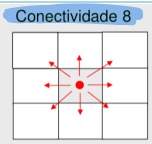
Preenchimento de Regiões

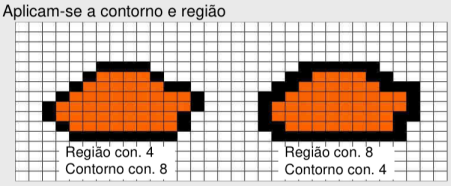
- Classificação dos algoritmos:
  - Preenchimento segundo contorno existente
    - ➢ Por difusão [flood-fill]:
      a. Limitado por contorno
      b. Limitado ao interior de região
    - ➢ Por análise do contorno [boundary algorithm]
  - Preenchimento por varrimento segundo descrição de contorno [scan conversion]
    - ➢ Algoritmo da lista de pontos de fronteira ordenados
    - ➢ Algoritmo da lista de arestas ativas

Conectividade 4

Conectividade 8



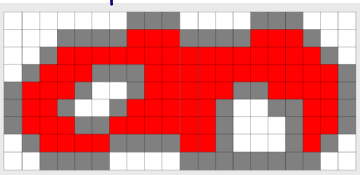
Aplicam-se a contorno e região



Preenchimento segundo Contorno Existente

Por difusão (flood-fill)

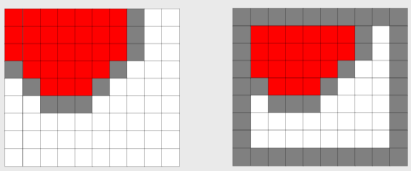
- Limitado pelo Contorno



Princípio: Começa num ponto interior e "espalha-se" como se fosse líquido.

Funciona em regiões com buracos.

Fronteira não completamente fechada → pode originar erro durante a execução



Evitam-se os erros se a leitura pointColor(x,y) fornecer o valor correspondente a ContourColor no caso do ponto se encontrar fora do ecrã.

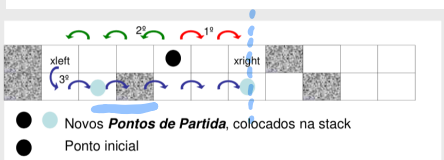
- Limitado ao interior da região

```
void floodfill(int x, int y)
{  if (pointColor(x,y) == RegionColor)
   {    ChangeColor(x,y, FillColor);
        // apelo recursivo aos 4 vizinhos
        floodfill(x+1,y);
        floodfill(x-1,y);
        floodfill(x,y+1);
        floodfill(x,y-1);
   }
}
```

Aplicação: para substituir uma cor por outra
Problemas: consumo de *stack* (pilha)
Solução:
- Evitar declarar variáveis locais
- Não passar a cor de preenchimento como parâmetro

Notar que agora não existe o problema da fronteira incompleta.

Por Análise do Contorno (boundary algorithm)



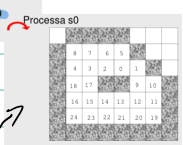
Novos ***Pontos de Partida***, colocados na stack

Ponto inicial

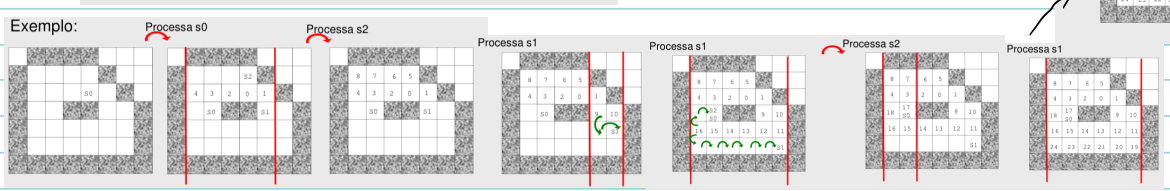
Princípio: trabalha linha a linha e apenas coloca na pilha alguns "pontos de partida".

