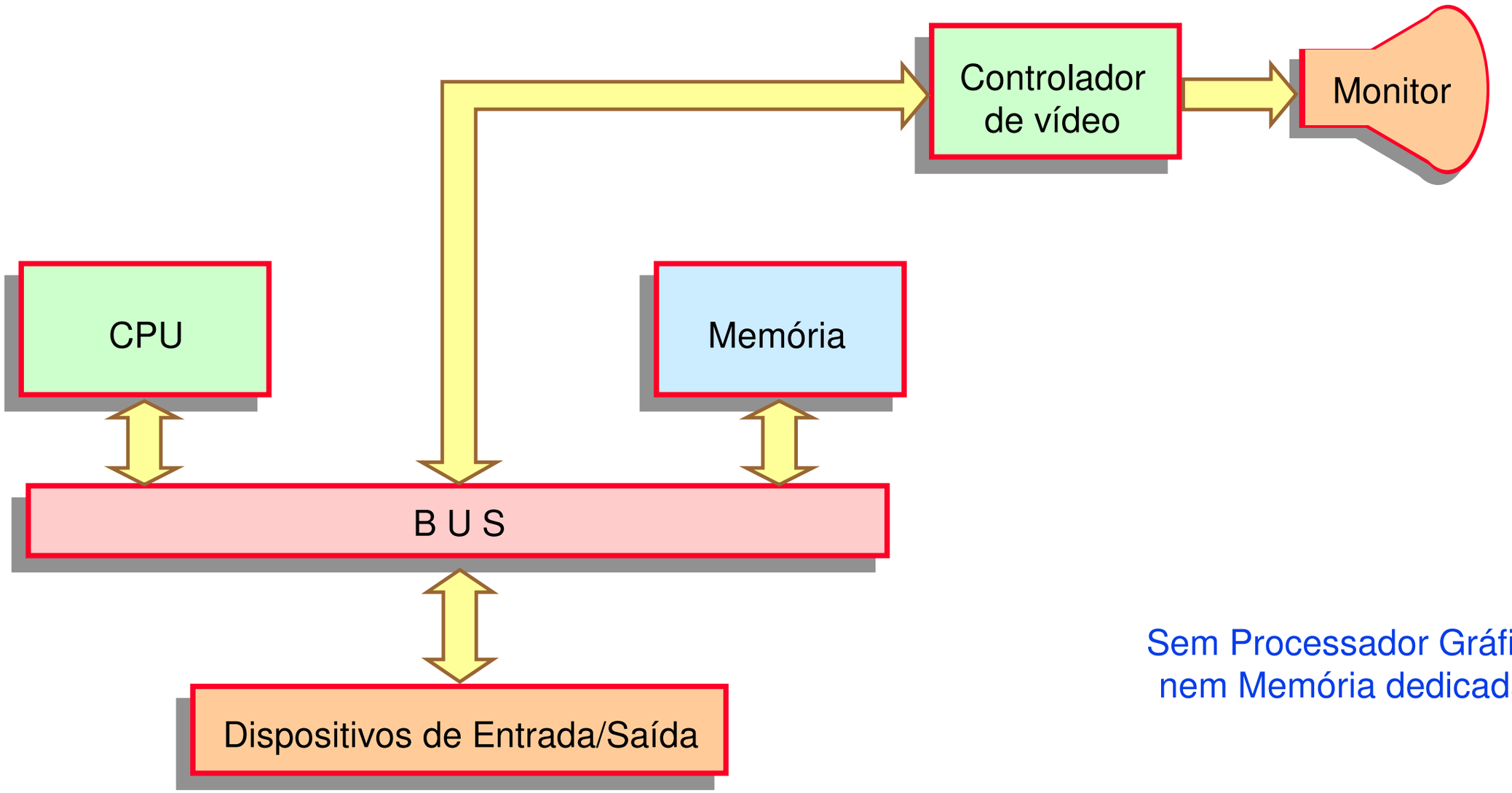
