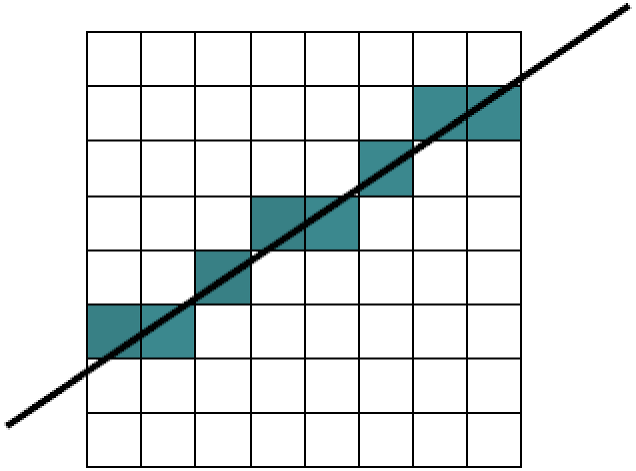
Universidade Federal de Ouro Preto - JM
Departamento de Computação e Sistemas – DECSI
ICEA - Instituto de Ciências Exatas e Aplicadas

Rasterização de Círculos e Polígonos

Gilda Aparecida de Assis

Rasterização

- Rasterização é o processo de conversão entre representações vetorial e matricial

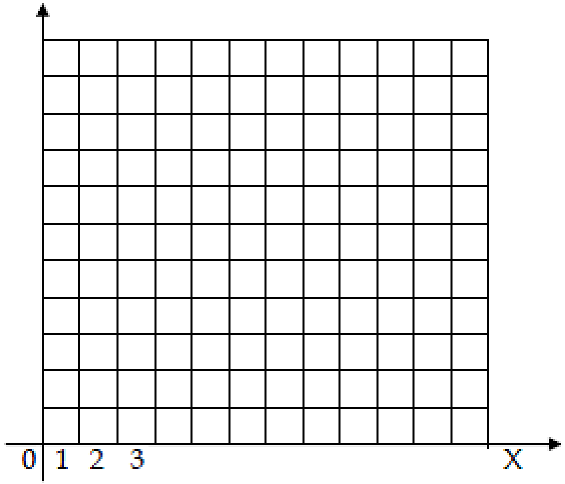


O problema da Rasterização

- Entrada:
  - Primitivas 2D no espaço
- Saída
  - Coleção de fragmentos de pixels a serem pintados junto com os atributos (cor, profundidade, etc.) interpolados para cada pixel

Considerações Gerais

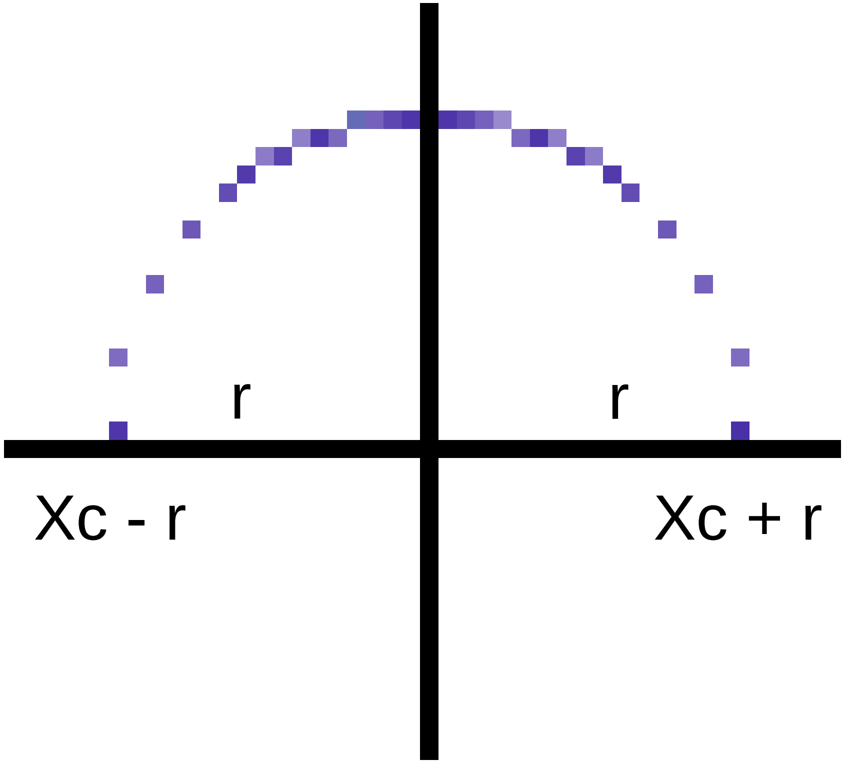
- Vamos supor que o espaço de exibição (frame buffer) é uma matriz referenciada pelo seguinte sistema de coordenadas:



Rasterização de Círculos

- Um circulo pode ser definido como um conjunto de pontos que estão uma dada distância r de um ponto centra (xc,yc).
- Um ponto de um circulo deve obedecer a expressão
  - $(x - xc)^2 + (y - yc)^2 = r^2$
- Se usarmos essa equação para calcular os pontos do circulo com x variando de xc -r até xc + r.

