Preenchimento de Áreas

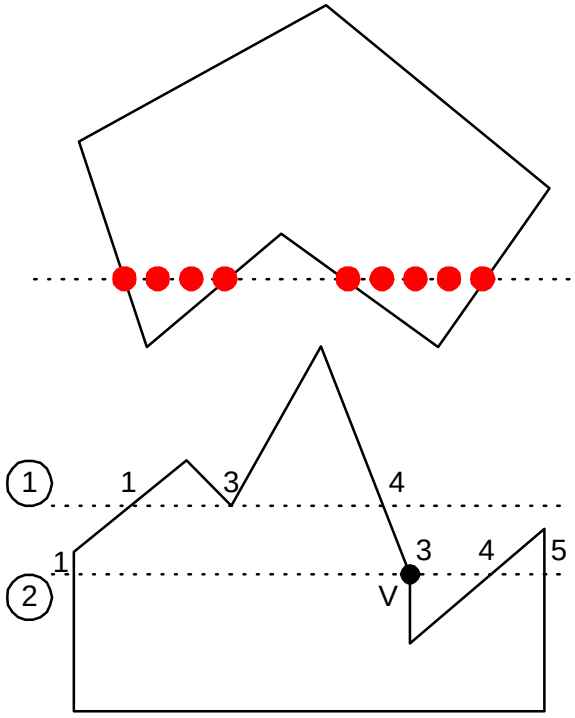
- Algoritmo *Scan-line* (varrimento de linha)

O algoritmo identifica os *pixels* que estão dentro da área do polígono através da intersecção da linha de varrimento horizontal com os limites do polígono.

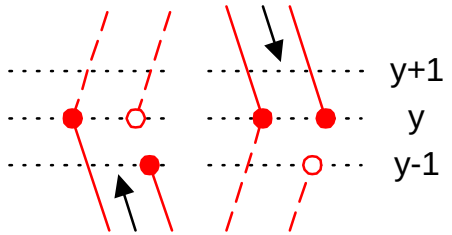
- Para cada linha de varrimento:

  Os pontos situados à direita de um número ímpar de intersecções são preenchidos.

- Podem surgir problemas quando a linha de varrimento intersecta vértices do polígono (um vértice corresponde a duas arestas):
  - ① Situação correcta
  - ② Situação incorrecta. A zona 3-4 é preenchida incorrectamente. O vértice V só deveria contar uma vez.



- Torna-se necessário considerar a topologia do polígono:
  - Ordenar a lista de arestas (p.ex.: no sentido dos ponteiros do relógio)
  - Contar 1 vértice se a variável y varia de forma monotónica.
    - p.ex.: decrementar o valor da ordenada do vértice de uma das arestas.
  - Contar 2 vértices se a variável y não variar de forma monotónica.



- Utilização das propriedades de coerência
  - O algoritmo pesquisa as regiões de cima para baixo e da esquerda para a direita. A partir de uma determinada linha de varrimento, podem ser gerados os pontos de intersecção com as arestas da linha seguinte usando um algoritmo incremental:

$$y_{k+1} = y_k - 1$$

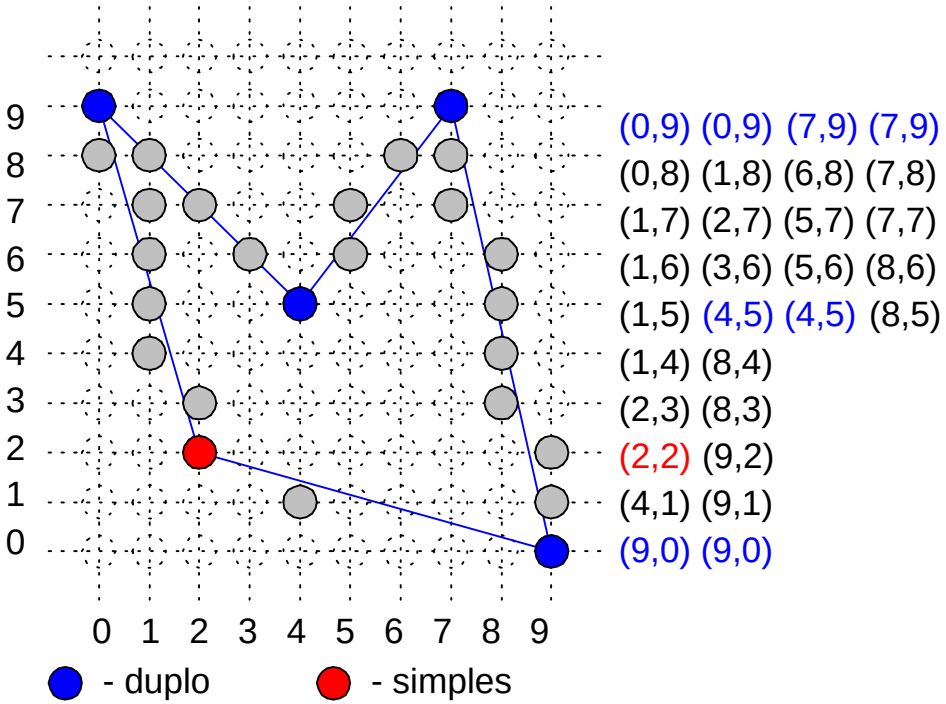
Preenchimento de Áreas *(cont.)*

- O declive da aresta pode ser determinado por:

$$m = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}$$

- e a intersecção da linha de varrimento seguinte com a aresta do polígono será:

$$x_{k+1} = x_k - \frac{1}{m}$$



Algoritmo

```
procedimento scanline;
início
  para linha_scan = ymax até ymin faça
  início
     calcular intersecções da linha de scan com as arestas;
     ordenar intersecções por valor crescente de x;
     xmin = getx( intersecção[1] );
     xmax = getx( intersecção[length(intersecção)] );
     n_intersecções  = 0;
     ind_int = 1;
     para x = xmin até xmax faça
     início
        fronteira = Falso;
        enquanto getx(intersecção[ind_int]) = x então faça
        início
            n_intersecções  = n_intersecções  + 1;
            ind_int = ind_int + 1;
           fronteira = Verdade;
        fim {enquato}
        se ímpar(n_intersecções)  ou fronteira então
           set_pixel(x, linha_scan);
        fim {se}
     fim (para)
fim {procedimento scanline}
```

