Primitivas de Saída

Computação Gráfica

Rasterização de Circunferências

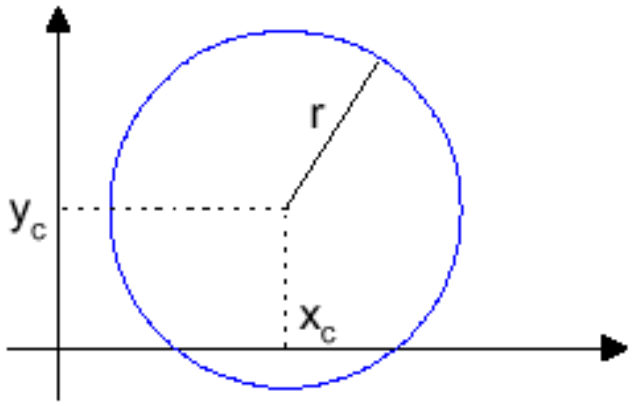
- Uma circunferência é definida como o conjunto de todos os pontos que ficam a uma distância r do centro.
- Equação da circunferência:

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

  - X varia de x_c-r até x_c+r

    $y = y_c \pm \sqrt{r_2 - (x_c - x)^2}$

  - Elevada carga computacional
  - Espaçamento não uniforme entre pixels



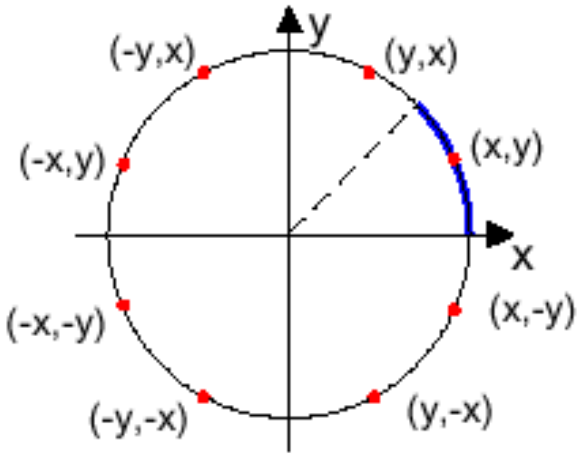
Rasterização de Circunferências

- Equações paramétricas (coordenadas polares):

  $x = x_c + r.\cos(\theta)$

  $y = y_c + r.\text{sen}(\theta)$

- Usando um passo angular constante, tem-se distâncias uniformes entre pixels.
- Aproveitar simetria da circunferência
  - Calcular apenas 1 octante



Rasterização de Circunferências

- Generalização do algoritmo para retas
- Considerando o ponto atual (P_k), e para $45° \leq \theta \leq 90°$, o ponto seguinte (P_{k+1}) só poderá ser o ponto à direita (E) e o ponto abaixo à direita (SE).
- A escolha do ponto seguinte depende de d1 e d2: $d_1 - d_2 < 0 \Rightarrow$ ponto E

  $d_1 - d_2 \geq 0 \Rightarrow$ ponto SE

Rasterização de Circunferências

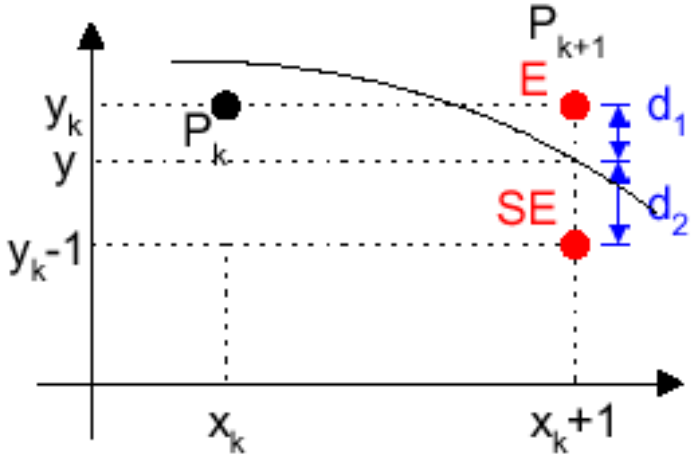
- Considerando apenas o caso $45° \leq \theta \leq 90°$, temos

  $y^2 = r^2 - (x_k+1)^2$

- Calcula-se duas medidas relacionadas com a distância:

  $d_1 = y_k^2 - y^2 = y_k^2 - r^2 + (x_k+1)^2$

  $d_2 = y^2 - (y_k-1)^2 = r^2 - (x_k+1)^2 - (y_k-1)^2$



Rasterização de Circunferências

- Determina-se a diferença entre ambas:

  $p_k = d_1 - d_2 = y_k^2 - 2r^2 + 2(x_k+1)^2 + (y_k-1)^2$

- Pode-se agora calcular p_k de forma incremental:

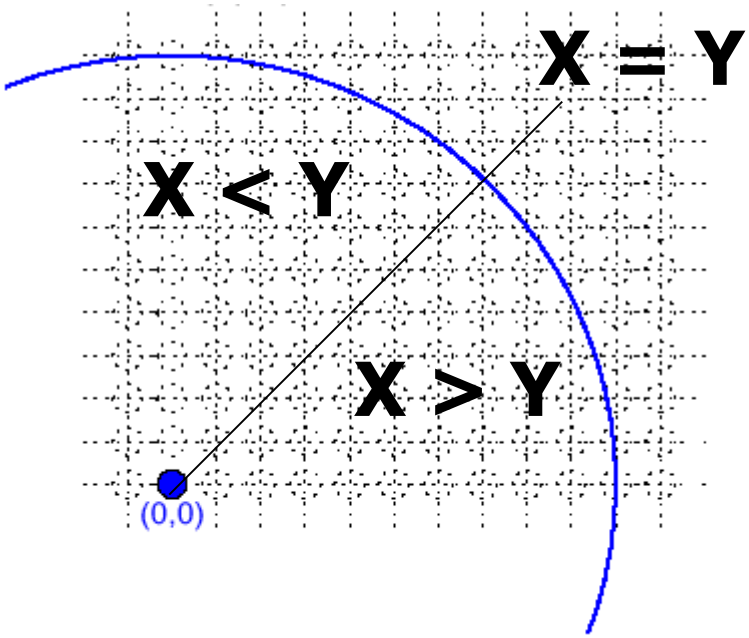
  $p_{k+1} = y_{k+1}^2 - 2r^2 + 2((x_k+1)+1)^2 + (y_{k+1}-1)^2$

  $p_{k+1} - p_k = 4x_k + 6 + 2(y_{k+1}^2 - y_k^2) - 2(y_{k+1}-y_k)$

Rasterização de Circunferências

- Onde:

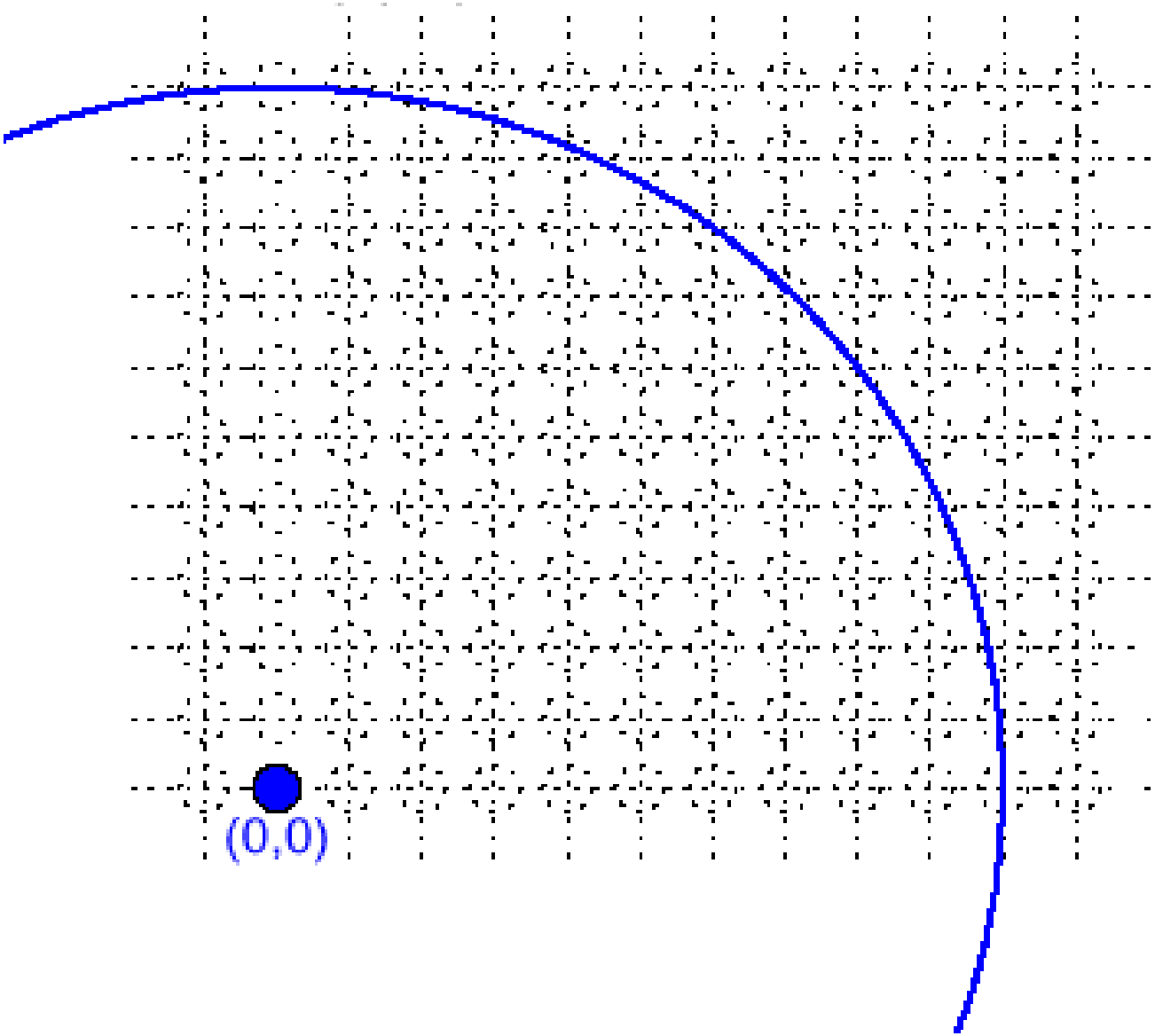
- E também



Algoritmo (considerar 45º<=θ<=90º)

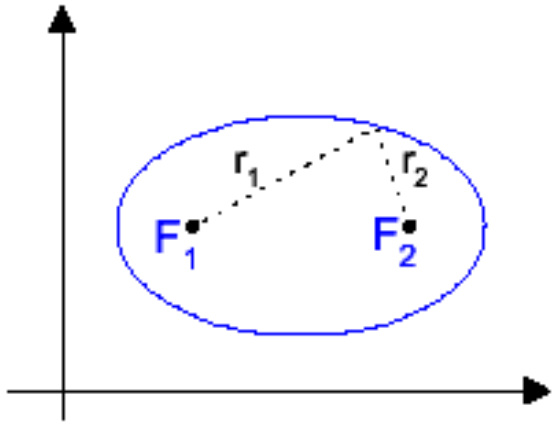
```
procedimento circBresenhams (xc,yc,r:inteiro);
  var
    x, y, p : inteiro;
  procedimento plot_circle_points;
    início
      set_pixel(xc+x,yc+y);
          ...
      set_pixel(xc-y,yc-x);
    fim{procedimento plot_circle_points}
início
  x = 0;  y= r; p = 3 - 2*r
  plot_circle_points;
  enquanto x < y faça
    início
      se p < 0 então
        início
          p = p + 4*x + 6;
        senão
          p = p + 4*(x - y) + 10;
          y = y - 1;
        fim {se}
      x = x + 1;
      plot_circle_points;
    fim (enquanto)
fim {procedimento circBresenhams}
```

```
set_pixel(xc+x,yc+y);
set_pixel(xc-x,yc+y);
set_pixel(xc+x,yc-y);
set_pixel(xc-x,yc-y);
set_pixel(xc+y,yc+x);
set_pixel(xc-y,yc+x);
set_pixel(xc+y,yc-x);
set_pixel(xc-y,yc-x);
```


(0,0)

Rasterização de Primitivas

- Uma elipse é definida como o conjunto de todos os pontos, cuja soma das distâncias a pontos denominados focos é constante.
- Extensão do algoritmo de rasterização de circunferências.
- Equação da elipse:



Rasterização de Primitivas

- Caracteres:
  - O estilo do desenho de um conjunto (tipo) de caracteres denomina-se font ou typeface.
  - Duas formas de representação:
    - Bitmapped font: armazenado em uma malha retangular.
      - Ocupa bastante memória, pois **todo** tipo de letra deve ser armazenado.

Rasterização de Primitivas

- Caracteres
  - Outline font: definidas a partir de um conjunto de primitivas geométricas.
    - Ocupa menos memória, pois diferentes tipos de letras podem ser obtidos por manipulação da sua definição geométrica.
    - Maior carga computacional.