



IGCE – Instituto de Geociências e Ciências Exatas
DEMAC – Departamento de Estatística, Matemática e
Computação

Computação Gráfica

Daniel Pedronette
pedronette@gmail.com

Circunferência

- A equação de uma circunferência com centro na origem e raio R é dada por:

$$X^2 + y^2 = R^2$$

- Circunferências não centradas na origem, podem ser transladadas para o ponto $(0,0)$ e ao traçar os pixels reais, realiza-se deslocamento dado pelo centro.

Circunferência

- Outra possível abordagem seria utilizar a equação paramétrica da circunferência:

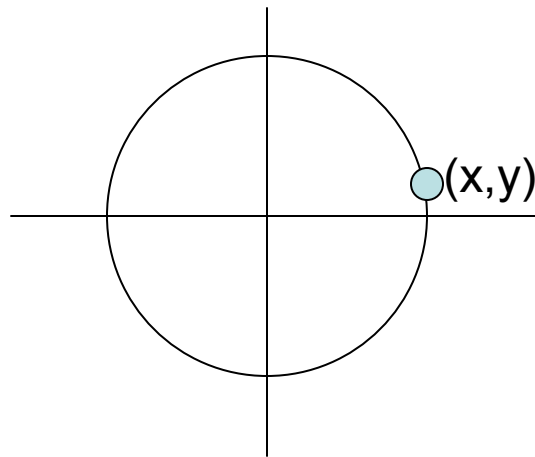
$$x = R * \cos(t)$$

$$y = R * \sin(t)$$

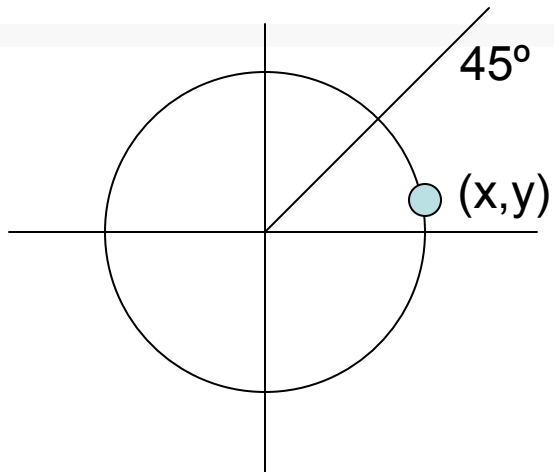
, onde $0 \leq t \leq 2$

Circunferência

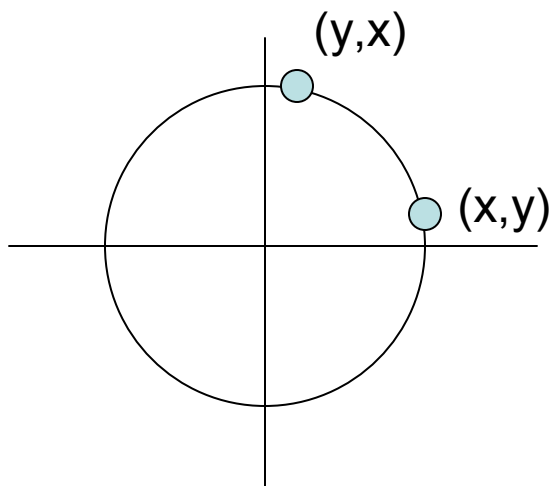
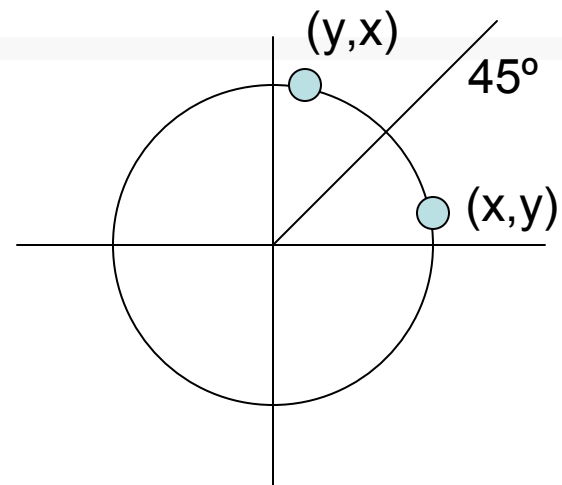
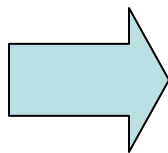
- Traçado de uma circunferência pode tirar proveito da simetria da forma.
- Vamos considerar uma circunferência centrada na origem $(0,0)$ e o ponto (x,y) pertencente a circunferência.