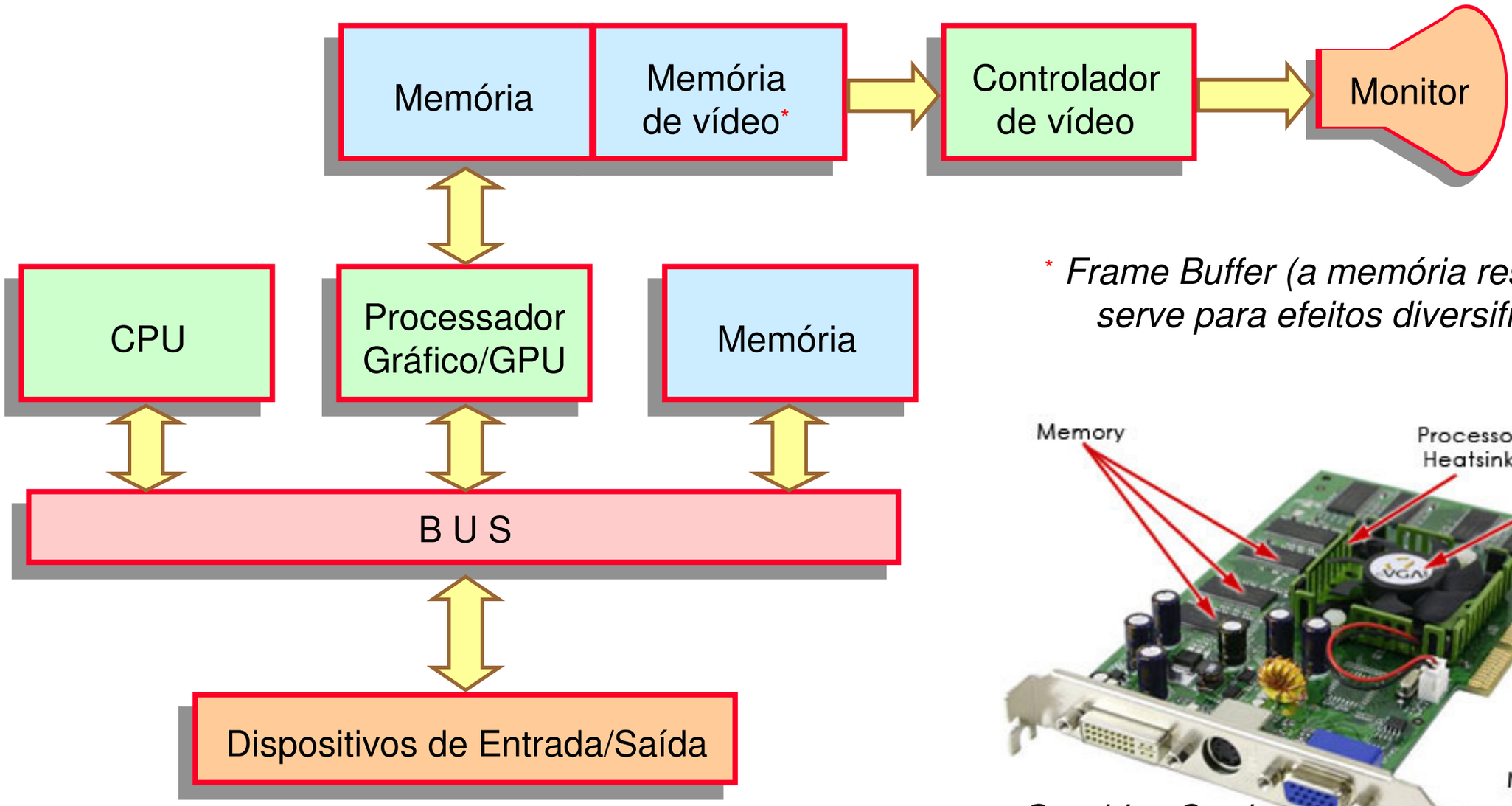
Rasterização

