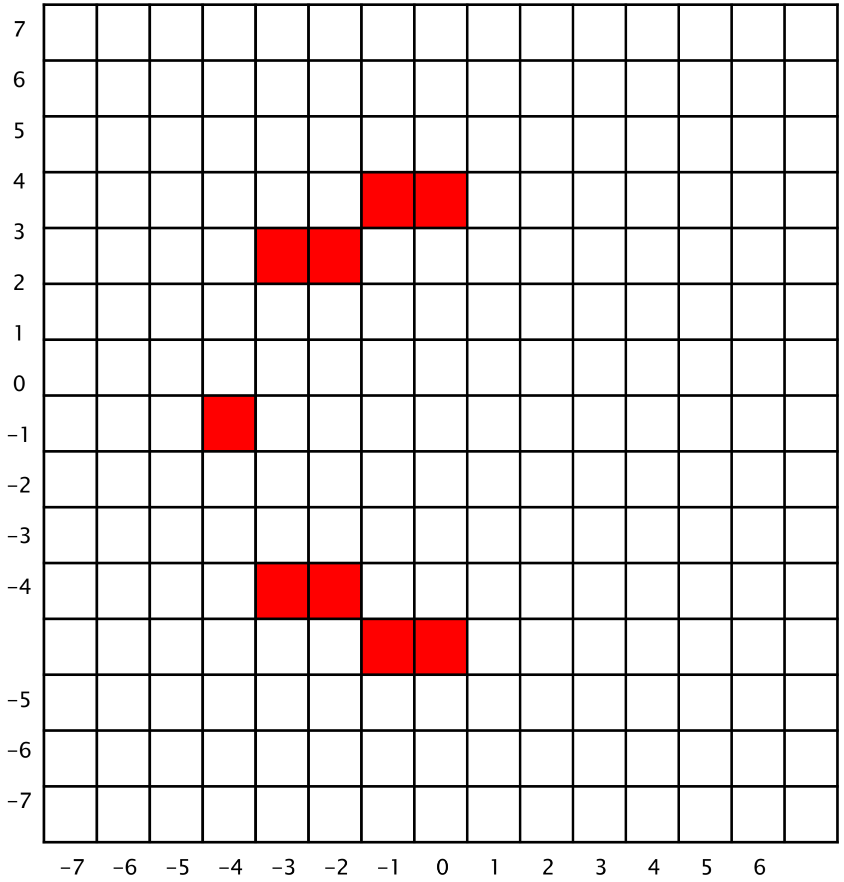
Para x=0:

$$\boldsymbol{y = \sqrt{r^2 - x^2}}$$
$$\boldsymbol{y = \sqrt{4^2 - (-0^2)}}$$
$$\boldsymbol{y = \sqrt{16 - 0}}$$
$$\boldsymbol{y = \sqrt{16} = 4}$$

→ $y_1=4$

→ $y_2=-4$

Pintamos os pontos **(0,4)** e **(0,-4)**



# Solução 1

- Exemplo: rasterizar um círculo de raio = 4

| x | -4 | -3 | -2 | -1 | 0 | 1 | | | |
|---|---|---|---|---|---|---|---|---|---|
| $y_1$ | 0 | 3 | 3 | 4 | 4 | 4 | | | |
| $y_2$ | 0 | −3 | −3 | −4 | −4 | −4 | | | |

Para x=1:

$$y = \sqrt{r^2 - x^2}$$

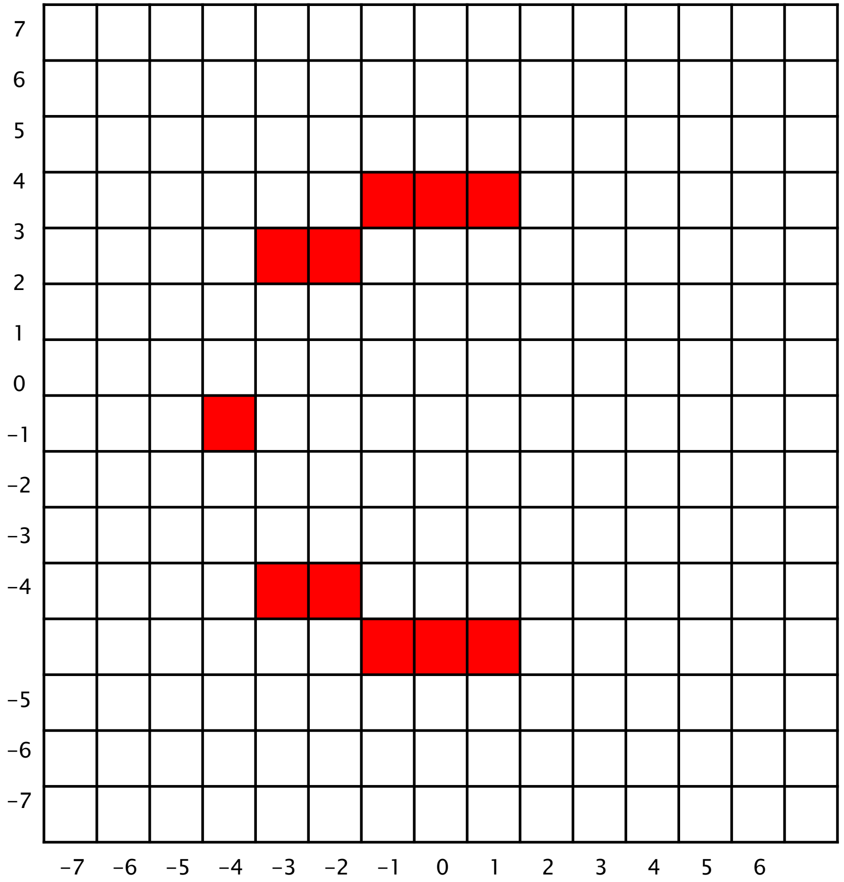
$$y = \sqrt{4^2 - (1^2)}$$

$$y = \sqrt{16 - 1}$$

$$y = \sqrt{15} = 3.87 \quad \rightarrow y_1 = 4$$

(arredondando) $\rightarrow y_2 = -4$

Pintamos os pontos **(1,4)** e **(1,-4)**



# Solução 1

- Exemplo: rasterizar um círculo de raio = 4

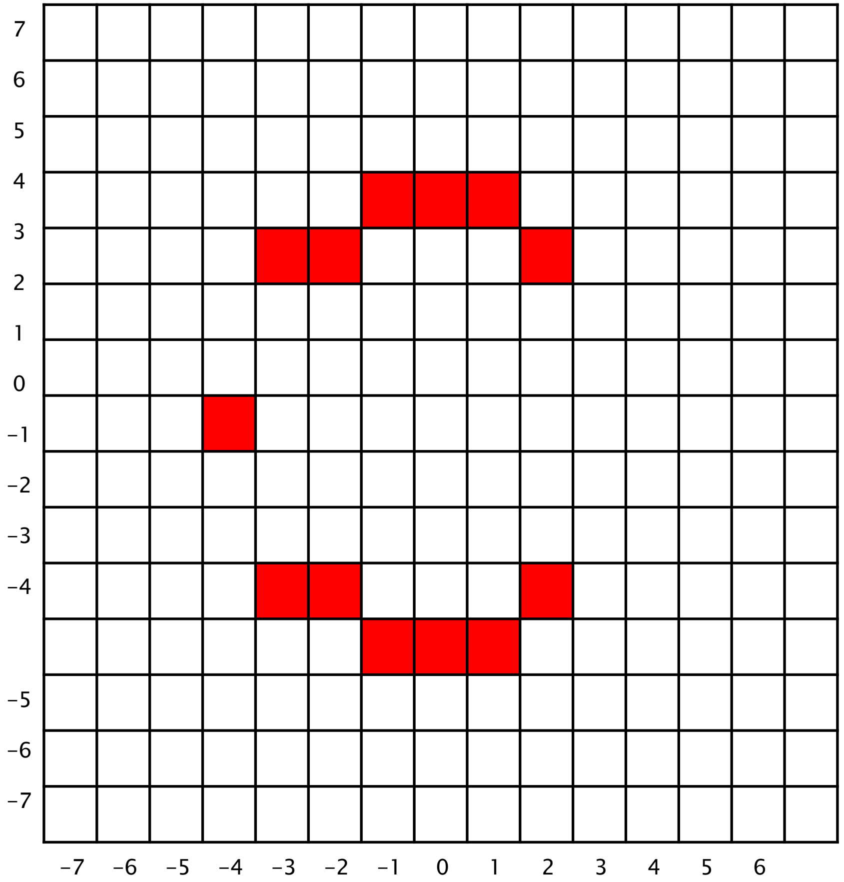
| x | -4 | -3 | -2 | -1 | 0 | 1 | 2 | | |
|---|---|---|---|---|---|---|---|---|---|
| $y_1$ | 0 | 3 | 3 | 4 | 4 | 4 | 3 | | |
| $y_2$ | 0 | −3 | −3 | −4 | −4 | −4 | −3 | | |

Para x=2:

$$y = \sqrt{r^2 - x^2}$$
$$y = \sqrt{4^2 - (2^2)}$$
$$y = \sqrt{16 - 4}$$
$$y = \sqrt{12} = 3.46 \quad \rightarrow y_1 = 3$$

(arredondando) $\rightarrow y_2 = -3$

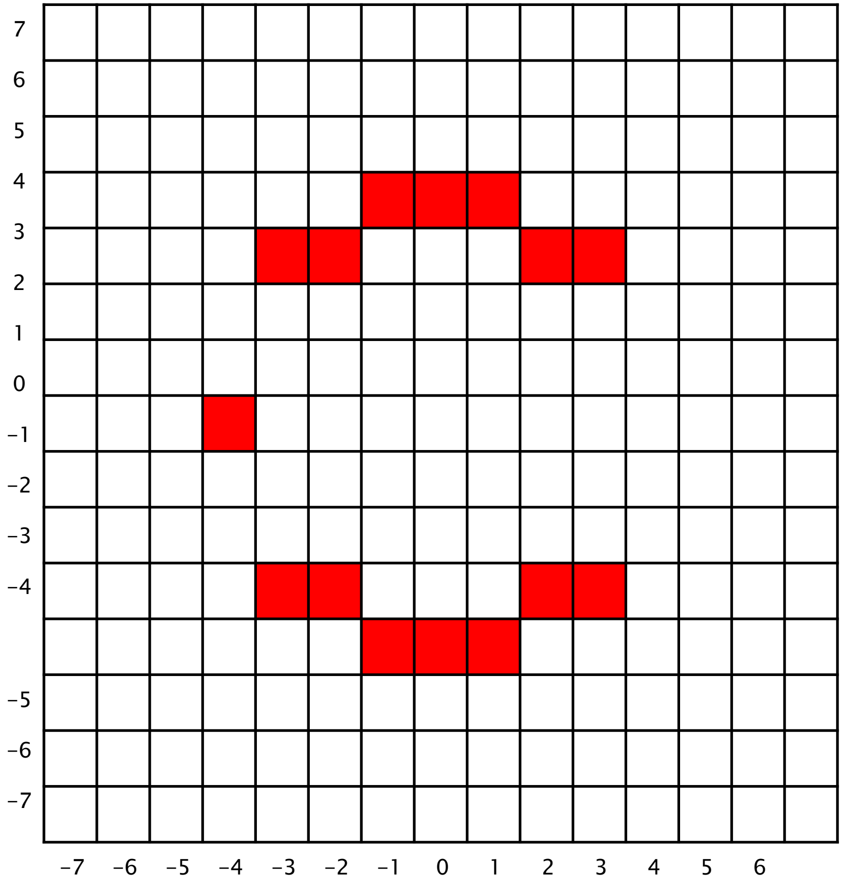
Pintamos os pontos **(2,3)** e **(2,-3)**



# Solução 1

- Exemplo: rasterizar um círculo de raio = 4

| x | -4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | |
|---|---|---|---|---|---|---|---|---|---|
| $y_1$ | 0 | 3 | 3 | 4 | 4 | 4 | 3 | 3 | |
| $y_2$ | 0 | −3 | −3 | −4 | −4 | −4 | −3 | −3 | |




Para x=3:

$$\boldsymbol{y = \sqrt{r^2 - x^2}}$$
$$\boldsymbol{y = \sqrt{4^2 - (3^2)}}$$
$$\boldsymbol{y = \sqrt{16 - 9}}$$
$$\boldsymbol{y = \sqrt{7} = 2.64}$$