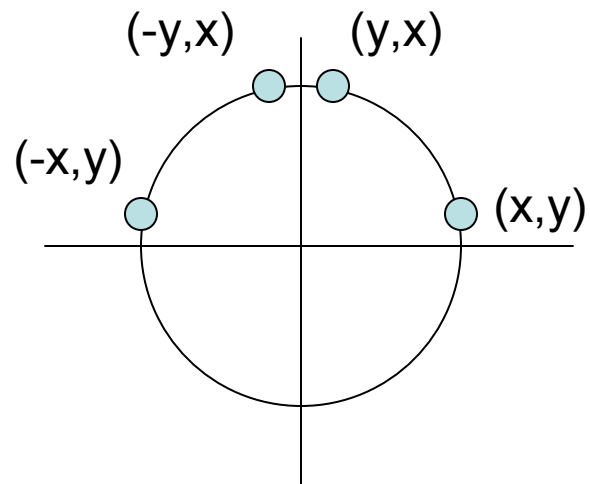
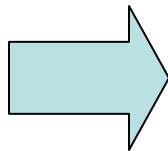
Circunferência



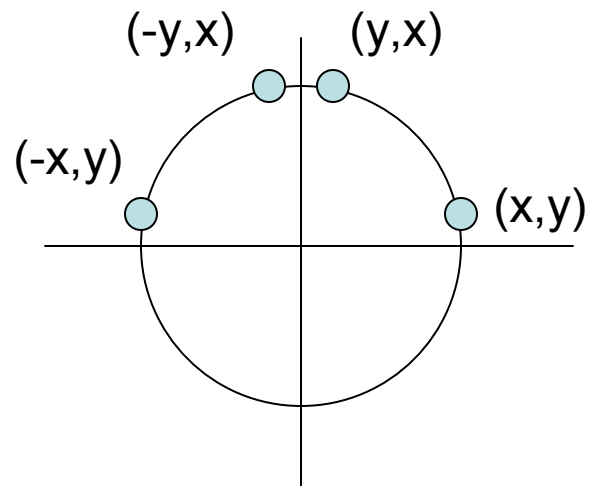
Espelho
 45°



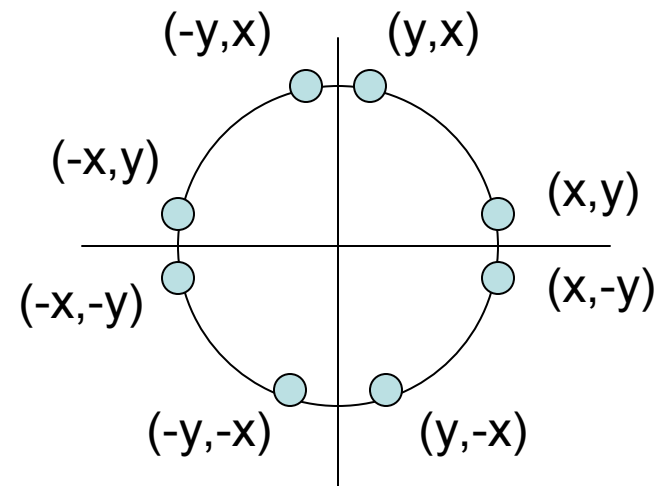
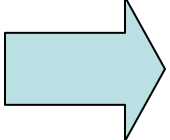
Espelho
Eixo Y