Arquitetura de um sistema gráfico raster



Sem Processador Gráfico
nem Memória dedicada

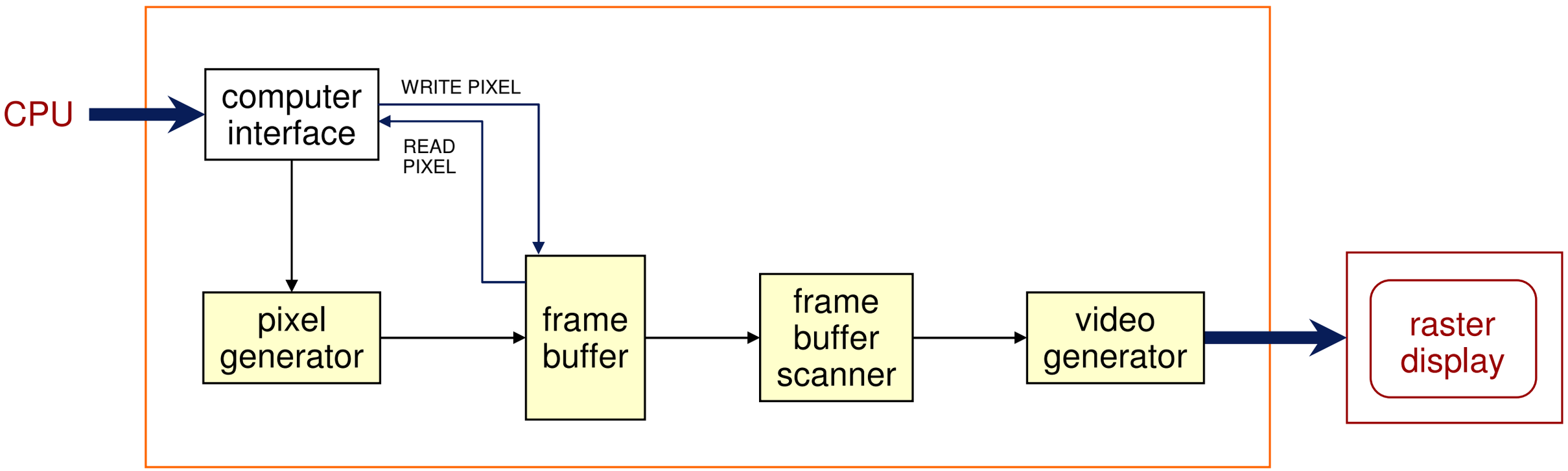
Arquitetura de um sistema gráfico raster



* *Frame Buffer (a memória restante serve para efeitos diversificados)*

Graphics Card

Diagrama funcional básico duma placa gráfica raster



Modificações dinâmicas (*feedback* da interação) requerem uma geração de pixels a, pelo menos, **10 Hz**, ao passo que o refrescamento exige **25 Hz**, no mínimo.

Funções e Componentes básicas

Principais funções duma placa gráfica

- Fornecer os sinais adequados ao sistema de visualização (*display*), interpretando os comandos vindos do CPU e construindo a imagem em pixels na memória de refrescamento.
- Executar o varrimento da memória de refrescamento e interpretar o seu conteúdo.
- Gerar os três sinais de saída independentes correspondentes às três cores primárias R, G e B.

Pixel Generator (integrado no GPU)

- Interpretador das instruções do CPU (e.g. *line*), mas escrevendo no *frame buffer* só quando o *scanner* não o estiver a utilizar (tipicamente, com um só *buffer*, no retorno vertical do feixe).

Frame Buffer

- Memória de refrescamento (*refresh buffer*), cujas dimensões em *x* e em *y* tanto podem ser potências de 2 como os valores correspondentes às dimensões do sistema de visualização.

Frame Buffer Scanner

- Módulo de varrimento da memória de refrescamento (da esquerda para a direita e de cima para baixo) com modulação do sinal de saída proporcionalmente aos valores nessa memória.

Video Generator

- Produz o sinal de vídeo para o sistema de visualização.

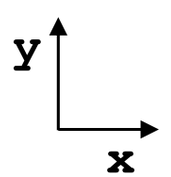
M.Próspero

RASTERIZAÇÃO DE PRIMITIVAS GRÁFICAS

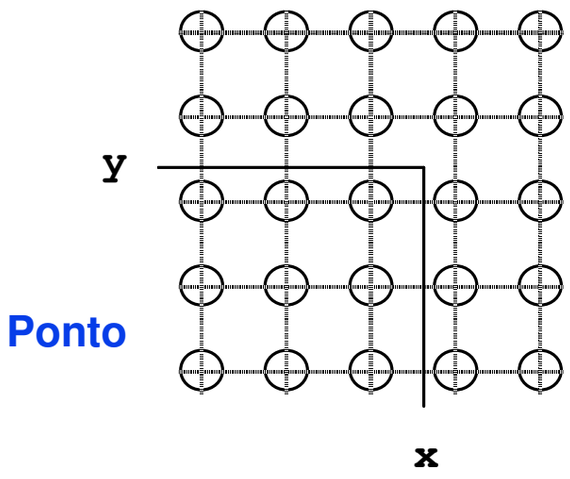
Compromisso em tecnologia Raster → **imagens de boa qualidade**