Rasterização de Círculos



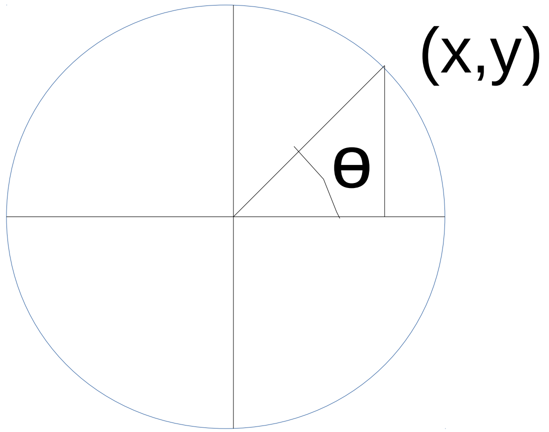
Rasterização de Círculos

- Outra possibilidade é usar as equações

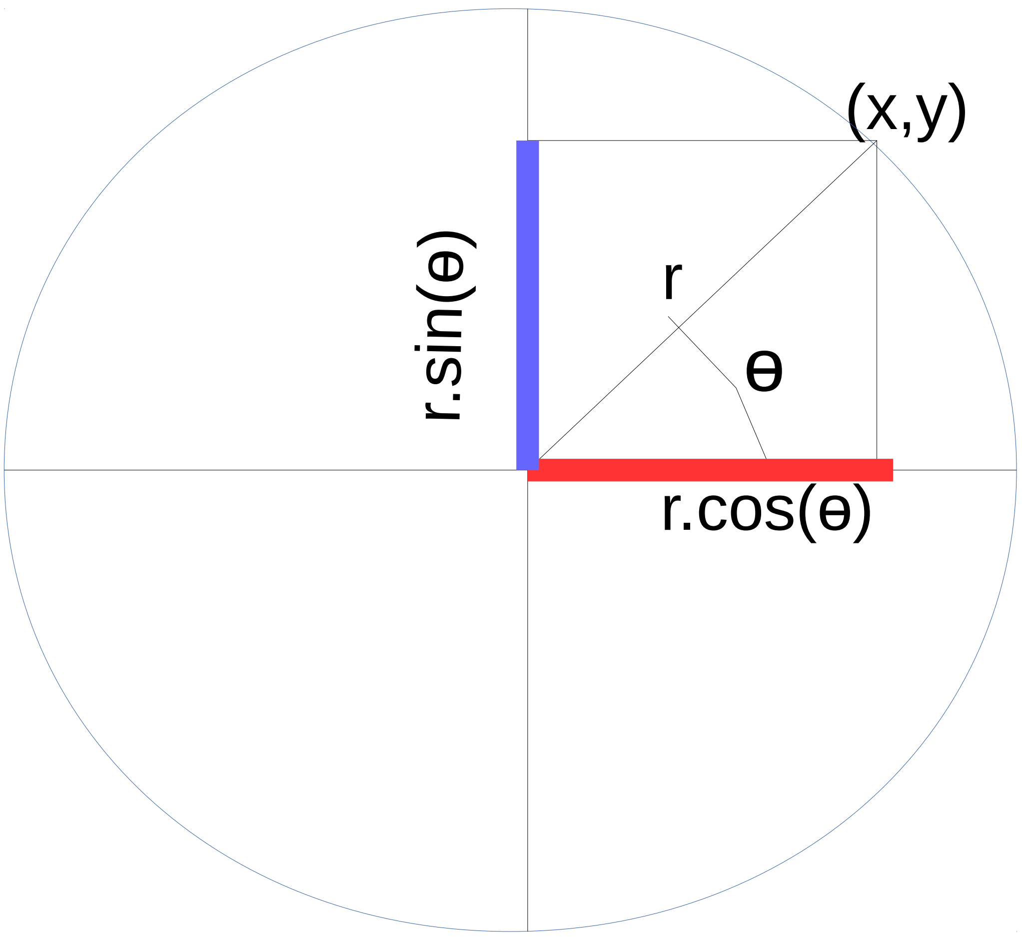
$$x = xc + r\cos\theta$$

$$y = yc + r\text{sen}\theta$$

- Construir o circulo com incrementos de θ.