Circunferência



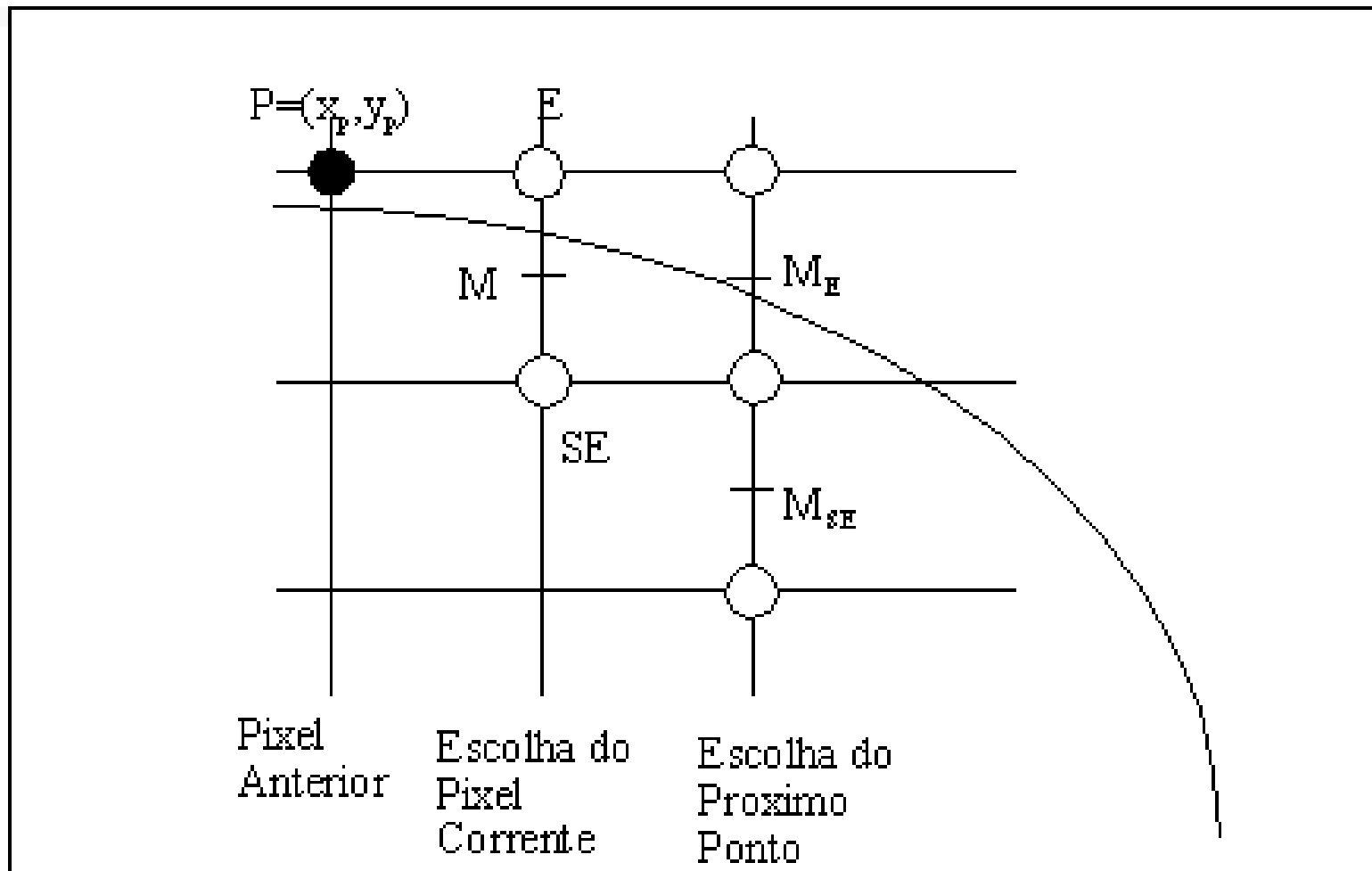
Espelho
Eixo X



Circunferência

- Consideraremos apenas um arco de 45° da circunferência:
 - de $x=0, y=R$ a $x=y=R/(2)$, e usaremos o procedimento *CirclePoints* para traçar todos os pontos da circunferência.
- Assim como o algoritmo gerador de linhas, a estratégia é selecionar entre 2 pixels na malha aquele que está mais próximo da circunferência, avaliando-se uma função no ponto intermediário entre os dois pixels.