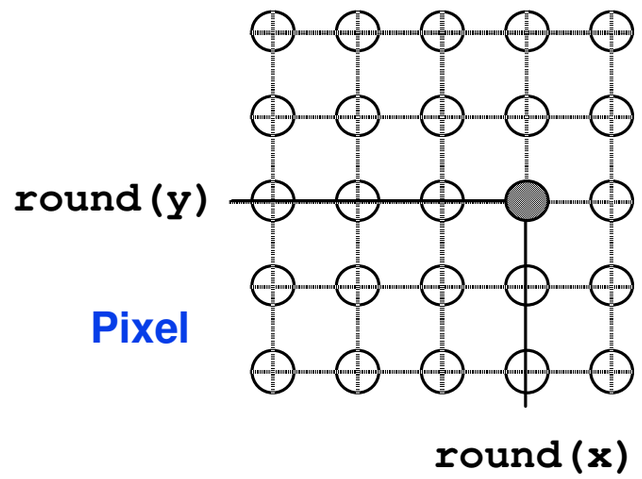
→ **algoritmos de execução rápida**

Nota: Salvo indicação em contrário, usaremos o seguinte sistema de coordenadas cartesianas



PONTOS



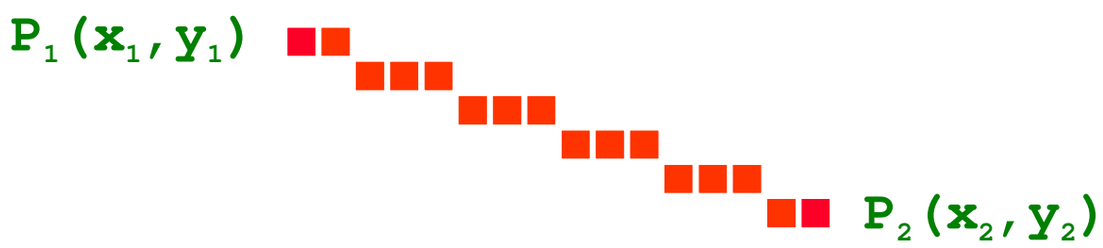



```
WRITE_PIXEL(round(x),round(y),value)
```

M.Próspero

LINHAS

Segmentos de reta definem-se pelas coordenadas dos extremos (P_1 e P_2), por hipótese valores inteiros.



A cor, a grossura e o tipo de traço são atributos possíveis para as linhas.

Há implementações desta primitiva em que o pixel correspondente a P_2 não é ativado, permitindo que não seja escrito mais do que uma vez se se tratar de uma linha poligonal (e.g. um triângulo $[P_1 , P_2 , P_3]$).

Existem modos de escrita no *frame buffer* que podem originar efeitos diferentes quando se repete a escrita de um mesmo pixel (caso do modo XOR).

M.Próspero

MODOS DE ESCRITA

Escrita de pixels (Source) em modo XOR (disjunção exclusiva) no *frame buffer* (Destination/Result)

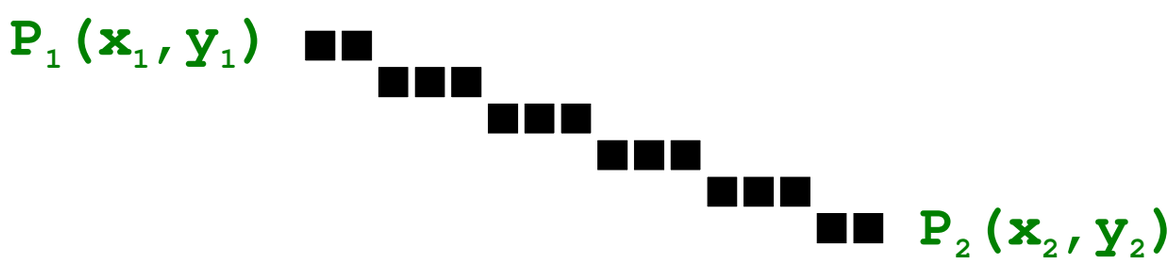
Source	Destination	Result
0	0	0
0	1	1
1	0	1
1	1	0

OBS.:
O modo por omissão costuma ser **Result**=**Source**, o que se resume a cópia (modo de cópia).

