UNIVERSIDADE FEDERAL DE RORAIMA CENTRO DE CIÊNCIA E TECNOLOGIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
DCC703 - COMPUTAÇÃO GRÁFICA (2024.2)
Prof. LUCIANO FERREIRA SILVA

Nome: Marcia Gabrielle Bonifácio De Oliveira - 2020011319

TRABALHO 2

INTRODUÇÃO

Este trabalho apresenta diferentes algoritmos utilizados para a geração de circunferências em computação gráfica. Dentre os métodos abordados, estão a equação paramétrica, o método incremental com simetria e o algoritmo de Bresenham. Cada um desses métodos possui características específicas que os tornam mais ou menos eficientes dependendo da aplicação desejada.

2. DESCRIÇÃO DOS ALGORITMOS

2.1 EQUAÇÃO PARAMÉTRICA

O **método da Equação Paramétrica** para desenho de circunferências baseia-se diretamente nas fórmulas de seno e cosseno. Para cada ângulo θ, convertemos esse ângulo em radianos e obtemos as coordenadas:

$$x = xc + r.\cos(t)$$

$$y = yc + r.\text{sen}(t)$$

onde (xc,yc) é o centro do círculo e r é o seu raio. Em implementações simples, percorre-se θ de 0 a 360 graus (ou 0 a 2π radianos) em pequenos incrementos, e então arredonda-se x(θ) e y(θ) para ativar o pixel apropriado na tela.

Esse método é **simples de entender e de codificar**, mas faz **uso intensivo de operações em ponto flutuante** (funções trigonométricas e arredondamentos) a cada incremento angular. Além disso, ele não explora simetrias do círculo, o que o torna menos eficiente que outros algoritmos mais avançados.

```python
def parametric_circle(xc, yc, r):
    points = []
    for t in range(0, 360):  # Percorre os ângulos de 0 a 360 graus
        rad = np.radians(t)  # Converte para radianos
        x = xc + round(r * np.cos(rad))  # Equação paramétrica para x
        y = yc + round(r * np.sin(rad))  # Equação paramétrica para y
        points.append((x, y))
    return points
```

2.3 MÉTODO INCREMENTAL COM SIMETRIA

O Método Incremental com Simetria explora o fato de que um círculo é simétrico em relação aos seus eixos e diagonais (octantes). Ao invés de calcular ponto a ponto em toda a circunferência, podemos calcular apenas 1/8 do círculo e espelhar esses pontos nos demais 7 octantes.

Para isso, define-se um pequeno incremento angular $\Delta\theta$, muitas vezes tomado como $1/r$. Começando em $\theta=0$ até $\theta=\pi/4$, calculamos $(\text{round}(r\cos\theta),\text{round}(r\sin\theta))$. Cada ponto encontrado é então reproduzido em todos os octantes do círculo, aproveitando as transformações.

Esse algoritmo é menos custoso que o paramétrico puro, pois, ao invés de 360 iterações (em graus) ou de 2π\piπ (em radianos), percorremos apenas um oitavo da circunferência, e usamos reflexão para obter o restante. Ainda assim, há chamadas a seno e cosseno, bem como arredondamentos.

```python
def incremental_circle_with_symmetry(xc, yc, r):

    points = []
    theta_step = 1 / r  # Incremento angular (1 / r)
    theta = 0

    # Calcula os pontos para 1/8 da circunferência
    while theta <= math.pi / 4:
        x = round(r * math.cos(theta))
        y = round(r * math.sin(theta))
        points.append((x, y))
        theta += theta_step

    # Espelha os pontos para os octantes restantes
    symmetric_points = []
    for x, y in points:
        symmetric_points.extend([
            (xc + x, yc + y), (xc - x, yc + y),
            (xc + x, yc - y), (xc - x, yc - y),
            (xc + y, yc + x), (xc - y, yc + x),
            (xc + y, yc - x), (xc - y, yc - x)
        ])

    return symmetric_points
```

2.4 MÉTODO BRESENHAM

O Método de Bresenham para círculos adapta a mesma lógica de decisão do Bresenham de retas, mas aplicada a uma circunferência. A ideia central é manter um parâmetro de decisão que indica se devemos mover no eixo x (eixo principal) ou se devemos mover no eixo x e no eixo y simultaneamente (descendo ou subindo), de modo a permanecer "próximo" ao círculo ideal.
Nesse caso, começamos de um ponto fácil de conhecer (por exemplo, (0,r) e foi definido o parâmetro $p=1-r$. A cada passo:

- Se $p<0$, movemos em x e atualizamos $p \leftarrow p+2x+3$.
- Caso contrário, movemos em x e também decrementamos y, atualizando $p \leftarrow p+2x-2y+5$.