→ $y_1$=3

(arredondando) → $y_2$=−3

Pintamos os pontos **(3,3)** e **(2,-3)**

# Solução 1

- Exemplo: rasterizar um círculo de raio = 4

| x | -4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|
| $y_1$ | 0 | 3 | 3 | 4 | 4 | 4 | 3 | 3 | 0 |
| $y_2$ | 0 | −3 | −3 | −4 | −4 | −4 | −3 | −3 | 0 |

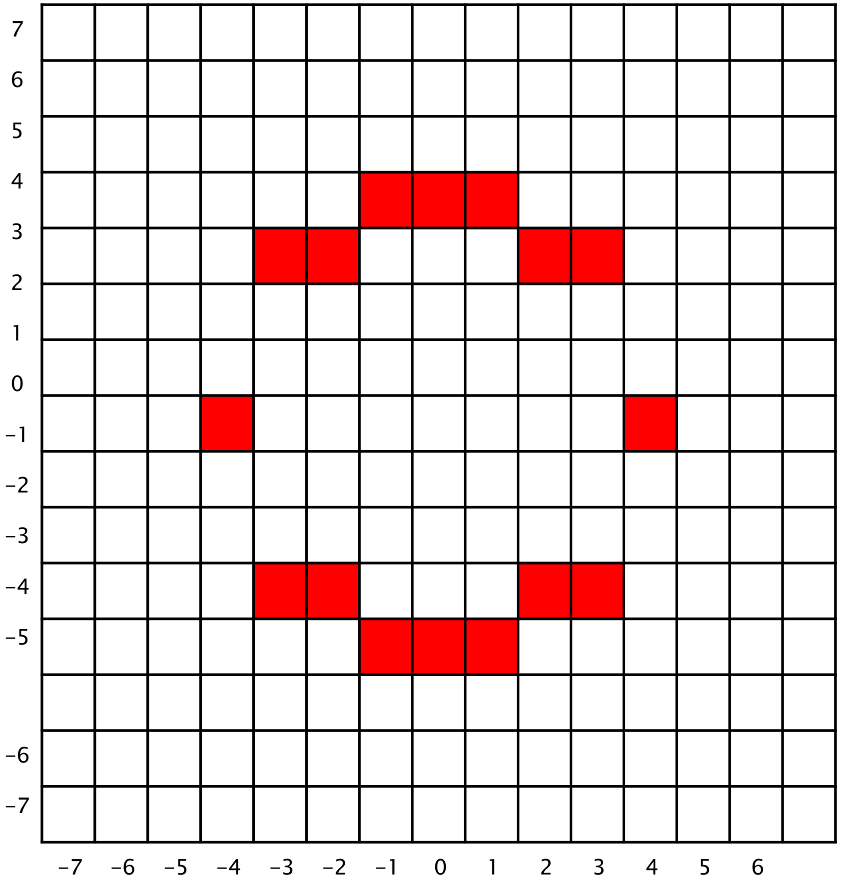
Para x=4:

$$y = \sqrt{r^2 - x^2}$$
$$y = \sqrt{4^2 - (4^2)}$$
$$y = \sqrt{16 - 16}$$
$$y = 0$$

→ $y_1$=0

→ $y_2$=0

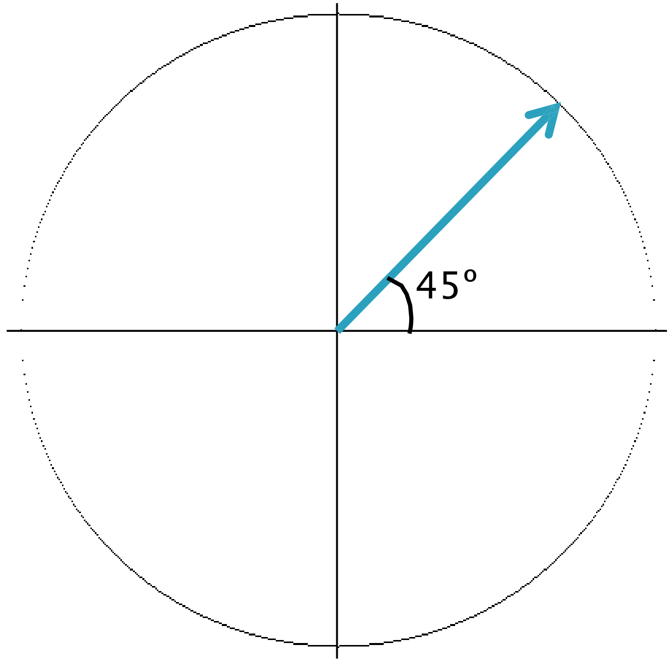
Pintamos os pontos **(4,0)** e **(4,0)**



# Solução 1

- Como podemos perceber, a solução 1 rasteriza um círculo, porém o mesmo é **descontínuo**;
- Um detalhe importante de se observar é que a descontinuidade acontece entre os ângulos de 0 e 45º.
- O mesmo algoritmo, utilizado para rasterizar um círculo em uma resolução maior, resulta em algo parecido com isso:

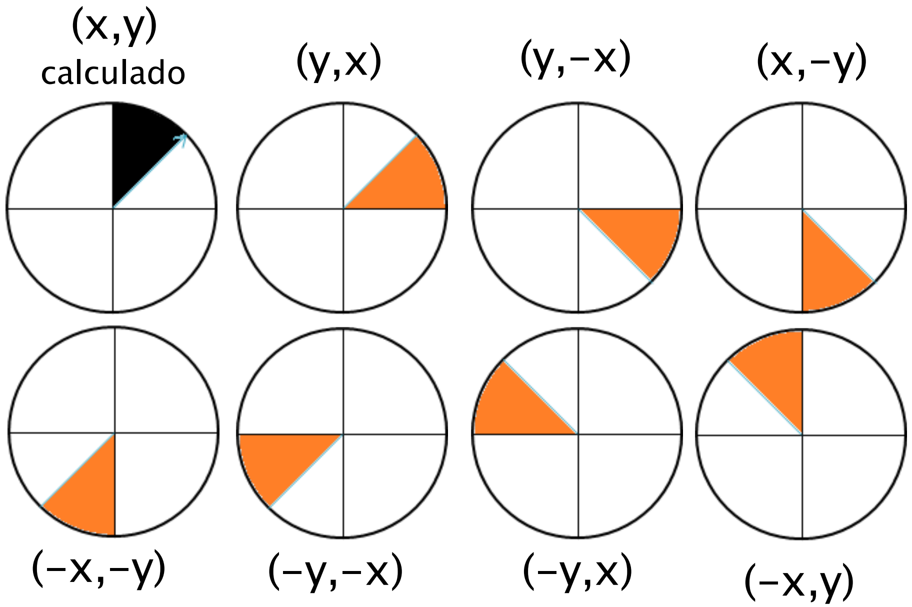
**Observa-se que apesar da descontinuidade acontecer entre 0 e 45º, ela não acontece entre os ângulos de 45º e 90º!**



# Solução 2

- Na solução anterior, nós processamos somente um semicírculo:
  - Pois para cada x calculado, calculamos o valor de y correspondente (x,y) e então espelhamos o mesmo (x,-y);

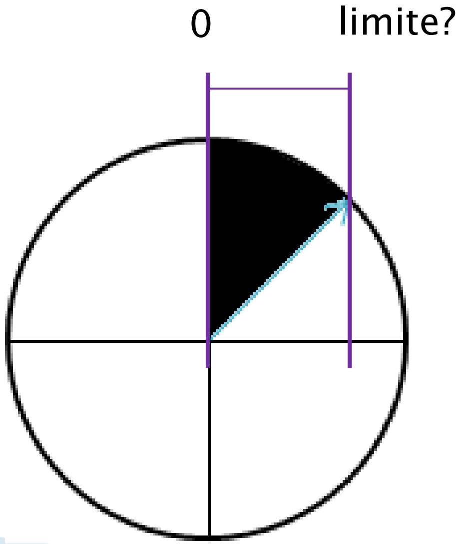
- Para resolver este problema, podemos seguir a mesma ideia, mas ao invés de calcular todos os pontos do semicírculo, calculamos somente os pontos entre **45º e 90º** (em que não há descontinuidade), e então espelhamos o mesmo nos **8 octantes**:



# Solução 2

- Na solução 1, fizemos o **x** variar de **–r** até **r**, percorrendo valores de 0º a 180º (semicírculo);

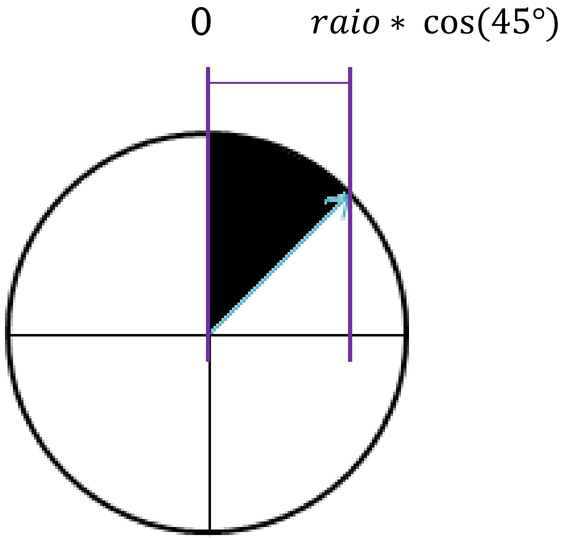
- Para percorrermos de 90º a 45º, precisamos fazer **x** variar de quanto até quanto?



# Solução 2

- De acordo com as **coordenadas polares**, podemos definir que limite é:

$$limite = raio * \cos(45°)$$



# Solução 2

- Exemplo: rasterizar um círculo de raio = 4

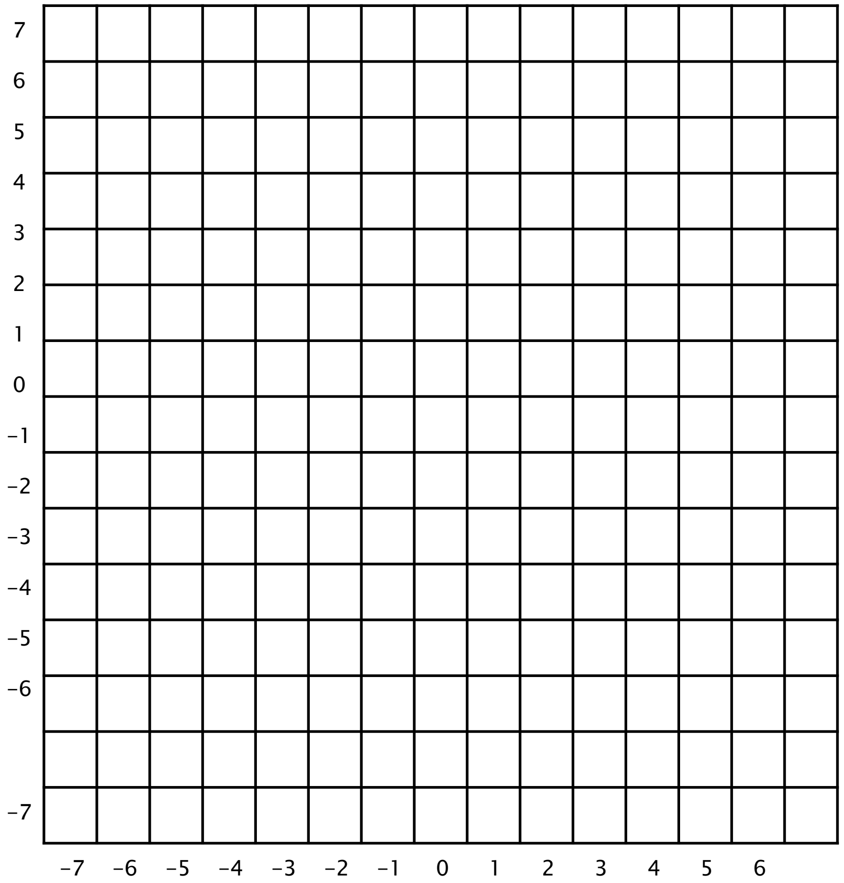
| **x** | | | |
|---|---|---|---|
| y | | | |

Calculamos o limite:

$$limite = raio * cos(45°)$$
$$limite = 4 * cos(45°)$$
$$limite = 4 * 0.7071 = 2.828$$

Como o limite precisa ser inteiro, podemos truncar para 2.

Desta forma, calculamos os valores de **y** para **x** variando de 0 a 2.