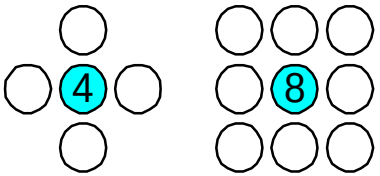
Preenchimento de Áreas *(cont.)*

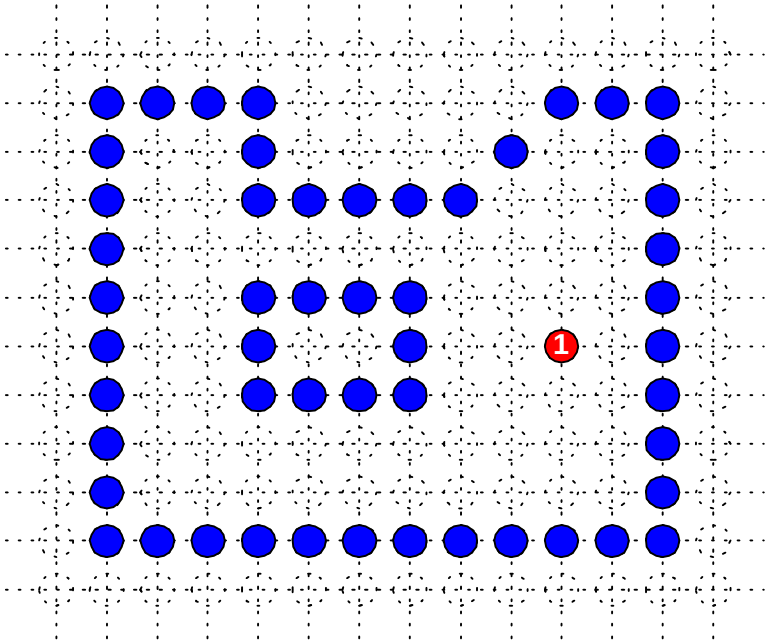
- Algoritmo Boundary-fill (preenchimento do contorno)

O algoritmo preenche a região a partir de um ponto interior, difundindo-se ("espalhando-se") até ao contorno desta.

- Muito utilizado no desenho interactivo.
- Necessita como parâmetros de entrada:
  - Ponto inicial
  - Cor para preenchimento da figura
  - Cor do contorno da região
- Conectividade
  - nº de *pixels* vizinhos testados.

Conectividade





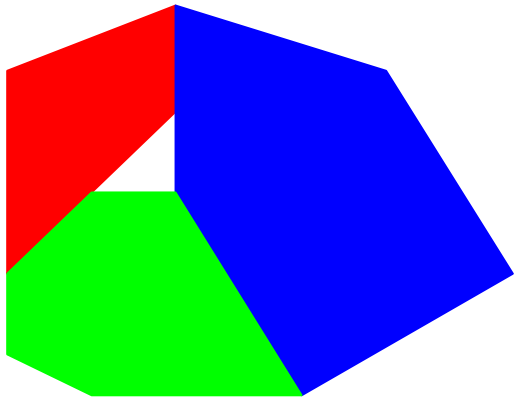
Algoritmo

```
procedimento boundary4(x, y, cor_preenche, cor_contorno : inteiro);
início
  cor_actual = inqurir_cor(x,y);
  se (cor_actual <> cor_contorno) e (cor_actual <> cor_ preenche) então
  início
     set_pixel(x,y,cor_preenche);
     {conectividade 4}
     bondary4(x+1,y, cor_preenche, cor_contorno );
     bondary4(x-1,y, cor_preenche, cor_contorno );
     bondary4(x,y+1, cor_preenche, cor_contorno );
     bondary4(x,y-1, cor_preenche, cor_contorno );
  fim {se}
fim {procedimento boundary4}
```

Preenchimento de Áreas *(cont.)*

Algoritmo Flood-fill

- Este algoritmo permite preencher áreas definidas , não pelo contorno, mas sim pela sua região interior através de uma determinada cor.
  - Algoritmo recursivo semelhante ao *Boundary-fill*.
  - Premite recolorir regiões.
  - Necessita como parâmetros de entrada:
    - Ponto inicial
    - Cor para preenchimento da figura
    - Cor do interior



- Algoritmo

```
procedimento flood4(x, y, cor_preenche, cor_antiga : inteiro);
início
    se (inquirir_cor(x,y) = cor_antiga) então
    início
       set_pixel(x,y,cor_preenche);
       {conectividade 4}
       flood4(x+1,y, cor_preenche, cor_ antiga);
       flood4(x-1,y, cor_preenche, cor_ antiga);
       flood4(x,y+1, cor_preenche, cor_ antiga);
       flood4(x,y-1, cor_preenche, cor_ antiga);
    fim {se}
fim {procedimento flood4}
```