Circunferência



Circunferência

- Seja a função $F(x,y) = x^2 + y^2 - R^2$; cujo valor é 0 sobre a circunferência, positivo fora dela e negativo dentro.
- Se o “ponto médio” entre os pixels E e SE está fora da circunferência, o pixel SE é escolhido, porque está mais próximo dela.
- Por outro lado, se o pixel intermediário está dentro da circunferência, então o pixel E é escolhido.

Circunferência

- **Resumindo**, os mesmos dois passos executados para o algoritmo do traçado de linhas são executados para o algoritmo de circunferências:
 - (1) escolher o pixel com base no sinal da variável d , calculada na iteração anterior;
 - (2) atualizar a variável d com o valor correspondente ao pixel escolhido. A diferença é que, na atualização de d , calculamos uma função linear do ponto de avaliação.

Circunferência

```
void MidpointCircle (int raio, int valor)
/* Assume que o centro da circunferencia e' a origem */
{ int x,y;
  float d;

  /* Inicialização das variaveis */
  x=0;
  y=raio;
  d=5/4 - raio;
  CirclePoints (x,y,valor)

  while (y > x) {
    if (d < 0) { /* Seleciona E */
      d=d + 2*x + 3;
      x++;
    }
    else { /* Seleciona SE */
      d=d + 2*(x-y) + 5;
      x++; y--;
    }
    CirclePoints (x,y,valor)
  } /* Fim while */
} /* Fim da rotina MidpointCircle */
```

Exercício

- **Objetivo:** sedimentar conhecimentos sobre o algoritmo de circunferência.
- **Descrição:** faça um algoritmo genérico para criação de circunferências com centro em qualquer ponto.

Circunferência

```
void circleMidPoint(int xCenter, int yCenter, int radius)
{
    int x = 0;
    int y = radius;
    int p = 1 - radius;
    void plotpoints(int, int, int, int);

    plotpoints (xCenter, yCenter, x, y);

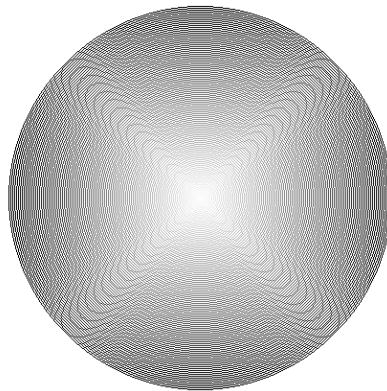
    while (x < y) {
        x++;
        if (p < 0)
            p += 2 * x + 3;
        else {
            y--;
            p += 2 * (x - y) + 5;
        }
        plotpoints (xCenter, yCenter, x, y);
    }
}
```

Circunferência

```
void plotpoints (int xCenter, int yCenter, int x, int y)
{
    myputpixel (xCenter + x, yCenter + y, cor);
    myputpixel (xCenter - x, yCenter + y, cor);
    myputpixel (xCenter + x, yCenter - y, cor);
    myputpixel (xCenter - x, yCenter - y, cor);
    myputpixel (xCenter + y, yCenter + x, cor);
    myputpixel (xCenter - y, yCenter + x, cor);
    myputpixel (xCenter + y, yCenter - x, cor);
    myputpixel (xCenter - y, yCenter - x, cor);
}
```

Exercício Final

- **Objetivo:** aplicar conhecimentos do algoritmo de circunferências.
- **Descrição:** crie uma primitiva gráfica que desenhe uma circunferência com preenchimento gradiente.



Referências

- HEARN, D. e BAKER, PAULINE - Computer Graphics with OpenGL, Prentice-Hall, 2004.
- FRANCIS S. HILL JR. - Computer Graphics, Macmillan Publishing Company, 1990.