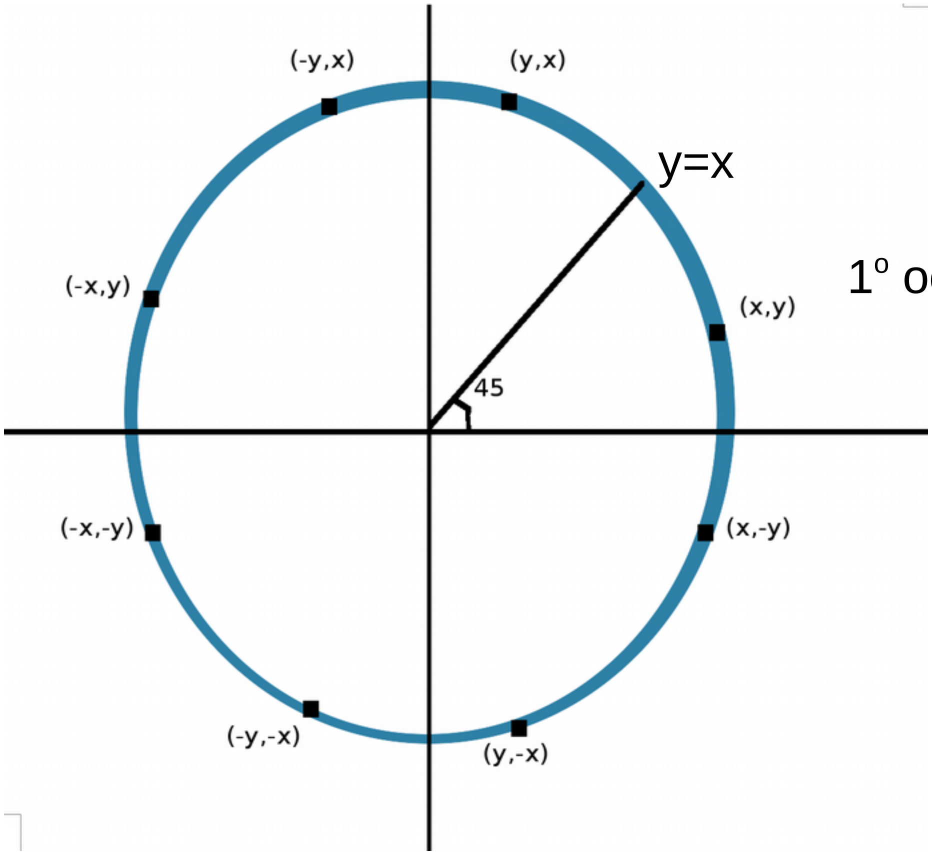
Rasterização de Círculos



Rasterização de Círculos

- Para reduzir os cálculos usamos a simetria do circulo.

Oct	Xn	Yn
1	x	y
2	y	x
3	y	-x
4	-x	y
5	-x	-y
6	-y	-x
7	-y	x
8	x	-y



1º octante

Algoritmo trivial

- Um algoritmo trivial de traçado de círculos consiste na escolha de um passo para θ (Δθ) e um loop que calcula os valores de x e y para cada valor de θ.
  - Cada ponto calculado é inserido numa tabela que mais tarde é fornecida como parâmetro para traçado de uma polilinha fechada.
  - O cálculo de um octante (1/8 do círculo, com θ de 0 a π/4) pode ser copiado para os outros octantes por simetria