# Solução 2

- Exemplo: rasterizar um círculo de raio = 4

| x | 0 | | |
|---|---|---|---|
| y | 4 | | |

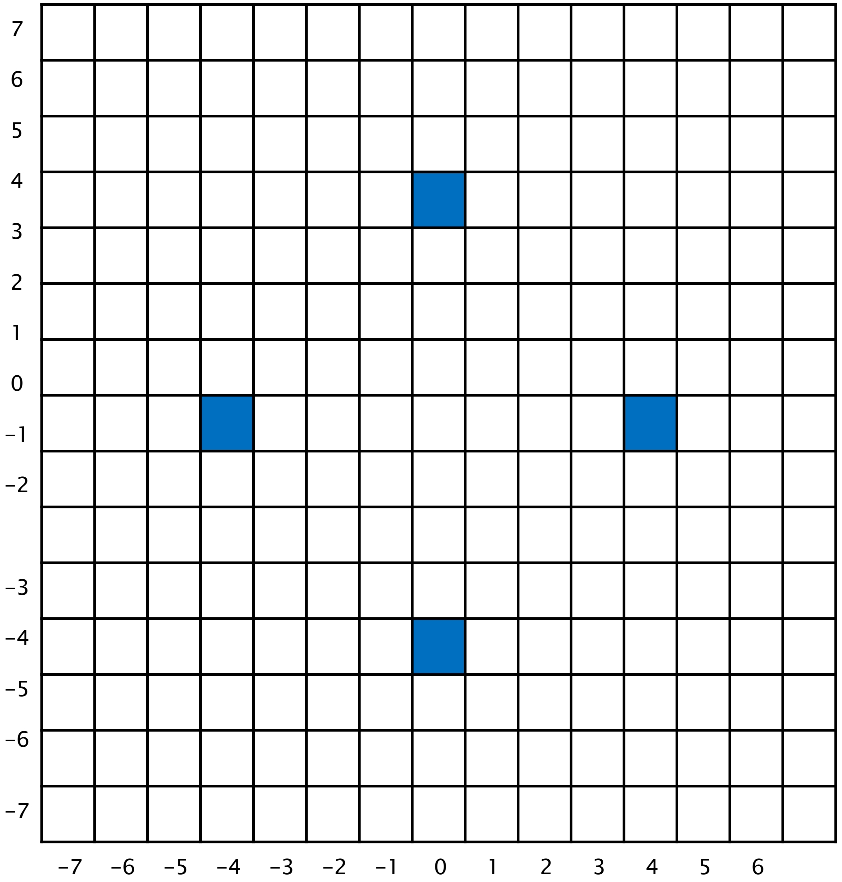
Para x=0:

$$y = \sqrt{r^2 - x^2}$$
$$y = \sqrt{4^2 - (0^2)}$$
$$y = 4$$

Pintamos os pontos:

(x,y) → (0,4)
(x,-y) → (0,-4)
(-x,y) → (0,4)
(-x,-y) → (0,-4)

(y,x) → (4,0)
(-y,x) → (-4,0)
(y,-x) → (4,0)
(-y,-x) → (-4,0)



# Solução 2

- Exemplo: rasterizar um círculo de raio = 4

| x | 0 | 1 | |
|---|---|---|---|
| y | 4 | 4 | |

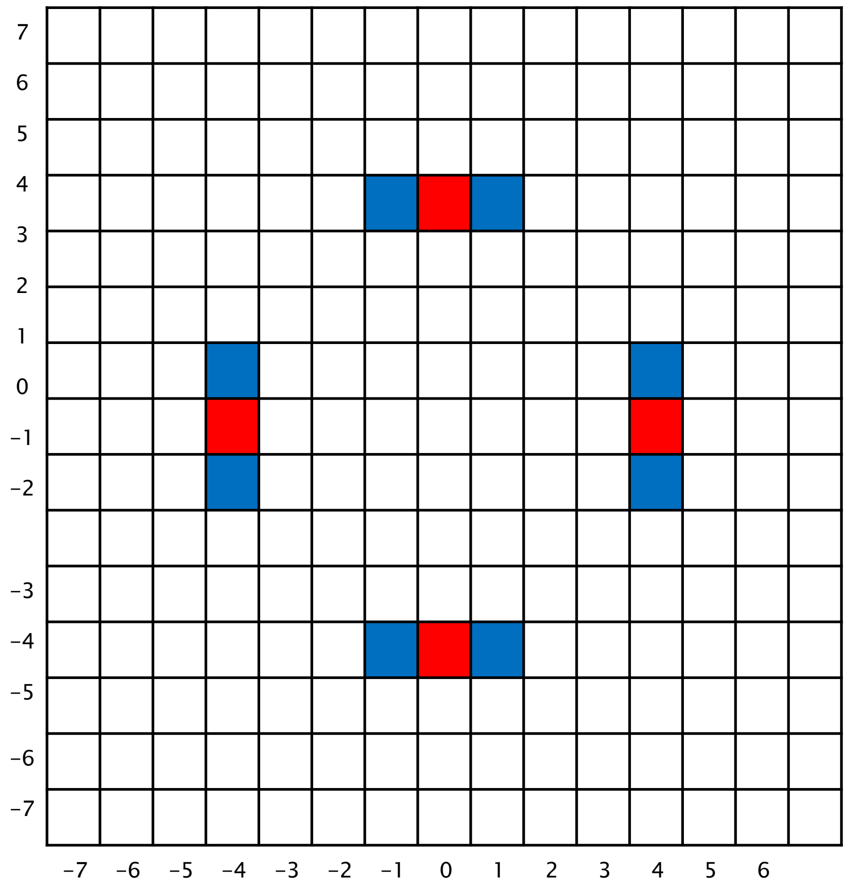
Para x=1:

$$y = \sqrt{r^2 - x^2}$$
$$y = \sqrt{4^2 - (1^2)}$$
$$y = 3.87 \cong 4$$

Pintamos os pontos:

(x,y) → (1,4)
(x,−y) → (1,−4)
(−x,y) → (−1,4)
(−x,−y) → (−1,−4)

(y,x) → (4,1)
(−y,x) → (−4,1)
(y,−x) → (4,−1)
(−y,−x) → (−4,−1)



# Solução 2

- Exemplo: rasterizar um círculo de raio = 4

| x | 0 | 1 | 2 |
|---|---|---|---|
| y | 4 | 4 | 3 |

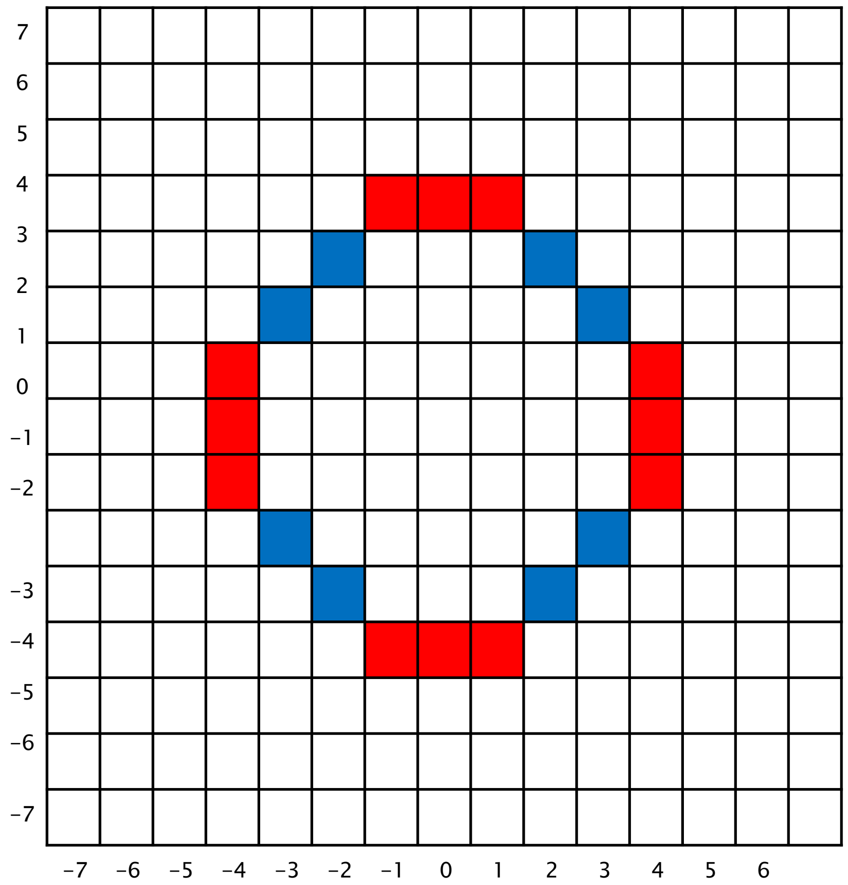
Para x=2:

$$y = \sqrt{r^2 - x^2}$$
$$y = \sqrt{4^2 - (2^2)}$$
$$y = 3.46 \cong 3$$

Pintamos os pontos:

(x,y) → (2,3)
(x,−y) → (2,−3)
(−x,y) → (−2,3)
(−x,−y) → (−2,−3)

(y,x) → (3,2)
(−y,x) → (−3,2)
(y,−x) → (3,−2)
(−y,−x) → (−3,−2)