Com outras cores, fazer procedimentos independentes para R, G e B

Exemplo aplicado a linhas B&W (1 bit/pixel), usando cor W sobre fundo W:

B(lack)=0
W(hite)=1



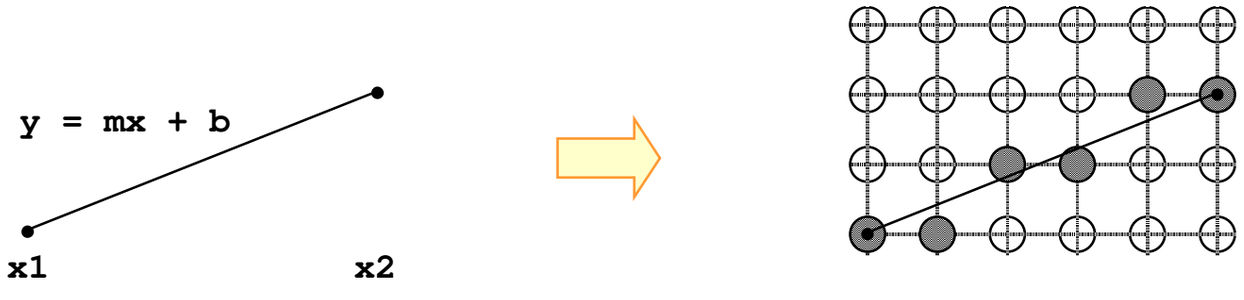
Assim, pode-se criar *rubber-banding* desenhando simplesmente 2 vezes, em cor W, o segmento a modificar (sendo a 1.ª vez sobre a posição anterior, para apagar essa versão, e a 2.ª actualizando-se apenas o ponto P_2 para a posição do cursor).

Pergunta: ***O que acontece quando uma linha de cor B, escrita em modo XOR, intersecta outra linha da mesma cor já existente no *frame buffer*?***

M.Próspero

ALGORITMOS DE RASTERIZAÇÃO DE LINHAS

Tratamento de segmentos de reta




```
for x from x1 to x2 step Δx do
    y = m*x + b
    WRITE_PIXEL(x,round(y),value)
endfor
```

Pseudo-código (a sintaxe correta dependerá da linguagem e da API gráfica)

Δx → Valor inteiro (=1)

y, m, b → Não são, em geral, valores inteiros

⇒ Há métodos algorítmicos mais eficientes!

M.Próspero

Como reduzir o tempo de execução ?

ALGORITMO INCREMENTAL (*DDA Algorithm* *)

$$y_{i+1} = m\, x_{i+1} + b = m\,(x_i + \Delta x) + b = m\, x_i + m\, \Delta x + b = y_i + m\, \Delta x$$

$$\Delta x = 1 \quad \Rightarrow \quad \Delta y = m\, \Delta x = m$$

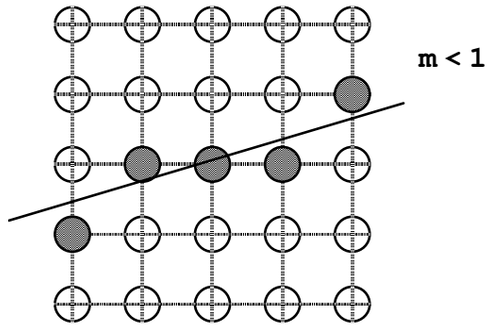
```
procedure LINE (x1,y1,x2,y2,value: integer);
                                          { when  -1 ≤ m ≤ +1 only }
var   dx,dy,x,y,m: real;

begin
      if x1<>x2 then
      begin
            dy := y2-y1; dx := x2-x1; m := dy/dx; y := y1;
            for x:=x1 to x2 do
            begin
                  WRITE_PIXEL(x,round(y),value); y := y + m
            end
      end
      else if y1=y2 then WRITE_PIXEL(x1,y1,value)
end;
```

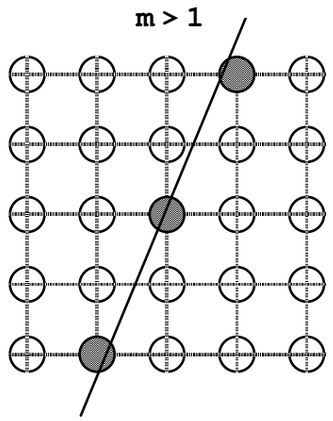
* *Digital Differential Analyser Algorithm*

M.Próspero

ALGORITMO INCREMENTAL

E se for $m > 1$?