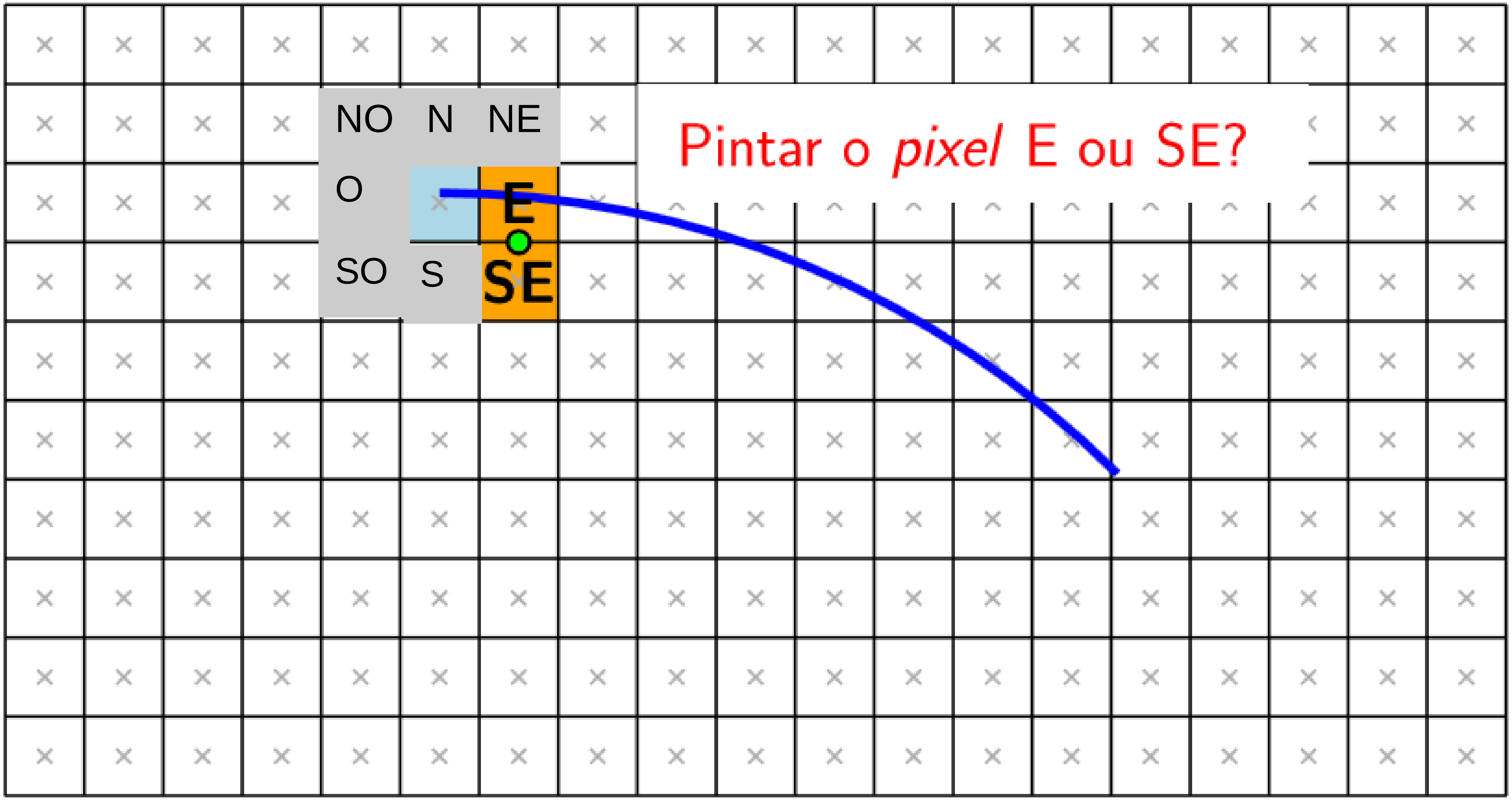
Algoritmo trivial

- Apesar de simples, esse algoritmo apresenta dois problemas:
  - A precisão é dependente do raio do círculo
  - O uso de cosseno e seno, mesmo para poucos pontos, utiliza aritmética de ponto flutuante e portanto precisa de arredondamento.

Algoritmo do Ponto Médio

- Avaliar incrementalmente uma função que classifica o ponto médio entre um pixel e outro com relação a uma função implícita
- Apenas um octante é avaliado, os demais são simétricos
  - Para cada pixel computado, oito são pintados
  - Derivação um pouco mais difícil que a da reta

Algoritmo do Ponto Médio



Algoritmo do Ponto Médio

- Podemos usar a função
  - $f(x,y) = x^2 + y^2 - r^2$, sendo $xc = 0$, $yc = 0$

- Se $f(x,y) < 0$ então Ponto dentro do circulo
- Se $f(x,y) = 0$ então Ponto sobre o circulo
- Se $f(x,y) > 0$ então Ponto fora do circulo

Algoritmo do Ponto Médio

- $P_k = f(x_k + 1, y_k - ½)$

  $= (x_k + 1)^2 + (y_k - ½)^2 - r^2$

- Se $P_k < 0$, o ponto médio está dentro do circulo e devemos escolher E = $y_{k+1} = y_k$

- Se $P_k > 0$, o ponto médio está fora do círculo e usamos SE = $y_{k+1} = y_k - 1$

Algoritmo do Ponto Médio

- Depois de calcular P_k e decidir qual será o y_{k+1} (y_k ou y_k-1) para x_k+1 o próximo passo, é incrementar x:
- $P_{k+1} = f(x_k+1 +1 , y_{k+1} – ½)$