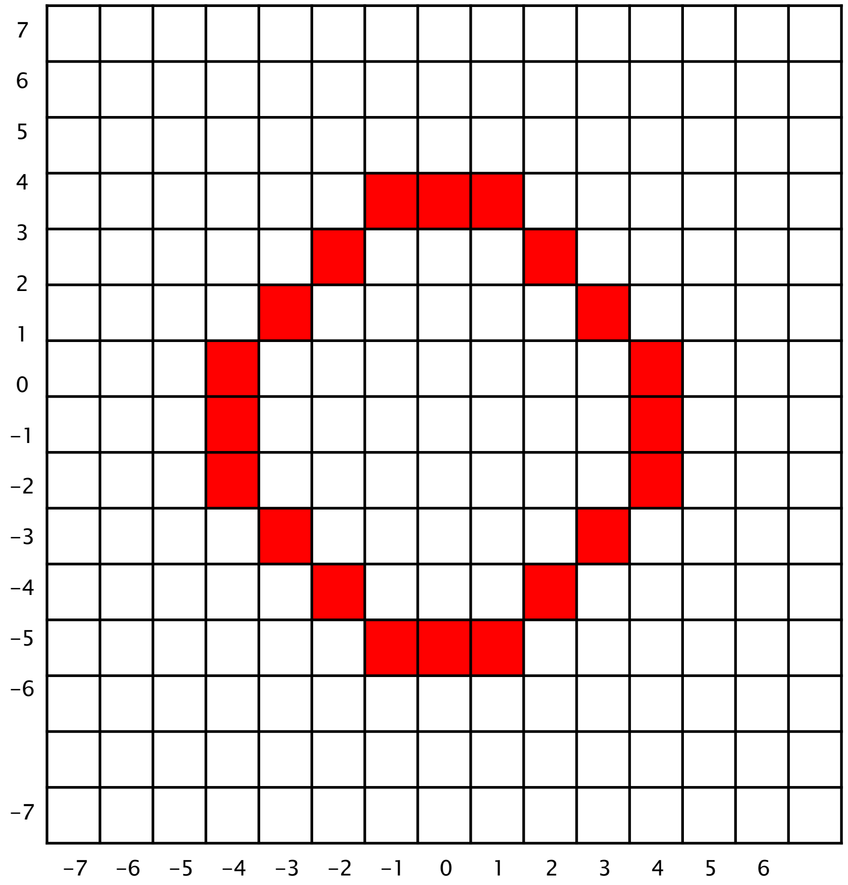


# Solução 2

- Exemplo: rasterizar um círculo de raio = 4

| x | 0 | 1 | 2 |
|---|---|---|---|
| y | 4 | 4 | 3 |

**Círculo finalizado!**



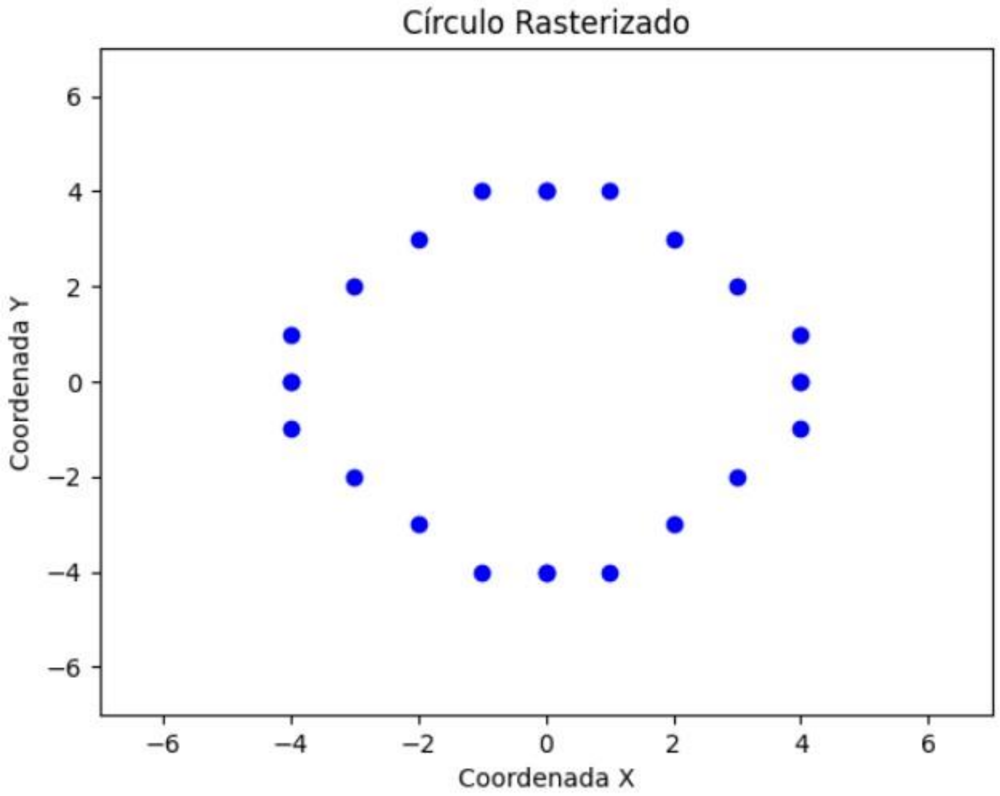
# Solução 2 - Algoritmo em python

```
import math
import matplotlib.pyplot as plt

# Defina o raio do círculo
raio = 4

# Inicialize a coordenada x
x = 0

# Lista para armazenar os pontos
pontos = []


# Calcule o limite para a varredura no eixo x
limite = raio * math.cos(math.radians(45))

# Loop enquanto x for menor que o limite
while x <= limite:
    # Calcule a coordenada y correspondente usando a equação do círculo
    y = round(math.sqrt(raio * raio - x * x))
    print("(",x,",",y,")")
    print("(",y,",",x,")")
    print("(",y,",",-x,")")
    print("(",x,",",-y,")")
    print("(",-x,",",-y,")")
    print("(",-y,",",-x,")")
    print("(",-y,",",x,")")
    print("(",-x,",",y,")")
```

```
# Adicione os pontos à lista considerando a simetria do círculo
    pontos.extend([(x, y), (y, x), (y, -x), (x, -y), (-x, -y), (-y, -x), (-y, x), (-x, y)])

    # Incremente x para avançar na varredura
    x += 1

# Separe as coordenadas x e y para plotagem
coordenadas_x, coordenadas_y = zip(*pontos)

# Plote os pontos
plt.plot(coordenadas_x, coordenadas_y, 'bo')  # 'bo-' para pontos azuis conectados por linhas

# Adicione título e rótulos dos eixos
plt.title("Círculo Rasterizado")
plt.xlabel("Coordenada X")
plt.ylabel("Coordenada Y")

# Mostre o gráfico
plt.show()
```

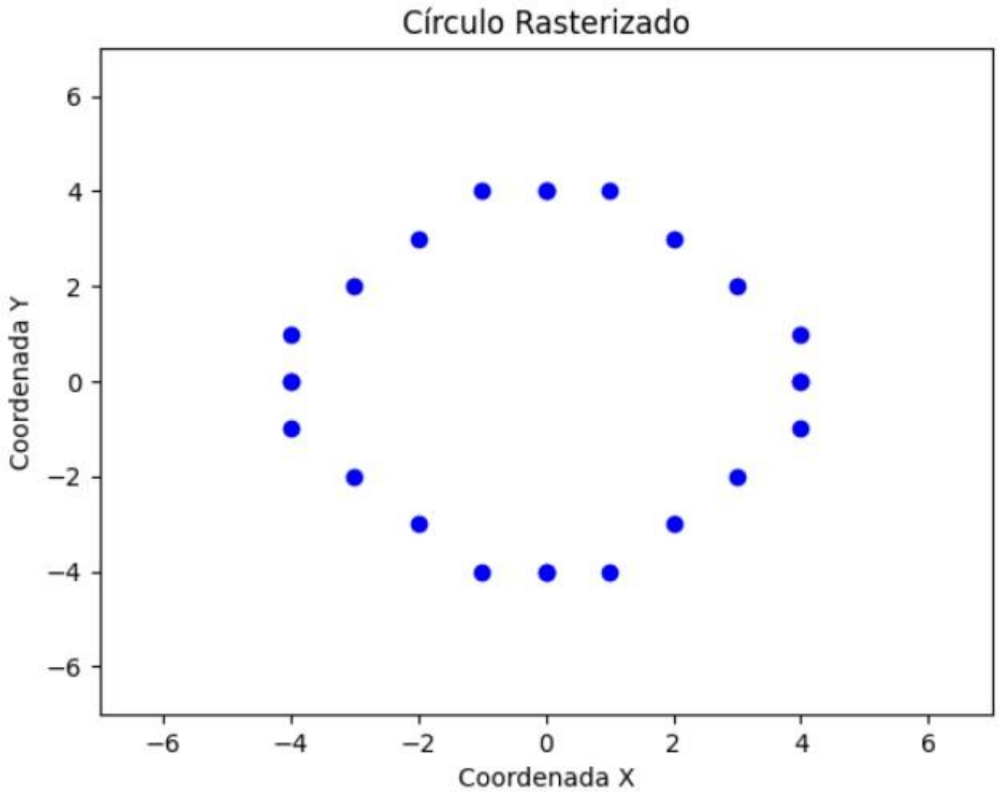
# Solução 2 - Algoritmo em python

Resultado:



# Solução 2 - Algoritmo em python

Se eu quiser deixar o círculo mais redondo? Teremos que adicionar mais valores para x, do tipo decimal.



## Solução 2 - Algoritmo em python

**Parte I do código (continua próximo slide)**

Se eu quiser deixar o círculo mais redondo? teremos que adicionar mais valores para x, do tipo decimal. Ex: 0.1 a 0.1

```
import math
import matplotlib.pyplot as plt

# Defina o raio do círculo
raio = 4

# Inicialize a coordenada x
x = 0.0  # Agora começa com um número decimal

# Lista para armazenar os pontos
pontos = []

# Calcule o limite para a varredura no eixo x
limite = raio * math.cos(math.radians(45))

# Loop enquanto x for menor que o limite
while x <= limite:
    # Calcule a coordenada y correspondente usando a equação do círculo
    y = round(math.sqrt(raio * raio - x * x), 2)  # Mantém valores decimais

    # Adicione os pontos à lista considerando a simetria do círculo
    pontos.extend([
        (round(x, 2), y), (y, round(x, 2)), (y, -round(x, 2)), (round(x, 2), -y),
        (-round(x, 2), -y), (-y, -round(x, 2)), (-y, round(x, 2)), (-round(x, 2), y)
    ])

    # Incremente x para avançar na varredura (agora em passos menores)
    x += 0.1  # Reduzindo o passo para melhorar a suavidade
```

## Solução 2 - Algoritmo em python

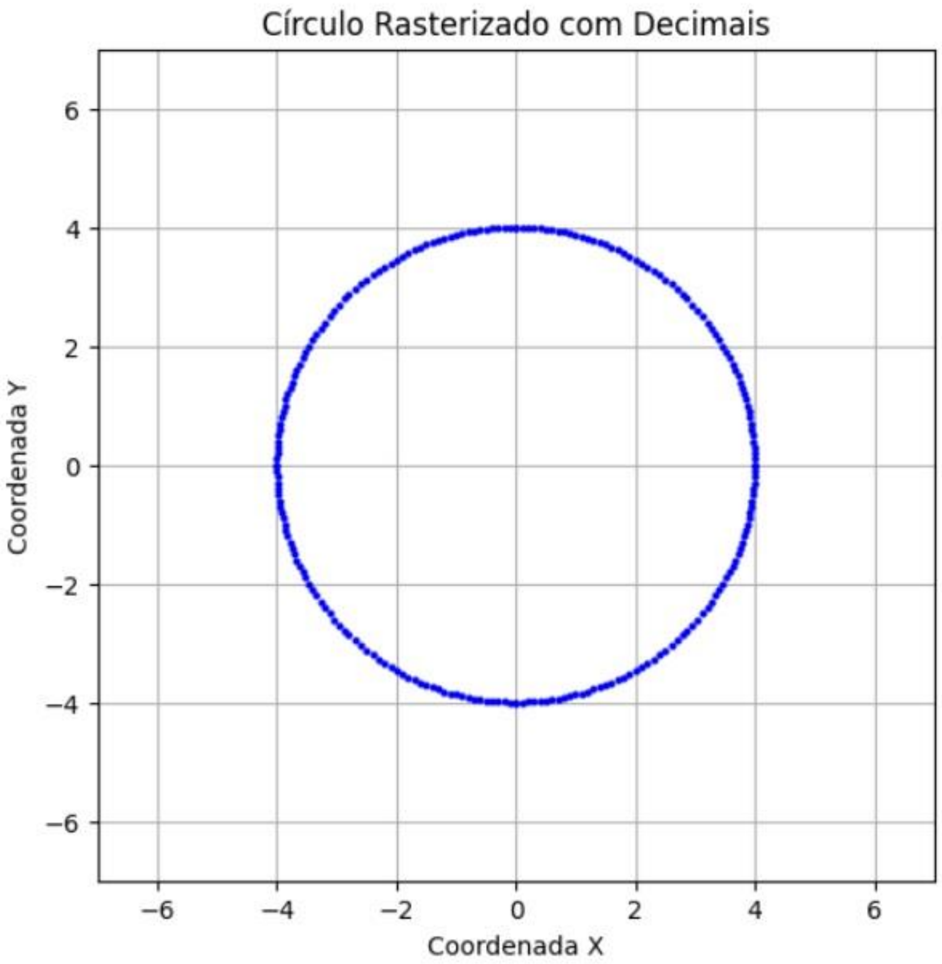
**Parte II do código**

Se eu quiser deixar o círculo mais redondo? teremos que adicionar mais valores para x, do tipo decimal. Ex: 0.1 a 0.1

```
# Separe as coordenadas x e y para plotagem
coordenadas_x, coordenadas_y = zip(*pontos)

# Plote os pontos
plt.figure(figsize=(6,6))
plt.plot(coordenadas_x, coordenadas_y, 'bo', markersize=2)  # Pontos menores para
melhor visualização

# Ajuste os limites dos eixos para -7 a 7
plt.xlim(-7, 7)
plt.ylim(-7, 7)

# Adicione grade e rótulos
plt.grid(True)
plt.title("Círculo Rasterizado com Decimais")
plt.xlabel("Coordenada X")
plt.ylabel("Coordenada Y")

# Mostre o gráfico
plt.show()
```

**Solução 2 - Algoritmo em python**

**Resultado**



# Algoritmo do ponto médio

- O grande problema da solução anterior é o fato de precisarmos realizar a operação de **raiz quadrada** para cada valor calculado:
  - Que possui um alto custo computacional;
- Além disso, trabalhamos também com **arredondamento/truncamento**, bem como valores em **ponto flutuante**, que também têm um **alto custo de processamento**;
- O **Algoritmo do Ponto médio**, criado por Pitteway e Van Aken, é uma solução que utiliza somente operações de **soma** e **subtração** envolvendo **números inteiros**.