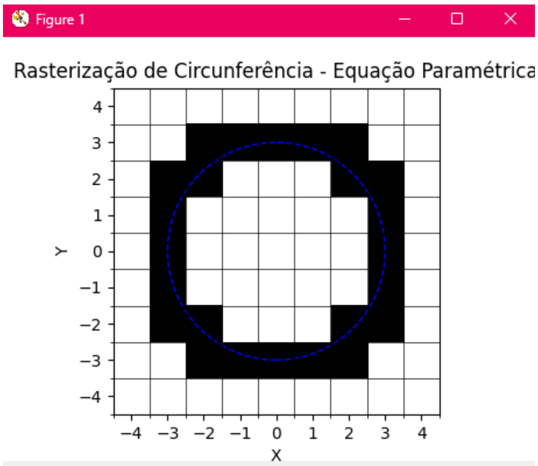
Como acontece no desenho de retas, o algoritmo de Bresenham se destaca por usar somente aritmética de inteiros, evitando funções trigonométricas ou multiplicações custosas. Ele também explora a simetria do círculo em oito octantes, desenhando apenas 1/8 e refletindo os pixels para os demais octantes (armazenando cada ponto em oito posições diferentes).

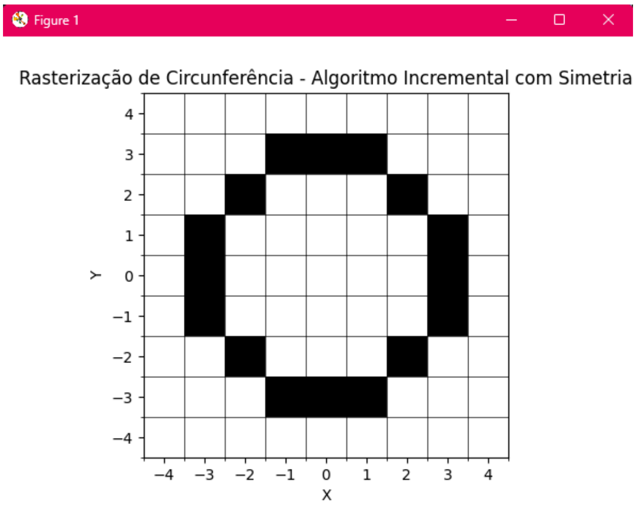
```python
def bresenham_circle(xc, yc, r):

    x = 0
    y = r
    p = 1 - r  # Parâmetro de decisão inicial

    points = []

    while x <= y:
        # Adiciona os pontos dos 8 octantes
        points.extend([
            (xc + x, yc + y), (xc - x, yc + y), (xc + x, yc - y), (xc - x, yc - y),
            (xc + y, yc + x), (xc - y, yc + x), (xc + y, yc - x), (xc - y, yc - x)
        ])

        if p < 0:
            p = p + 2 * x + 3
        else:
            p = p + 2 * x - 2 * y + 5
            y -= 1

        x += 1

    return points
```

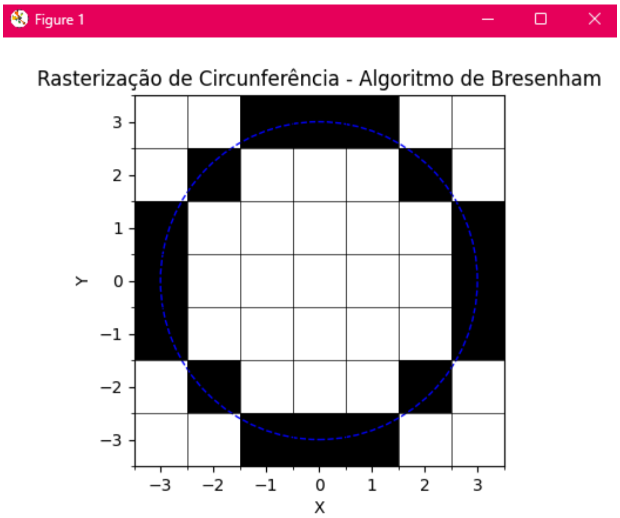
3. COMPARAÇÃO DOS ALGORITMOS

Círculo Pequeno:

Exemplo: (xc,yc)=(0,0), r=3.