Algoritmo:
1. Parte de um ponto inicial, situado no interior, que começa por ser colocado na pilha.
2. Se pilha vazia, então termina,
   senão retira um ponto da pilha.
3. A partir desse ponto preenche na horizontal, para a direita e, em seguida, para a esquerda até encontrar o contorno. Toma nota das extremidades Xleft e Xright.
4. Na linha imediatamente abaixo procura, entre Xleft e Xright, os novos pontos de partida. Estes pontos são colocados na pilha.
5. Idem 4, para a linha imediatamente acima.
6. Salta para 2.

Ponto de partida:
1- Pixel de região que possui à sua direita um pixel de contorno
2- Pixel de região na coordenada XRight

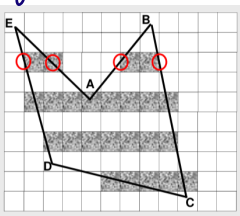


Exemplo:



Preenchimento por Varrimento segundo Descrição de Contorno (Scan Conversion)

Algoritmo da Lista de Pontos de Fronteira Ordenados



Algoritmo:

1. Determinação dos Pontos de Fronteira
   Intersecções das arestas com as linhas de varrimento do ecrã (utilizando, por exemplo, o algoritmo MidPoint modificado, de tal forma que produza um só ponto por horizontal)
2. Ordenação dos Pontos de Fronteira. Primeiro segundo Y e, em seguida, para o mesmo Y, segundo X.
   X1,Y1 precede X2,Y2 se (Y1 < Y2) OR
   se (Y1 = Y2) AND (X1 <= X2)
3. Os segmentos horizontais de preenchimento são agora especificados considerando pares de pontos consecutivos.

O algoritmo determina os pontos de interseção das arestas com as linhas de varrimento do ecrã e ordena-os. Os pontos a preencher estão entre pares de pontos.

Pontos de Fronteira obtidos

(0,9) (0,9)	(7,9) (7,9)
(0,8) (1,8)	(6,8) (7,8)
(1,7) (2,7)	(5,7) (7,7)
(1,6) (3,6)	(5,6) (8,6)
(1,5) (4,5)	(4,5) (8,5)
(1,4) (8,4)	
(2,3) (8,3)	
(2,2) (9,2)	
(4,1) (9,1)	
(9,0) (9,0)	

Cuidado com os vértices duplos

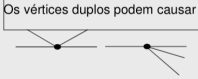
visto em baixo

Esta aresta só gera um ponto de intersecção

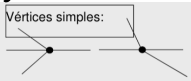
Simplificação da estrutura de dados: guardar por segmentos (x1, x2, y).
Ex: (0,0,9) (7,7,9)
(0,1,8) (6,7,8) ....

Problema dos Vértices Duplos

Os vértices duplos podem causar problemas:



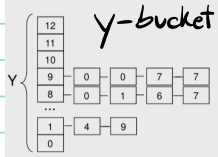
Vértices simples:



Desvantagem do algoritmo: a ordenação pode ser um processo lento por envolver um elevado número de pontos.

Melhoramento: **Algoritmo da tabela de listas de pontos ordenados**

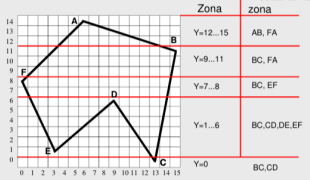
Consiste em construir uma lista ordenada de pontos para cada valor de Y.



Algoritmo:

1. Determinar as intersecções (x_i, y_i) para cada aresta. Para cada intersecção colocar x_i na lista y_i.
2. Em cada lista y_i, ordenar os valores X por ordem crescente.
3. Em cada lista y_i, considerar os pares de valores X consecutivos, que definem os segmentos horizontais a visualizar.