# Algoritmo do ponto médio

- Funcionamento:
  1. A partir de um círculo de raio **r**, desenhamos o primeiro ponto na tela **(0,r)**;
  2. Calculamos o valor inicial do parâmetro de decisão **p**:

$$p_1 = 1 - r$$

  3. Para cada ponto **x**, realizar então o seguinte teste:
     - Se $p_k < 0$, o próximo ponto do círculo será $(x_k+1, y_k)$ e o próximo valor de **p** será:

$$p_{k+1} = p_k + 2x_{k+1} + 1$$

     - Se $p_k >= 0$, o próximo ponto do círculo será $(x_k+1, y_k-1)$ e o próximo valor de **p** será:

$$p_{k+1} = p_k + 2x_{k+1} - 2y_{k+1} + 1$$

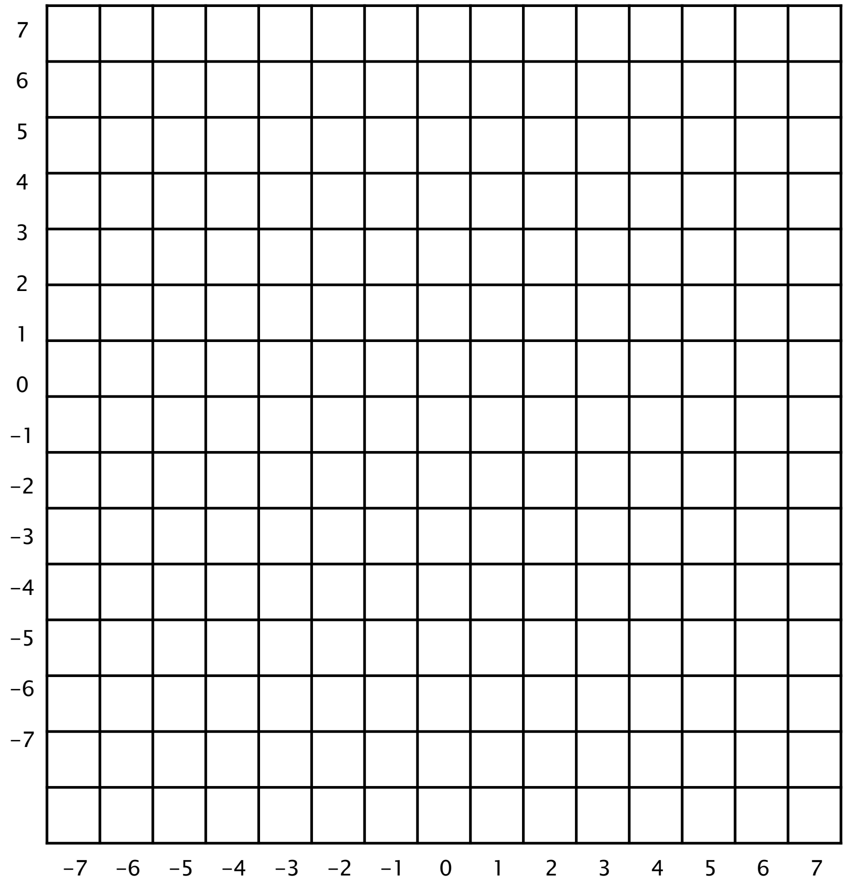
# Algoritmo do ponto médio

- Funcionamento:
  4. Repetir o passo 3 até que **$x>=y$**;
  5. Calcular os outros pontos dos 7 octantes por **simetria**.

# Algoritmo do ponto médio

- Exemplo: rasterizar um círculo de raio = 5

| x | y | $p_k$ |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

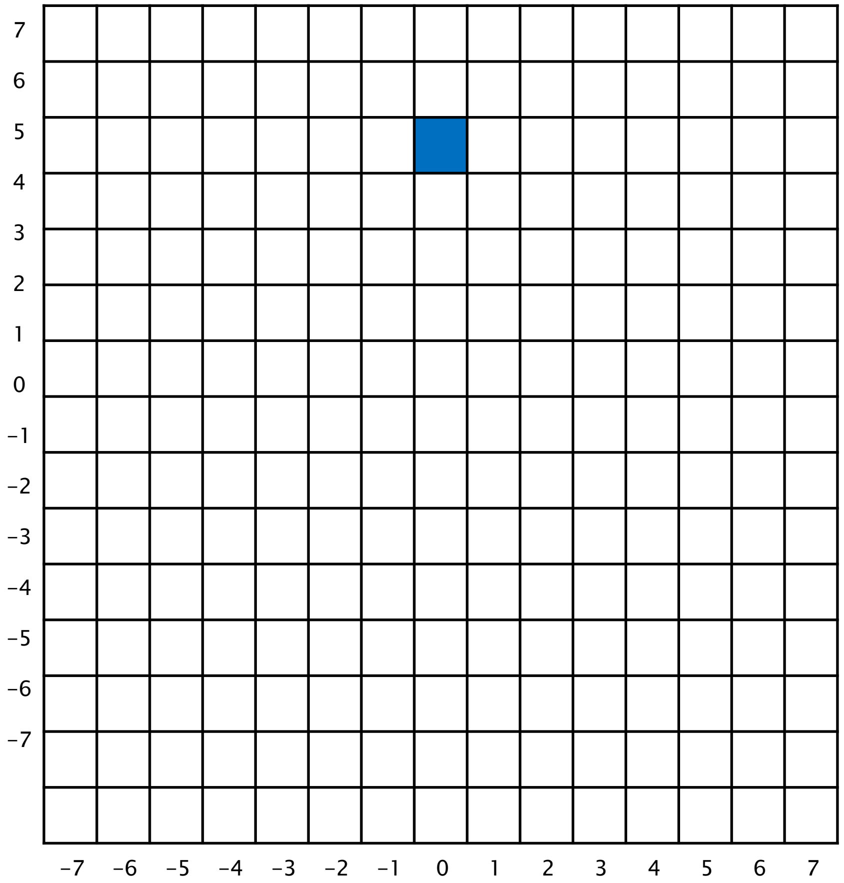


# Algoritmo do ponto médio

- Exemplo: rasterizar um círculo de raio = 5

| x | y | $p_k$ |
|---|---|---|
| 0 | 5 | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Primeiramente, com o raio é 5, desenhamos o ponto (0,5) na tela



# Algoritmo do ponto médio

- Exemplo: rasterizar um círculo de raio = 5

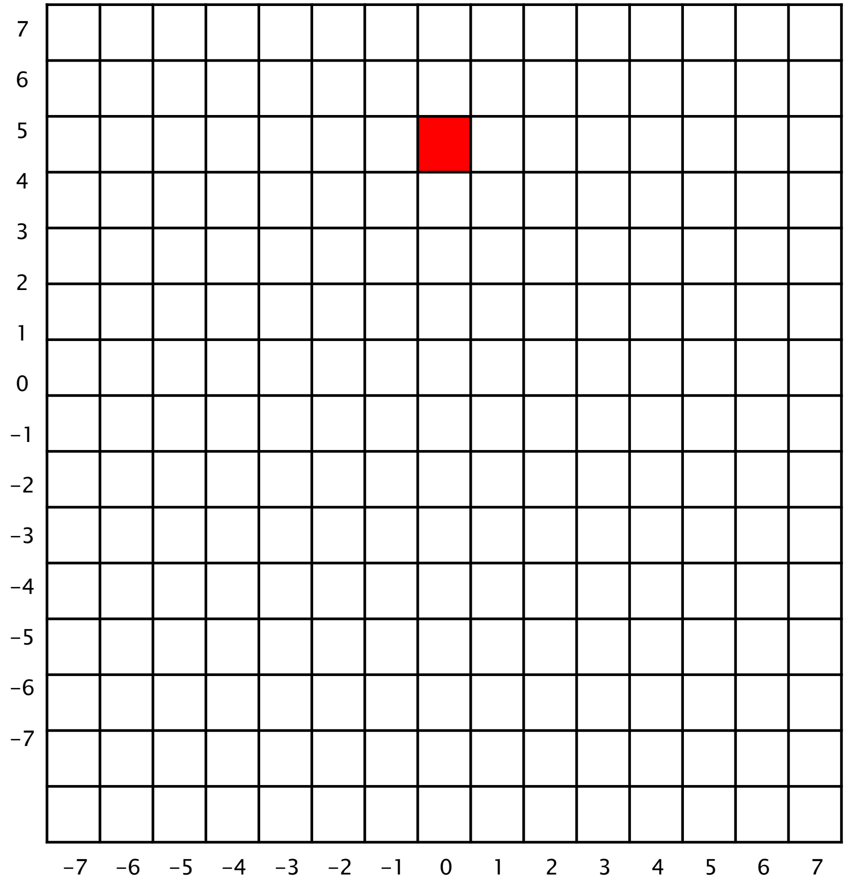
| x | y | $p_k$ |
|---|---|---|
| 0 | 5 | **-4** |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

E calculamos o valor inicial de $p_k$:

$P_k = 1 - r$

$P_k = 1 - 5$

$P_k = -4$



# Algoritmo do ponto médio

- Exemplo: rasterizar um círculo de raio = 5

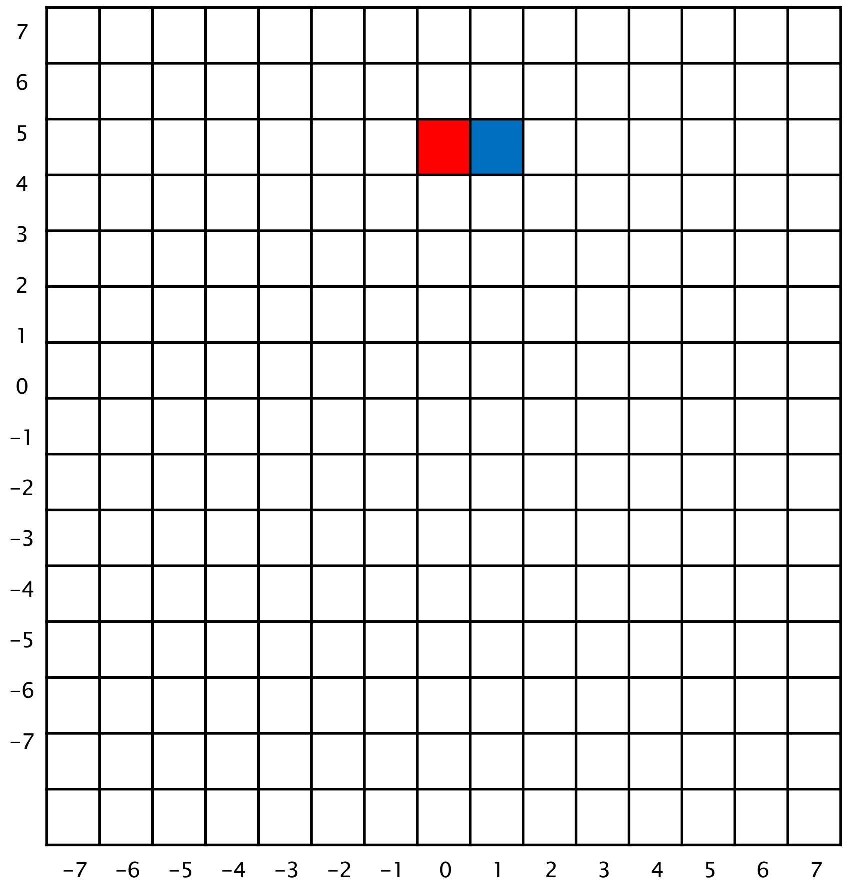
| x | y | $p_k$ |
|---|---|---|
| 0 | 5 | −4 |
| **1** | **5** | **-1** |
| | | |
| | | |
| | | |
| | | |
| | | |

Como $p_k < 0$, o próximo ponto a ser pintado é o ponto **$(x_k+1, y_k)$** e o próximo valor de $p_k$ será:

$p_{k+1} = p_k + 2x_{k+1} + 1$

$p_{k+1} = -4 + 2.1 + 1$

$p_{k+1} = -1$



# Algoritmo do ponto médio

- Exemplo: rasterizar um círculo de raio = 5

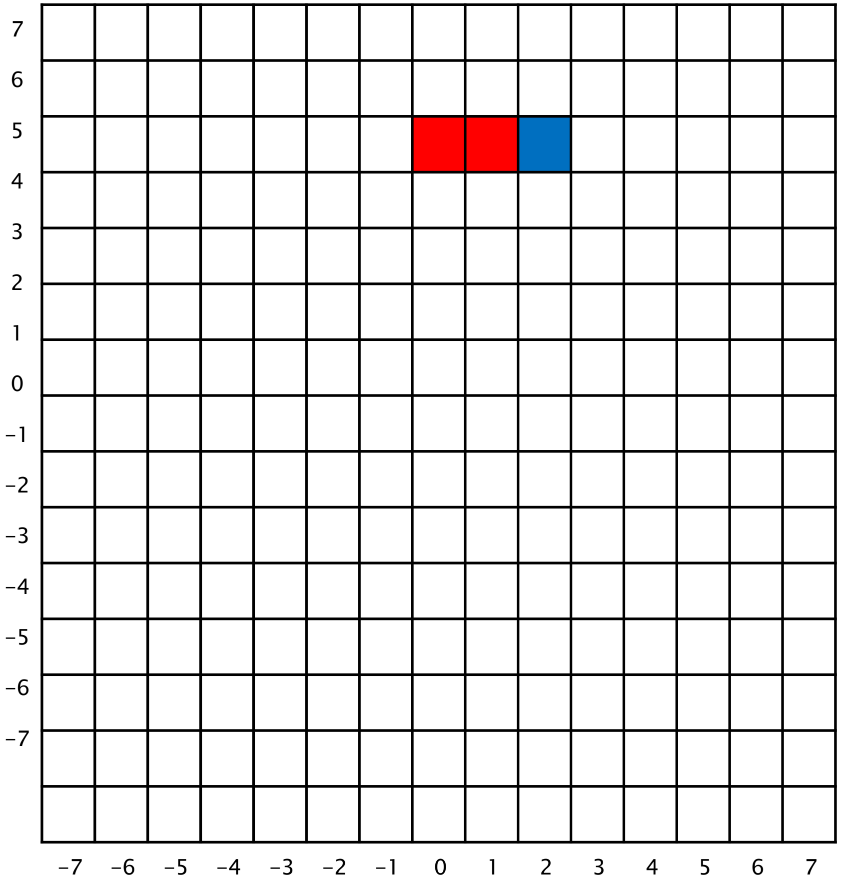
| x | y | $p_k$ |
|---|---|---|
| 0 | 5 | −4 |
| 1 | 5 | −1 |
| **2** | **5** | 4 |
| | | |
| | | |
| | | |
| | | |

Como $p_k < 0$, o próximo ponto a ser pintado é o ponto **$(x_k+1, y_k)$** e o próximo valor de $p_k$ será:

$p_{k+1} = p_k + 2x_{k+1} + 1$

$p_{k+1} = -1 + 2.2 + 1$

$p_{k+1} = 4$



# Algoritmo do ponto médio

- Exemplo: rasterizar um círculo de raio = 5

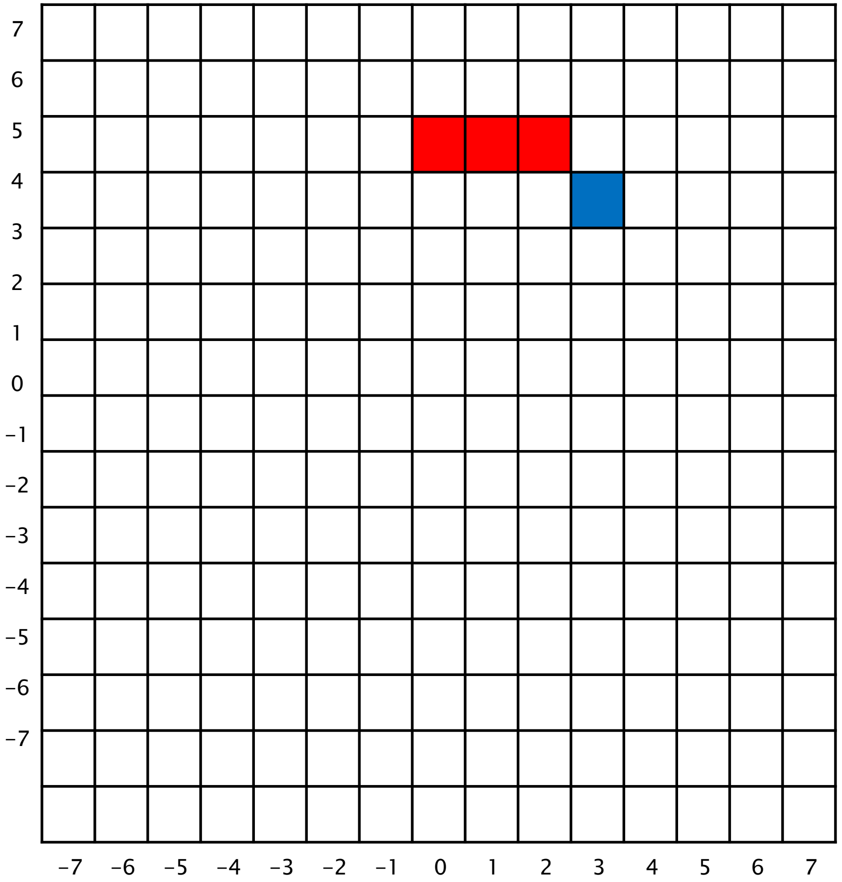
| x | y | $p_k$ |
|---|---|---|
| 0 | 5 | −4 |
| 1 | 5 | −1 |
| 2 | 5 | 4 |
| **3** | **4** | **3** |
| | | |
| | | |
| | | |

Como $p_k >= 0$, o próximo ponto a ser pintado é o ponto **$(x_k+1, y_k-1)$** e o próximo valor de $p_k$ será:

$p_{k+1} = p_k + 2x_{k+1} - 2y_{k+1} + 1$

$p_{k+1} = 4 + 2.3 - 2.4 + 1$

$p_{k+1} = 3$



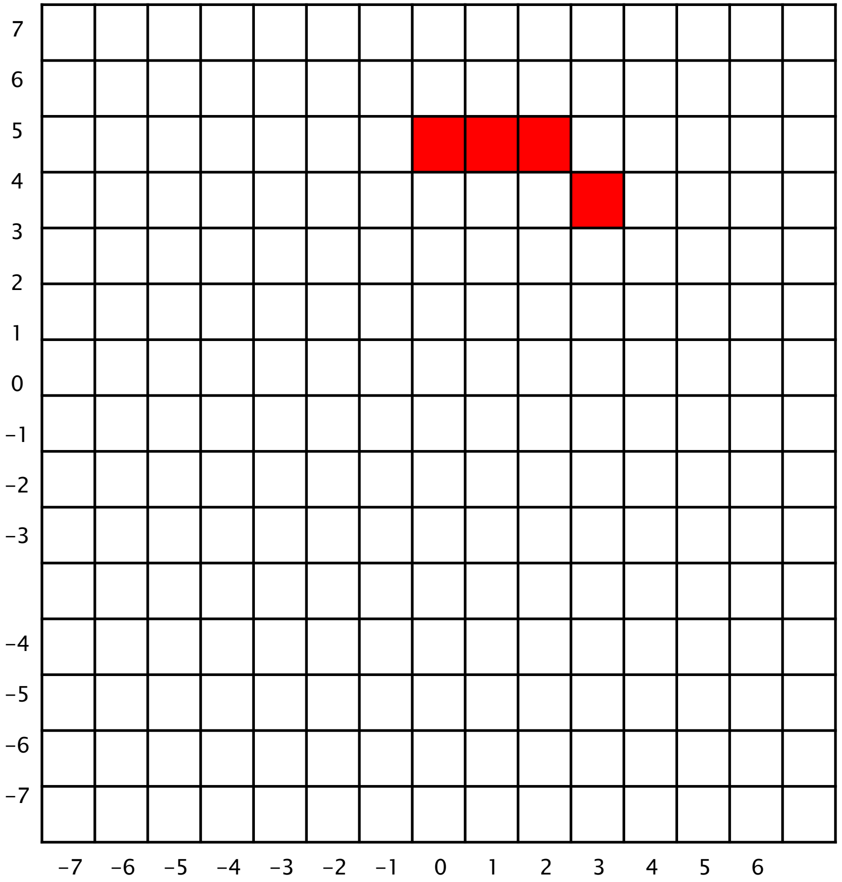
# Algoritmo do ponto médio

- Exemplo: rasterizar um círculo de raio = 5

| x | y | $p_k$ |
|---|---|---|
| 0 | 5 | –4 |
| 1 | 5 | –1 |
| 2 | 5 | 4 |
| 3 | 4 | 3 |
| 4 | 3 | |
| | | |
| | | |

Como $p_k >= 0$, o próximo ponto a ser pintado é o ponto **$(x_k+1, y_k-1)$**

Assim, chegamos a $x>y$, logo, o algoritmo termina,e agora precisamos refletor esses resultados para os outros octantes.

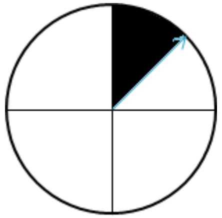


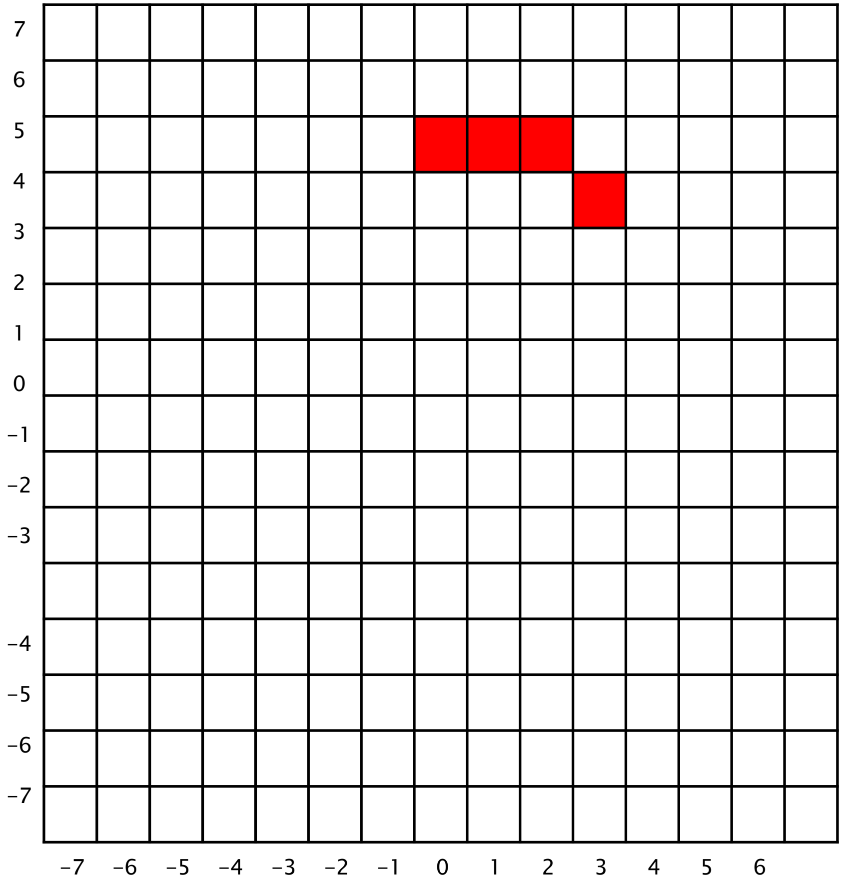
# Algoritmo do ponto médio

- Exemplo: rasterizar um círculo de raio = 5

| (x,y) |
|---|
| 0,5 |
| 1,5 |
| 2,5 |
| 3,4 |

(x,y)
calculado





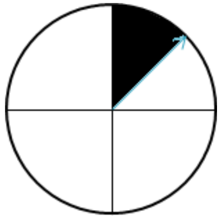
# Algoritmo do ponto médio

- Exemplo: rasterizar um círculo de raio = 5

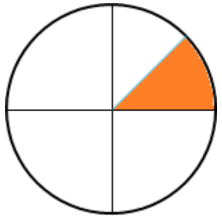
| (x,y) |
|---|
| 0,5 |
| 1,5 |
| 2,5 |
| 3,4 |