  $= ((x_k +1)+ 1)^2 + (y_{k+1} – ½)^2 - r^2$

  $= [(x_k +1)^2 + 2(x_k +1) + 1] + [y^2_{k+1} - y_{k+1} + (½)^2] - r^2$
- $P_k = (x_k +1)^2 + (y_k – ½)^2 - r^2 = (x_k +1)^2 +(y^2_k - y_k + (½)^2) - r^2$
- $P_k - y^2_k + y_k = (x_k +1)^2 + (½)^2 – r^2$, substituindo e agrupando em P_{k+1}
- $P_{k+1} = P_k + 2(x_k +1) + (y^2_{k+1} – y^2_k) - (y_{k+1} – y_k) + 1$
- Onde y_{k+1} é y_k ou y_k – 1 dependendo do sinal de P_k

Algoritmo do Ponto Médio

- $P_{k+1} = P_k + 2(x_k + 1) + (y^2_{k+1} – y^2_k) - (y_{k+1} – y_k) + 1$
- Onde y_{k+1} é y_k ou $y_k – 1$ dependendo do sinal de P_k
- Se $y_{k+1} = y_k$ então
  - $P_{k+1} = P_k + 2(x_k + 1) + (y^2_k – y^2_k) - (y_k – y_k) + 1$
  - $P_{k+1} = P_k + 2(x_k+1) + 1$

Algoritmo do Ponto Médio

- Se $y_{k+1} = y_k - 1$ então
  - $P_{k+1} = P_k + 2(x_k +1) + ((y_k-1)^2 – y^2_k) - (y_k - 1 - y_k) + 1$
  - $P_{k+1} = P_k + 2(x_k+1) + 1 - 2y_k+ 2$

  Como $y_{k+1} -1 = y_k$ então
  - $P_{k+1} = P_k + 2(x_k+1) + 1 - 2y_{k+1} - 2 + 2$
  - $P_{k+1} = P_k + 2(x_k+1) + 1 - 2y_{k+1}$

Caso base

- Fazendo a posição inicial $(x_0, y_0) = (0, r)$:
- $P0 = f(1, r - \frac{1}{2})$

  $= 1^2 + (r - \frac{1}{2})^2 - r^2$

  $= 1 + r^2 - r + (\frac{1}{2})^2 - r^2 =$

  $= 5/4 - r$
- Se r é inteiro, podemos arredondar para 1 - r

Algoritmo

- $X \leftarrow 0$; $y \leftarrow r$; $P \leftarrow 5/4 - r$
- Enquanto $x \leq y$ *faça:*
  - Pintar pixel (x, y)
  - $x \leftarrow x + 1$
  - Se P < *0*
    - Então P $\leftarrow P + 2 * x + 1$
  - Senão
    - $y \leftarrow y - 1$
    - $P \leftarrow P + 2 * x + 1 - 2 * y$

Exercício

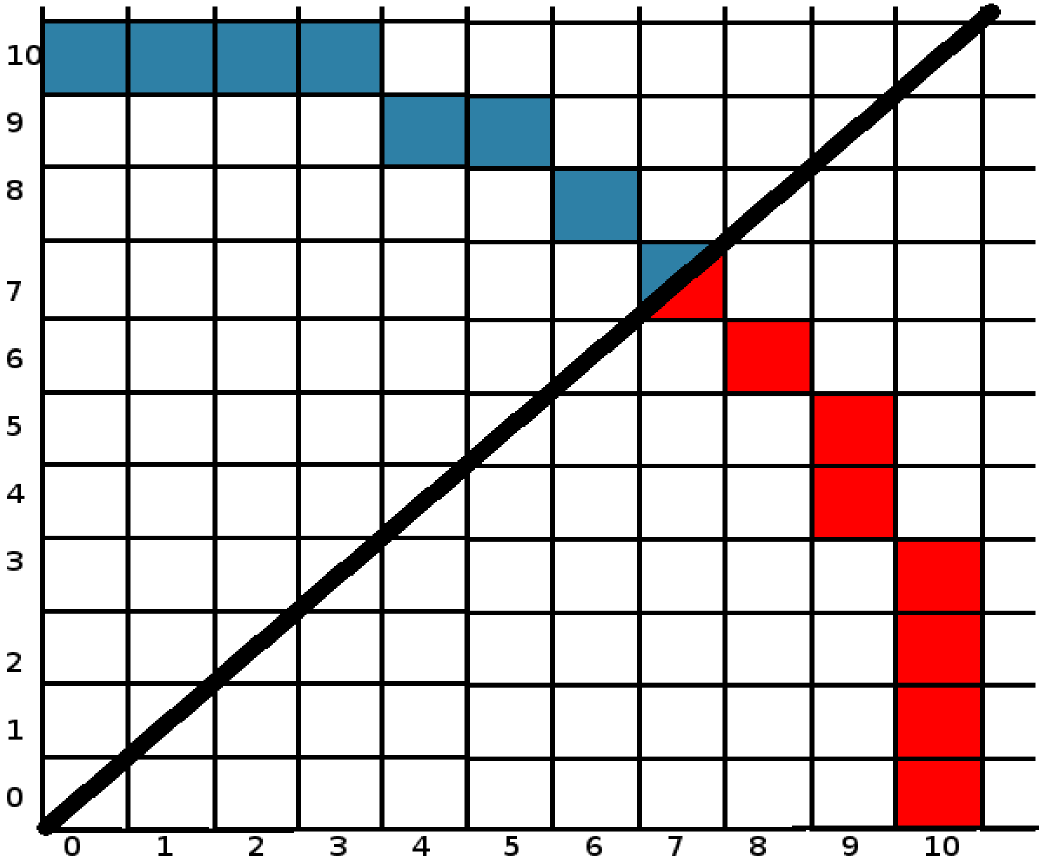
- Dado um circulo de raio = 10 centrado no (0,0), calcule os pontos utilizando o algoritmo do ponto médio.