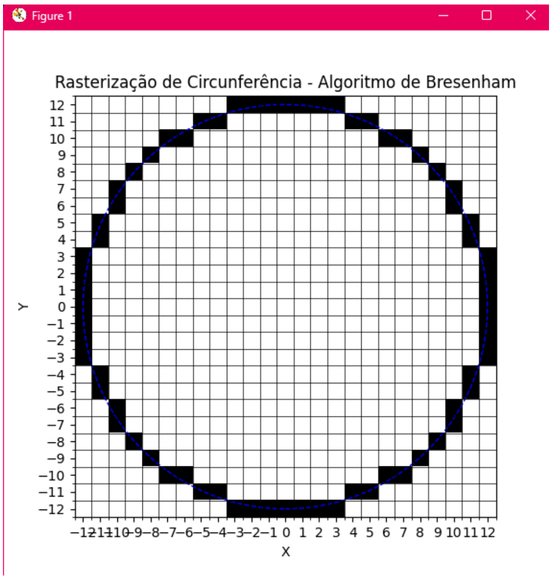


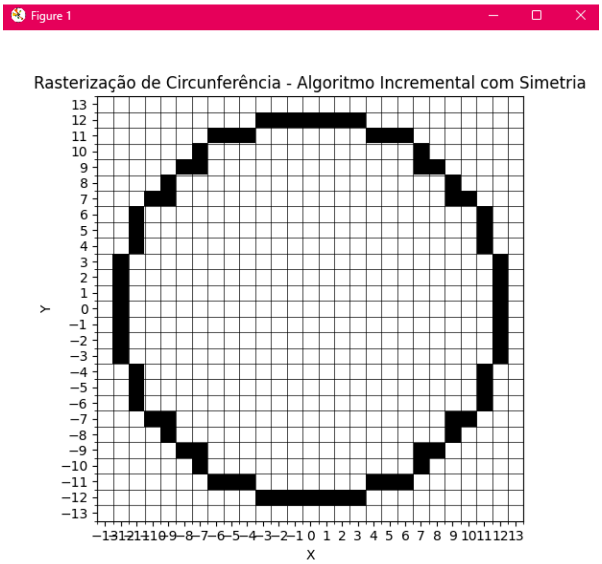


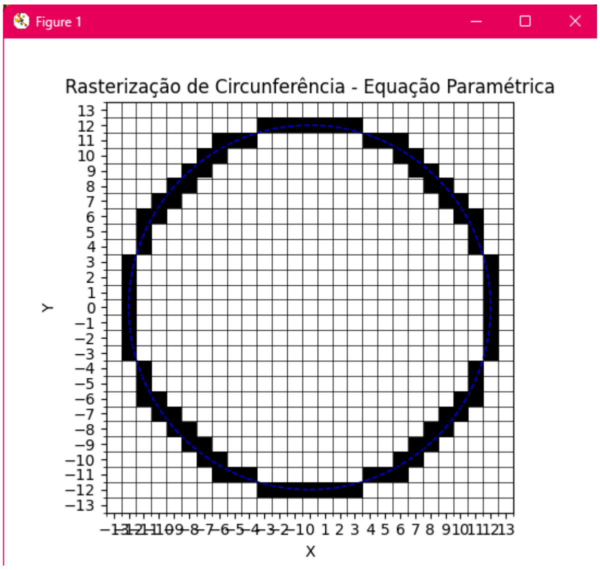


Círculo Médio

Exemplo: (xc,yc)=(0,0), r=12.