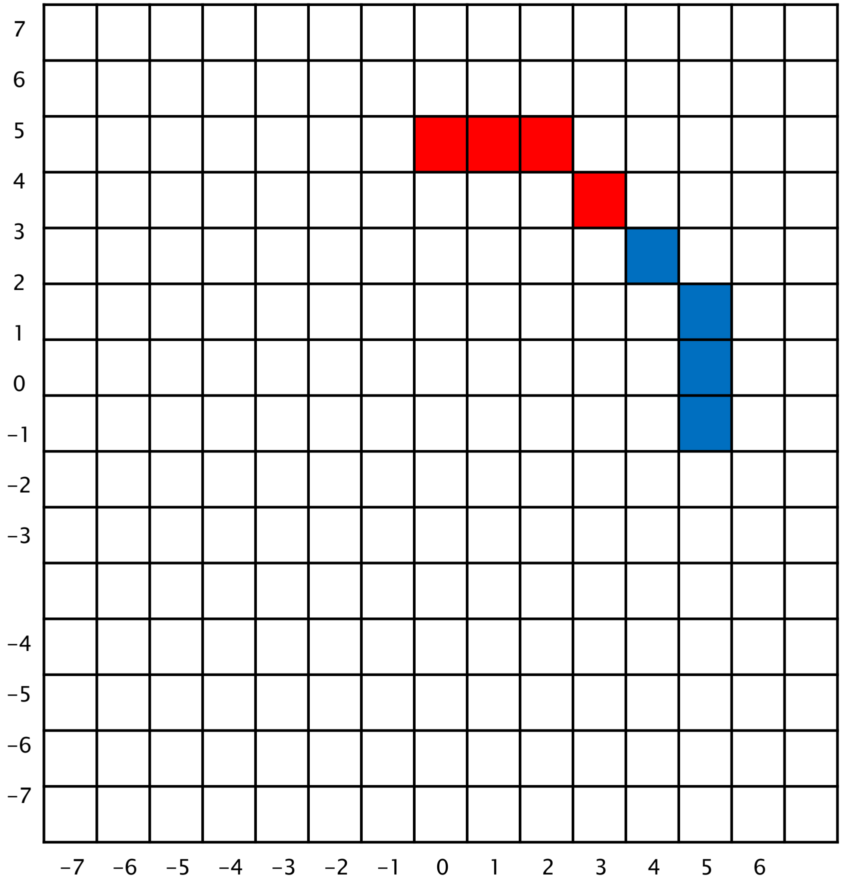
| (y,x) |
|---|
| 5,0 |
| 5,1 |
| 5,2 |
| 4,3 |

(x,y)
calculado



(y,x)





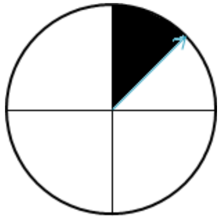
# Algoritmo do ponto médio

- Exemplo: rasterizar um círculo de raio = 5

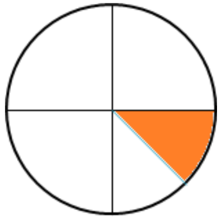
| (x,y) |
|---|
| 0,5 |
| 1,5 |
| 2,5 |
| 3,4 |

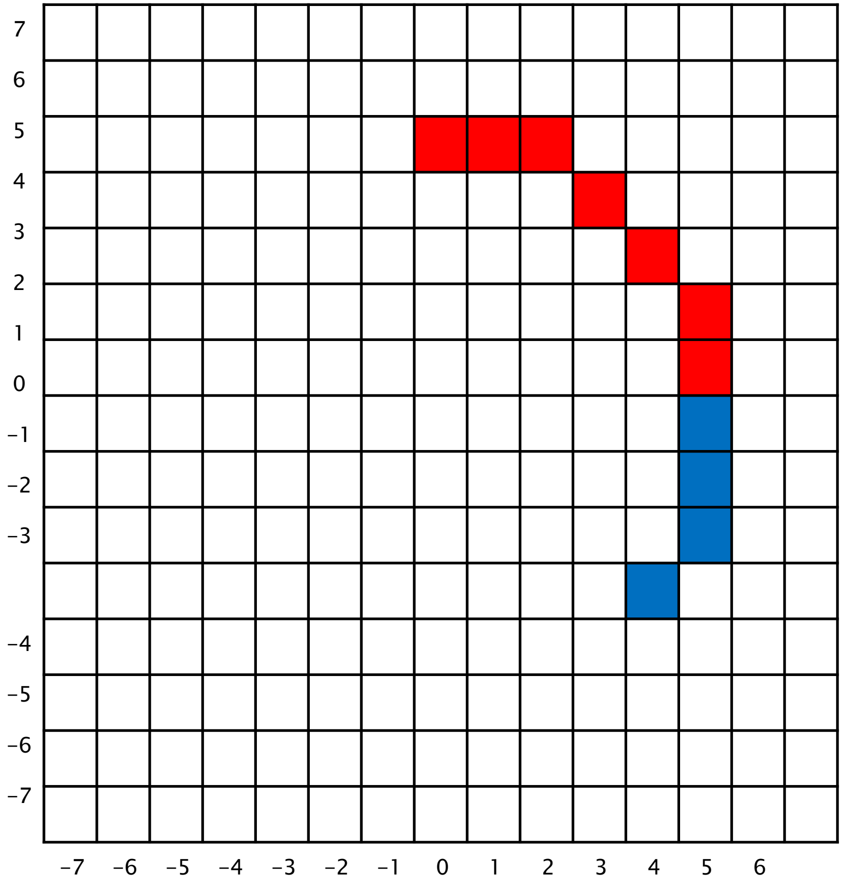
| (y,-x) |
|---|
| 5,0 |
| 5,−1 |
| 5,−2 |
| 4,−3 |

(x,y)
calculado



(y,−x)





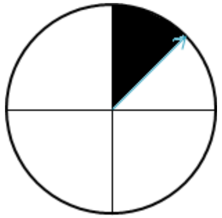
# Algoritmo do ponto médio

- Exemplo: rasterizar um círculo de raio = 5

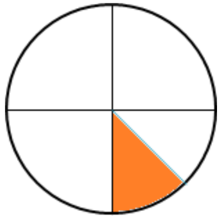
| (x,y) |
|---|
| 0,5 |
| 1,5 |
| 2,5 |
| 3,4 |

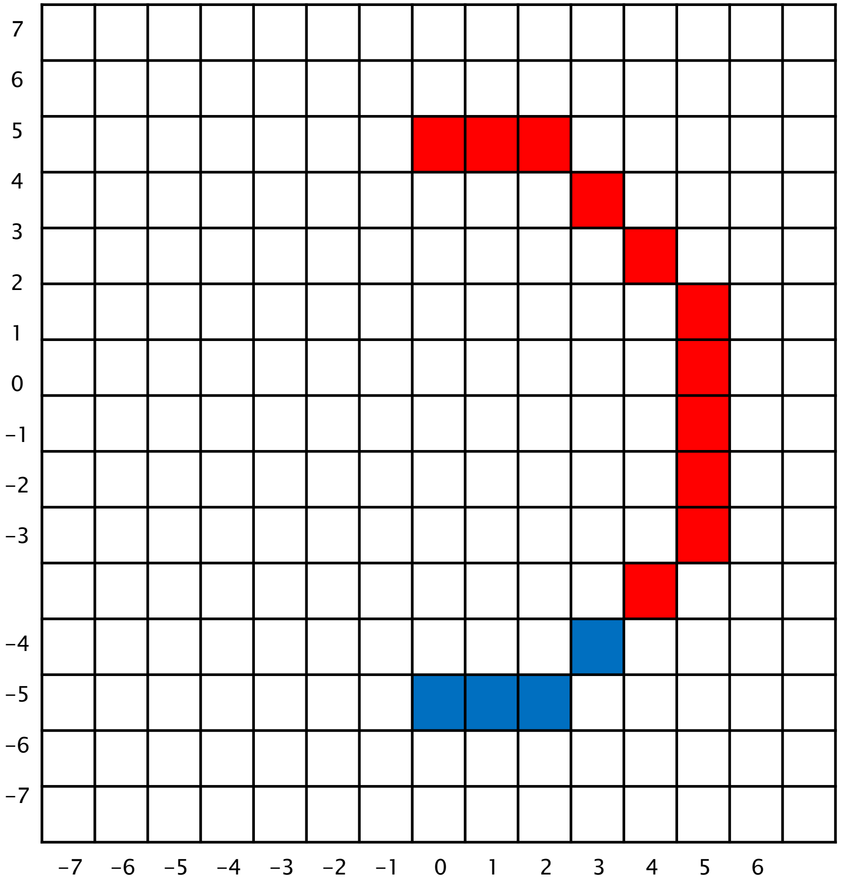
| (x,-y) |
|---|
| 0,–5 |
| 1,–5 |
| 2,–5 |
| 3,–4 |

(x,y)
calculado



(x,–y)





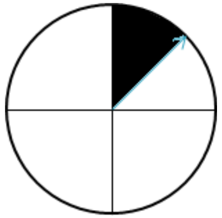
# Algoritmo do ponto médio

- Exemplo: rasterizar um círculo de raio = 5

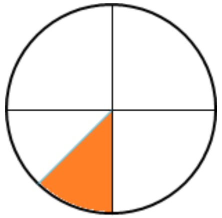
| (x,y) |
|---|
| 0,5 |
| 1,5 |
| 2,5 |
| 3,4 |

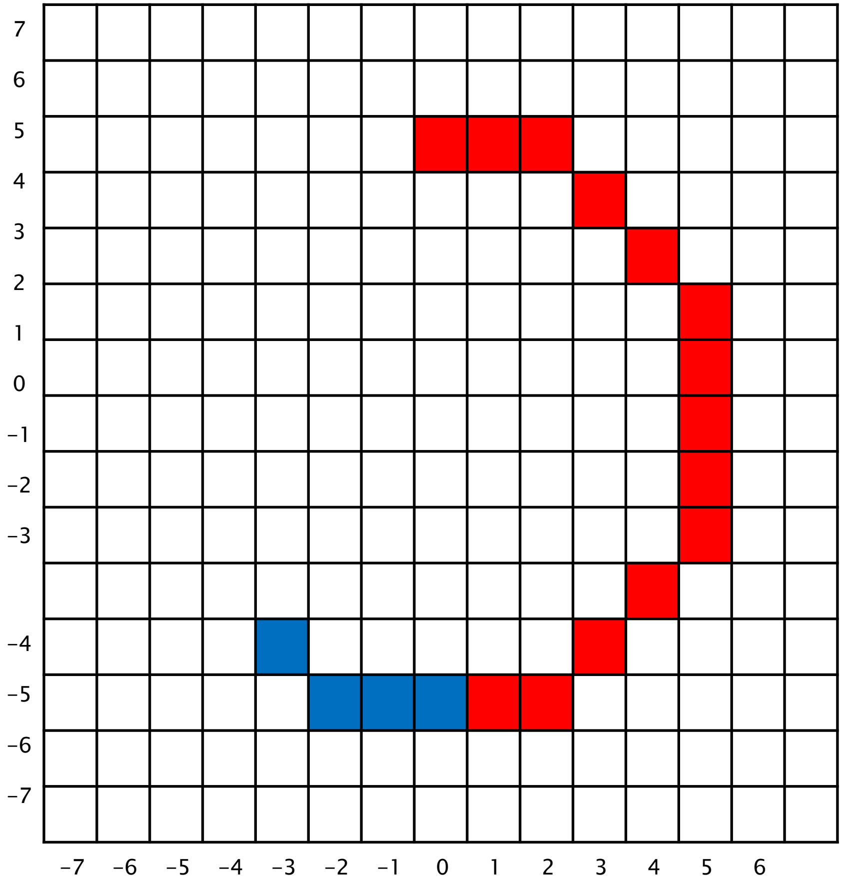
| (-x,-y) |
|---|
| 0,−5 |
| −1,−5 |
| −2,−5 |
| −3,−4 |

(x,y)
calculado



(−x,−y)



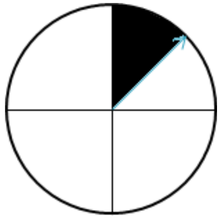


# Algoritmo do ponto médio

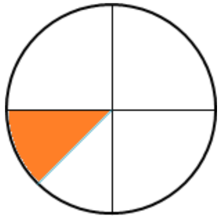
- Exemplo: rasterizar um círculo de raio = 5

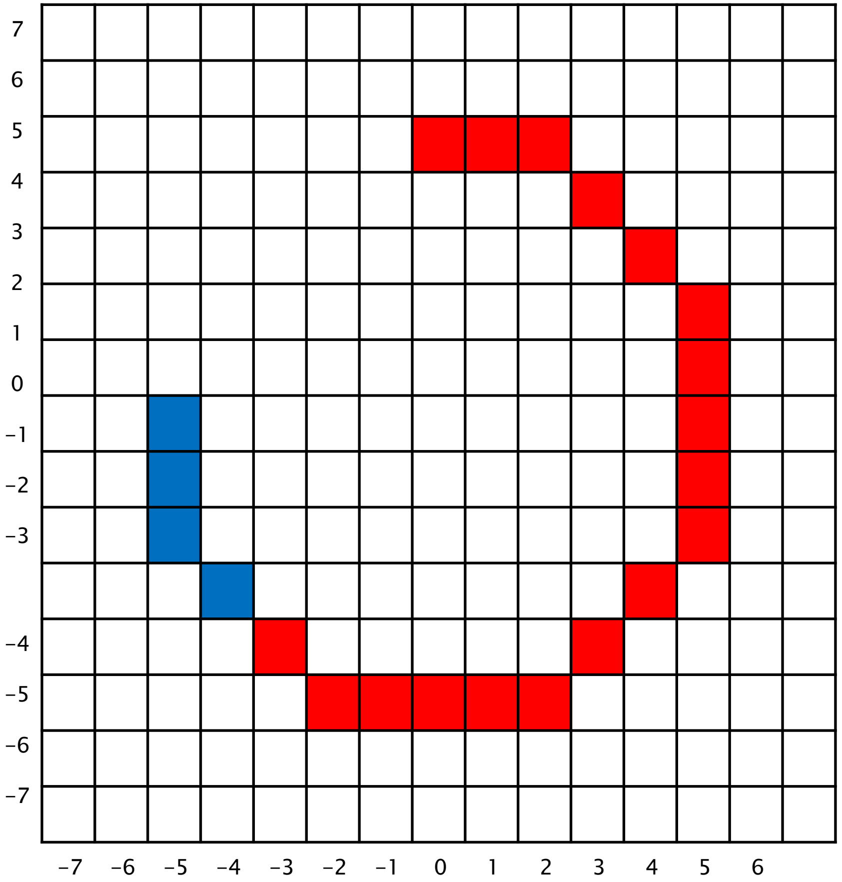
| (x,y) | (-y,-x) |
|---|---|
| 0,5 | −5,0 |
| 1,5 | −5,−1 |
| 2,5 | −5,−2 |
| 3,4 | −4,−3 |