Exercício

- Dado um circulo de raio 10 centrado no (0,0).

k	P	(x,y)	$2(x_{k+1})+1$	$-2y_{k+1}$
0	-9	(1,10)	3	
1	-6	(2,10)	5	
2	-1	(3,10)	7	
3	6	(4,9)	9	-18
4	-3	(5,9)	11	
5	8	(6,8)	13	-16
6	5	(7,7)	15	-14

Exemplo



Outras cônicas

- Outras cônicas podem também ser rasterizadas de forma semelhante

Preenchimento de Regiões

- Da mesma forma que no traçado de linhas e curvas, precisamos de um mecanismo eficiente para determinar quais pixels estão no interior de um região e devem ser “pintados”

Preenchimento de regiões

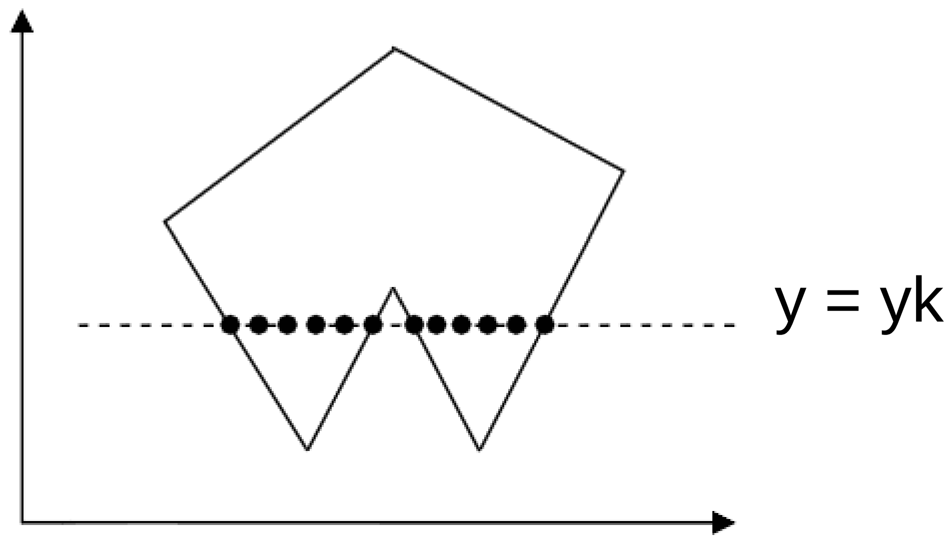
- Em geral, os métodos de especificação da região podem ser classificados em duas categorias:
  - Independem do estado atual da imagem
  - Aqueles que envolvem a avaliação de atributos da imagem

Algoritmo Scan-Line

- Vamos supor regiões poligonais descritas através dos seus vértices
- Algoritmo 1
  - para cada linha de varredura, determine os pontos de interseção entre essa linha e a região poligonal
  - ordene em ordem crescente de *x a lista de pontos obtida no passo anterior*
  - para cada par de pontos (pixels) da lista obtida acima pinte os pixels entre estes dois pontos

Scan-Line

Exemplo:

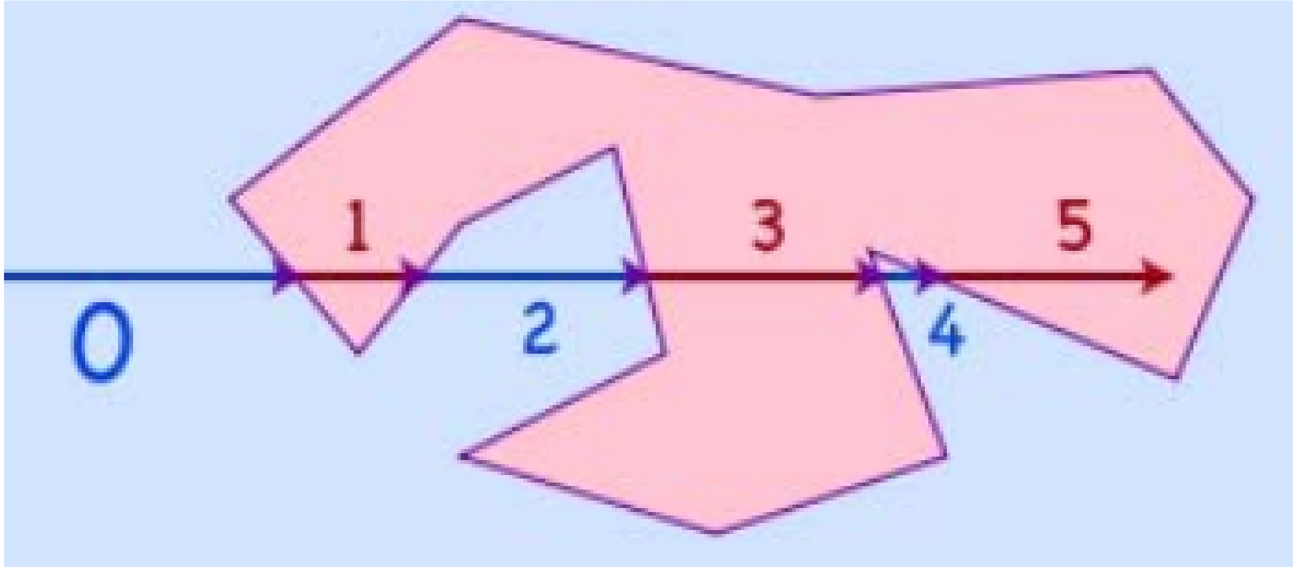


Regra da paridade

Como determinar se um ponto está dentro ou fora do polígono?
-Contar o número de arestas do polígono interceptadas pela scan line até o momento.