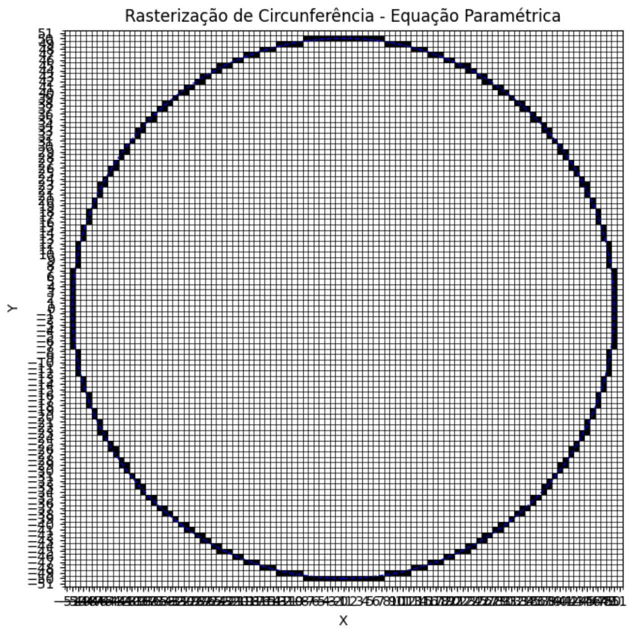


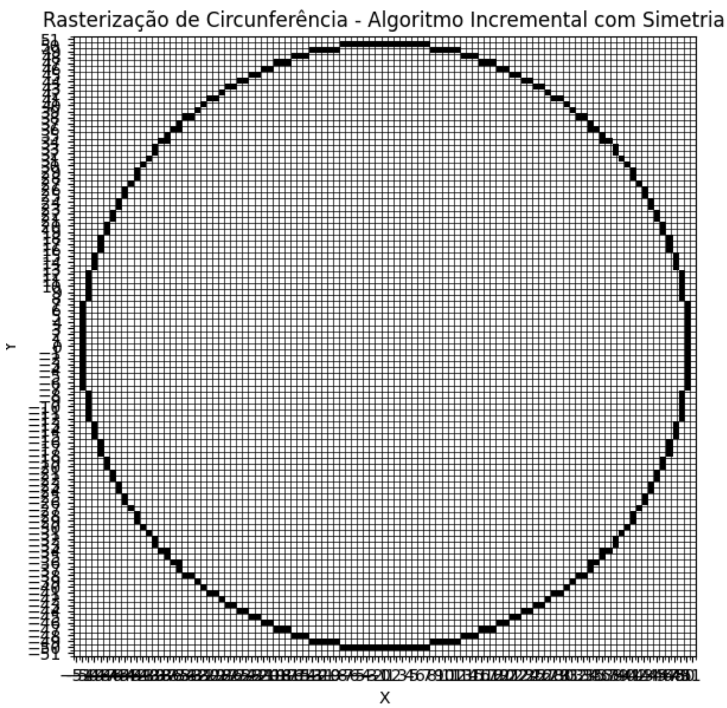


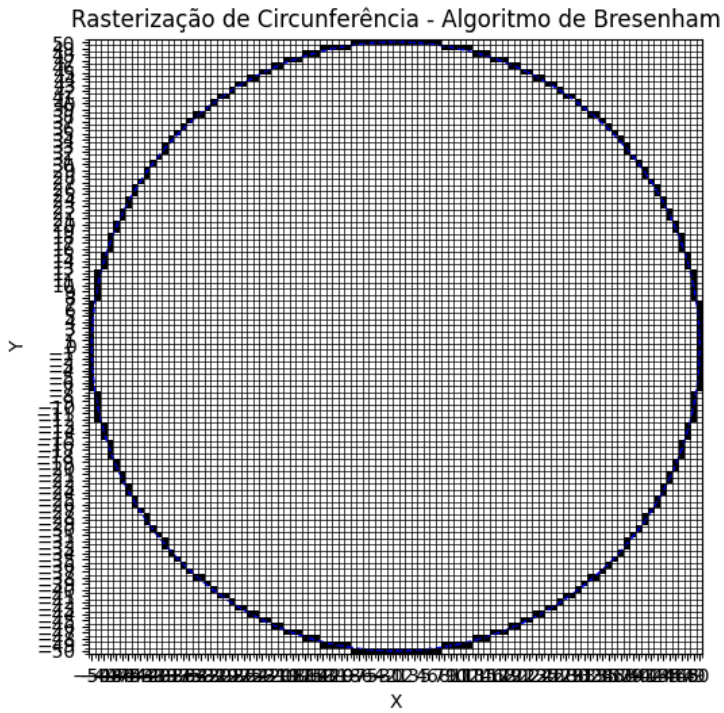


Círculo Grande

- **Exemplo: (xc,yc)=(0,0), r=50.**






Rasterização de Circunferência - Algoritmo de Bresenham
Y
X