Algoritmo da Lista de Arestas Ativas



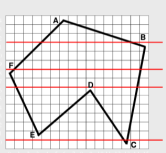
- O preenchimento realiza-se por linha de varrimento do ecrã, pelo que será viável tratar e memorizar apenas os pontos relativos a essa linha.
- Só as arestas ativas são usadas no preenchimento de uma linha de varrimento.
- → Interessa manter a **Lista das Arestas Ativas**.
- Esta lista é atualizada sempre que se entra numa nova zona vertical.

Algoritmo da Lista das Arestas Ativas (AEL):

1. Constituição da **Tabela das Arestas**
   - Para cada aresta é calculado e memorizado:
     - A coordenada X (valor inicial a partir do primeiro vértice).
     - DX, valor a adicionar a X, para encontrar o ponto seguinte da aresta quando se incrementa Y de 1.
     - LongY, altura da aresta segundo o eixo Y.
2. Para cada linha de varrimento:
   - Verificar na tabela de arestas se existem novas arestas nesta linha. Em caso afirmativo, juntam-se à AEL.
   - Ordenar AEL pelos valores de X.
3. Agrupa AEL aos pares de arestas, segundo os valores de X.
4. No final da linha preparar a informação para a linha seguinte:
   Para cada Aresta Ativa:
   Decrementar o valor LongY. Se LongY=0, então a aresta respetiva sai da lista das arestas ativas, senão é calculado o novo X, adicionando DX ao valor atual.
5. Voltar a 2.

Análise de cima para baixo

O primeiro passo do algoritmo será a classificação dos **vértices** em: **simples** ou **duplos**. A seguir constrói-se a tabela das arestas.



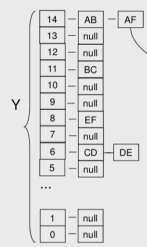
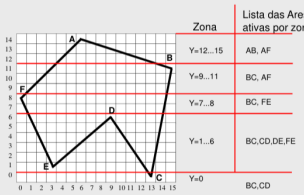


Tabela das arestas: regista as arestas que entram na AEL em cada linha.

{6, -1, 6}

Se vértice simples: longY = y2 – y1

Se vértice duplo: longY = y2 – y1 + 1



{X, DX, LongY}

AB → {6, 3, 3}
BC → {15, -0.18, 12}
CD → {9, 0.66, 7}
DE → {9, -1.2, 6}
FE → {0, 0.43, 8}
AF → {6, -1, 6}

1º Passo Y=14
Lista = {AB,AF}
Pares de valores X: (6,6)
AB→(9,3,2) AF→(5,-1,5)

2º Passo Y=13
Lista = {AB,AF}
Pares de valores X: (5,9)
AB→(12,3,1) AF→(4,-1,4)

3º Passo Y=12
Lista = {AB,AF}
Pares de valores X: (4,12)
~~AB→(15,3,0)~~ AF→(3,-1, 3)

4º Passo Y=11
Lista = {BC,AF}
Pares de valores X: (4,15)
BC→(14.82,-0.18,10) AF→(2,-1, 2)

...

Exercício

5. Seja um polígono definido pela sucessão de vértices {(1,6), (6,2), (6,6)} a ser preenchido pelo algoritmo da lista de pontos de fronteira ordenados.

a) Apresente o resultado dos dois passos iniciais do algoritmo, quando aplicado ao polígono em questão.

b) Explique como se efetua o preenchimento do polígono, com base nos resultados da alínea anterior.

(Exame de 20 de Junho de 2002)

6. Seja um polígono fechado, definido pela sucessão de vértices seguinte, a ser preenchido pelo algoritmo da Lista de Arestas Activas.

{(5, 1), (2, 4), (4, 6), (9, 6), (11, 4), (8, 1), (8, 4), (6, 2), (5, 3)}

a)- Mostre qual é o conteúdo da tabela de arestas inicial.

b)- Mostre qual é o estado da lista de arestas activas AEL nas linhas de varrimento 2, 3 e 4, logo após a inserção das novas arestas respectivas.

(Exame de 13 de Julho de 2002)