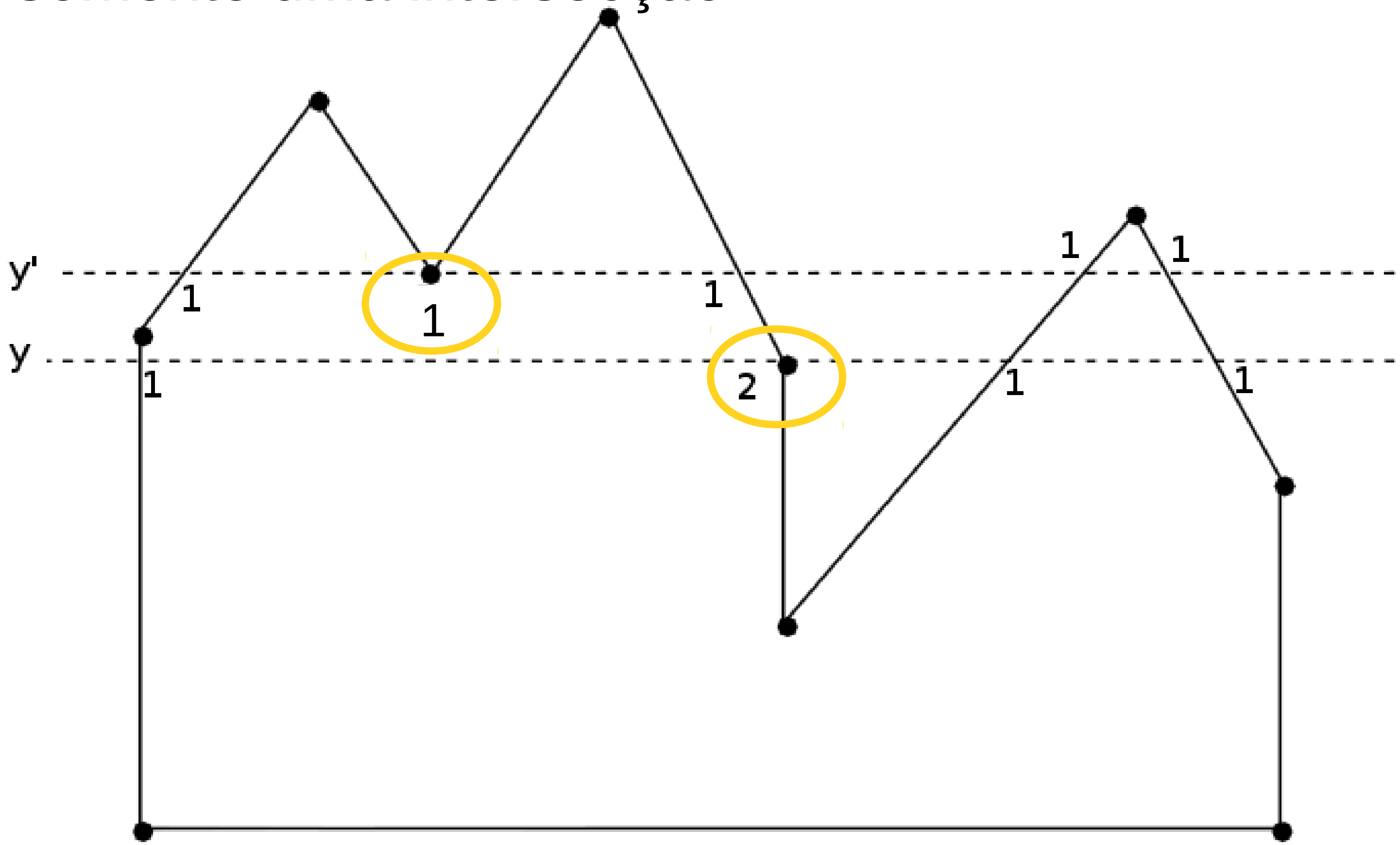
Se ímpar está dentro

Se par está fora



Dificuldades

E quando a scan line intercepta um vértice compartilhado por duas arestas?
Y’ : contar as 2 intersecções
Y: contar somente uma intersecção



DECSI

Solução

- Sempre que uma Scan-Line passar por um vértice podemos fazer o teste
  - Se ambos os vértices das arestas interceptadas (A1, A2) estão do mesmo lado da Scan-Line ele contará como 2 vezes na paridade pois vértice é máximo(A1, A2) ou mínimo(A1, A2).
  - Caso contrario contará como 1 pois vértice é máximo(A1) e mínimo (A2) ou vice-versa.

Scan line

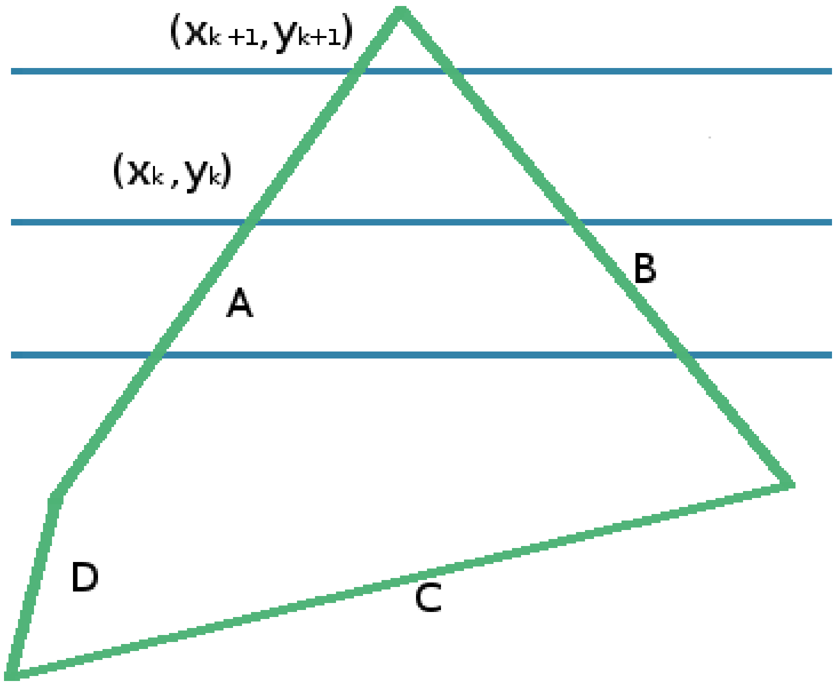
- Como calcular a intersecção de uma scan line $y = y_k$ com uma aresta do polígono?
  - Aresta $A_i \rightarrow y = mx+b$,
  - Onde $m = (y_1-y_0)/(x_1-x_0)$,
  - $b = y_0 – mx_0$
  - Para x variando de x_0 a x_1
  - Substituindo $y = y_k$, tem-se $x = (y_k-b)/m$
  - Se o valor calculado de xϵ [x0,x1] então intercepta
  - Caso contrário, não intercepta

Scan line

- Manter uma lista de arestas ativas para cada scan line y = yk.
  - Arestas ativas são aquelas em que yk está entre os valores mínimo e máximo de y da aresta e x atual da scan line está entre x mínimo e máximo da aresta

Propriedade de coerência

- Podemos usar a coerência entre duas Scan-Lines para reduzir o processamento.



Teste de interior/exterior (teste de paridade)

- A posição de um ponto em relação a um polígono, isto é, se o ponto está no interior ou no exterior do polígono, pode ser determinada testando a paridade do número de interseções entre a semi-reta com origem no ponto e as arestas do polígono.