(x,y)
calculado



(−y,−x)





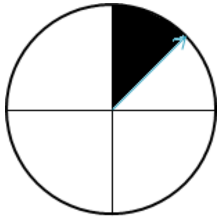
# Algoritmo do ponto médio

- Exemplo: rasterizar um círculo de raio = 5

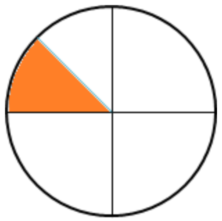
| (x,y) |
|---|
| 0,5 |
| 1,5 |
| 2,5 |
| 3,4 |

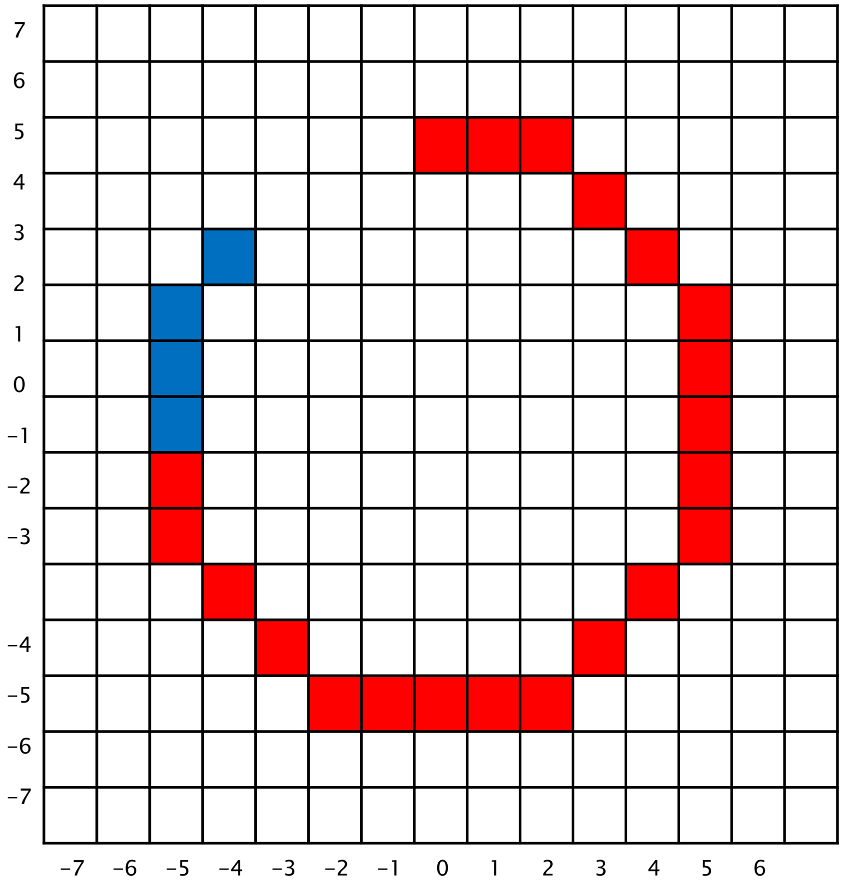
| (-y,x) |
|---|
| −5,0 |
| −5,1 |
| −5,2 |
| −4,3 |

(x,y)
calculado



(−y,x)



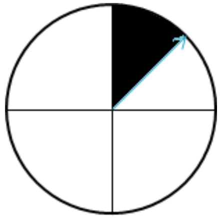


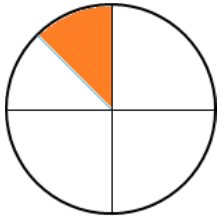
# Algoritmo do ponto médio

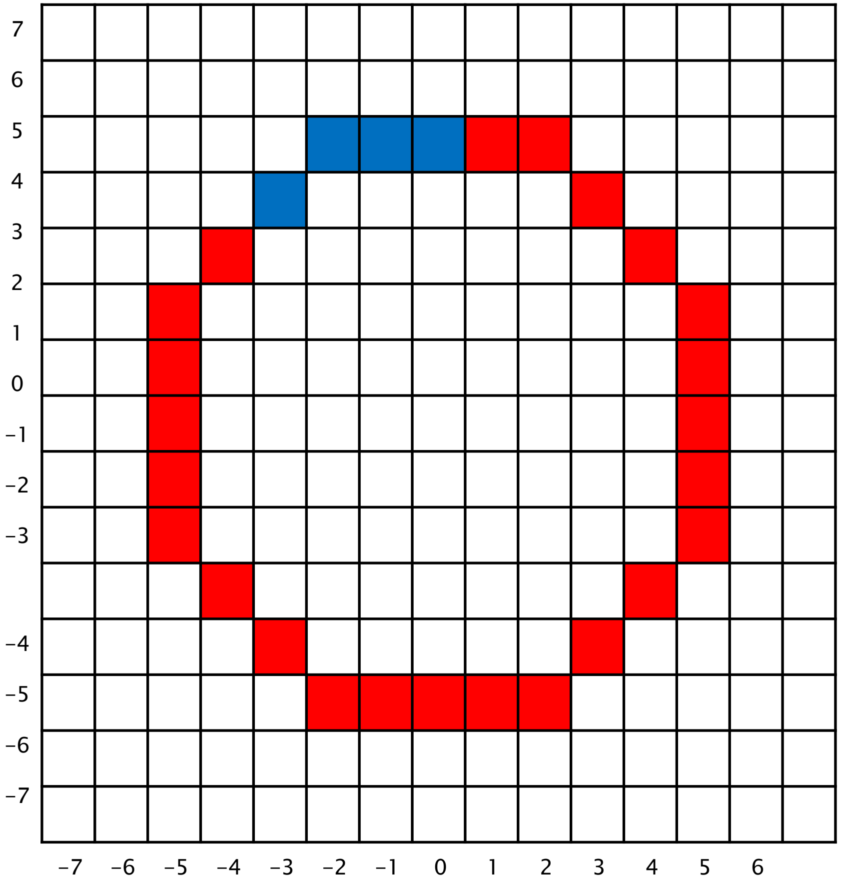
- Exemplo: rasterizar um círculo de raio = 5

| (x,y) |
|---|
| 0,5 |
| 1,5 |
| 2,5 |
| 3,4 |

| (-x,y) |
|---|
| 0,5 |
| −1,5 |
| −2,5 |
| −3,4 |

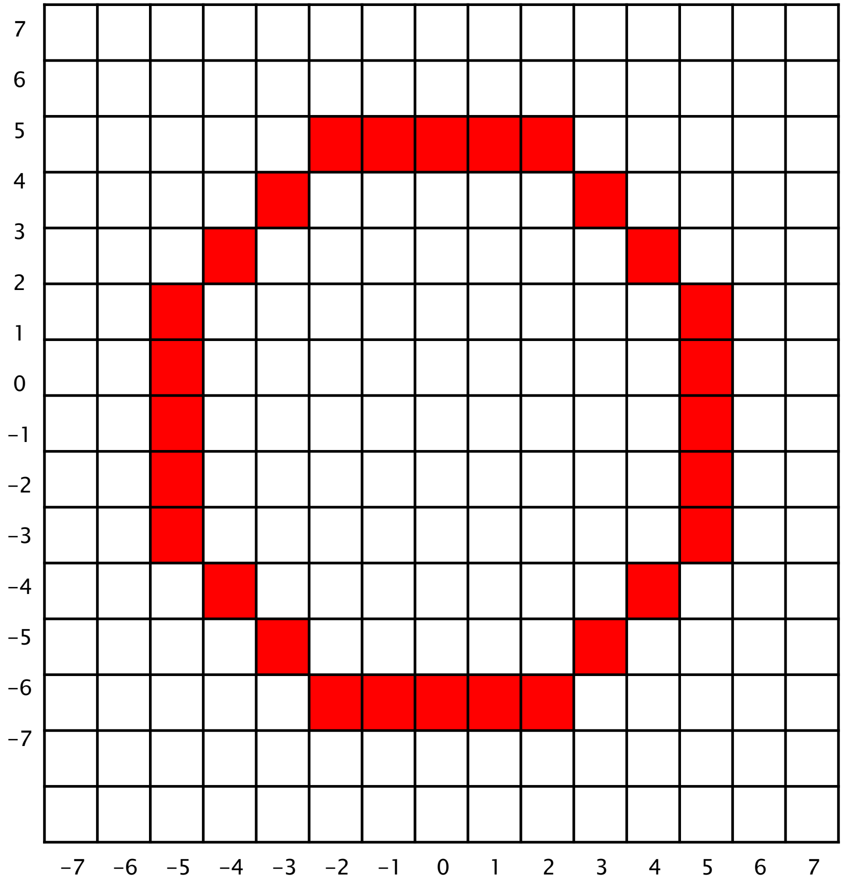


# Algoritmo do ponto médio

- Exemplo: rasterizar um círculo de raio = 5



# Algoritmo do ponto médio

```
import matplotlib.pyplot as plt

def draw_circle(radius):
    # Inicializa as coordenadas iniciais
    x = 0
    y = radius
    p = 1 - radius  # Calcula o parâmetro de decisão inicial

    # Lista para armazenar os pontos do círculo
    points = []

    # Desenha o primeiro ponto
    points.append((x, y))

    # Itera sobre os pontos do círculo
    while x < y:
        x += 1
        # Testa o parâmetro de decisão
        if p < 0:
            p += 2 * x + 1
        else:
            y -= 1
            p += 2 * x - 2 * y + 1
        # Adiciona os pontos simétricos às listas
        points.append((x, y))
        points.append((x, -y))
        points.append((-x, y))
        points.append((-x, -y))
        points.append((y, x))
        points.append((y, -x))
        points.append((-y, x))
        points.append((-y, -x))

    return points
```

```
# Defina o raio do círculo
radius = 5

# Desenha o círculo
circle_points = draw_circle(radius)

# Extrai as coordenadas x e y dos pontos
x_coords, y_coords = zip(*circle_points)

# Cria o gráfico do círculo
plt.figure()
plt.plot(x_coords, y_coords, 'bo')  # 'bo' para pontos azuis
plt.gca().set_aspect('equal', adjustable='box')  # Define o aspecto igual para evitar
a distorção
plt.title("Círculo Gerado pelo Algoritmo do Ponto Médio")
plt.xlabel("Coordenada X")
plt.ylabel("Coordenada Y")
plt.grid(True)  # Adiciona uma grade ao gráfico
plt.show()
```

# Referências e material de apoio

Material do Professor Guilherme Chagas Kurtz, 2023.

GOMES, Jonas; VELHO, Luiz. Computação gráfica. Rio de Janeiro: Impa, 1998.
HEARN, Donald; Baker, M. Pauline. Computer grafhics: C version. London: Prentice Hall, 1997.
HETEM JUNIOR, Annibal. Computação gráfica. Rio de Janeiro, RJ: LTC, 2006. 161 p. (Coleção Fundamentos de Informática).
HILL Jr, Francis S. Computer graphics using open GL. New Jersey: Prentice Hall, 2001.
WATT, Alan. 3D computer graphics. Harlow: Addison-Wesley, 2000
Material Prof. Guilherme Chagas Kurtz, 2023.

# Thank you for your attention!!


Email: **andre.flores@ufn.edu.br**

**Santa Maria – RS  2025**