$m > 1 \Rightarrow \Delta y > 1 \Rightarrow$ **Ficam pixels por intensificar !**

Solução: Troca de variável independente no algoritmo, resultando $\Delta y = 1$ e $\Delta x = 1/m$.

Avaliação do algoritmo:

⊕ **Eliminação do produto $m*x$**

⊖ **Acumulação de erros !!**

Há algoritmos melhores do que este!

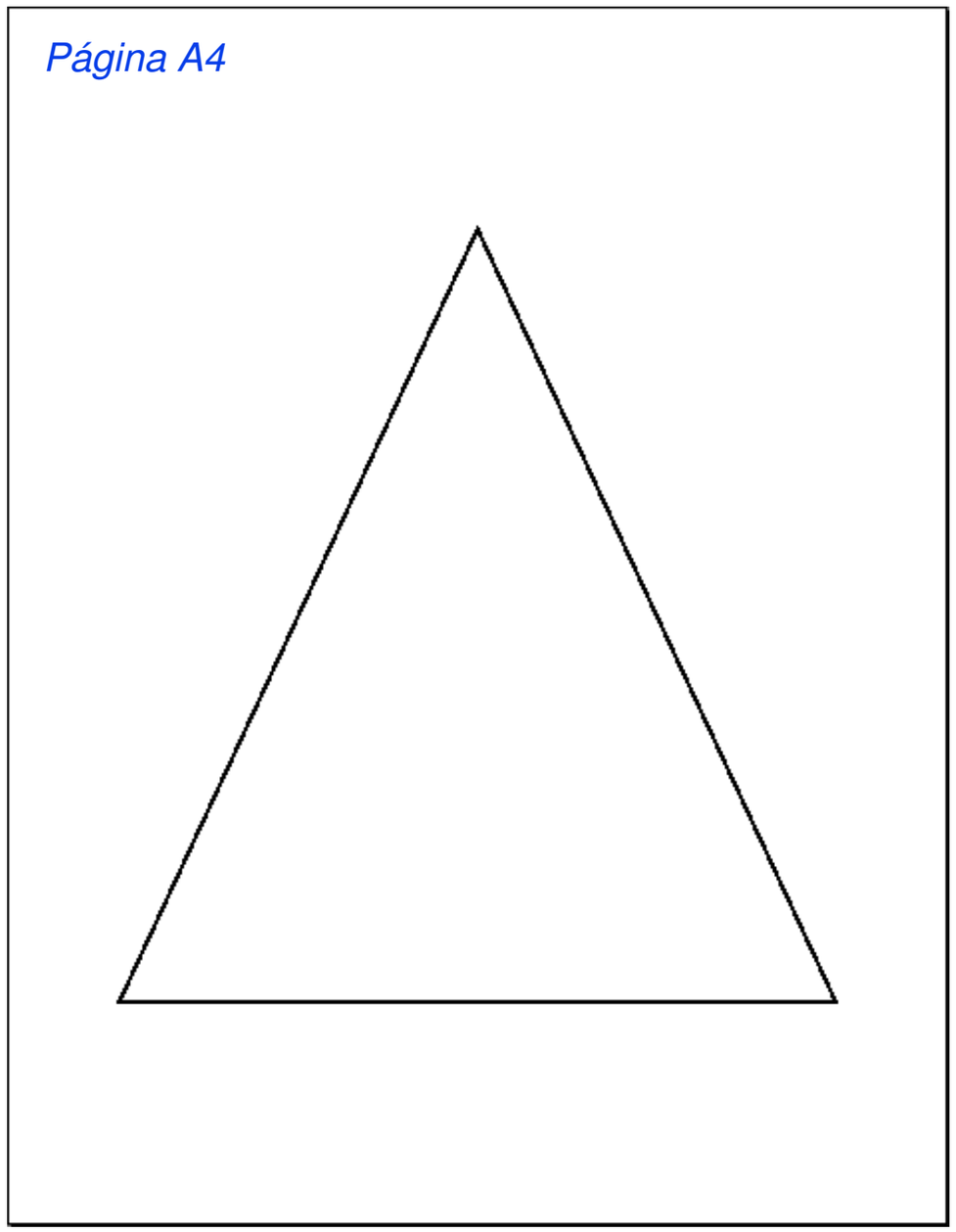
M.Próspero

Outros exemplos de utilização de primitivas gráficas (nível API)

Programação em PostScript

```
%!PS
% Triangle_1
% Triângulo centrado em página A4
% ( unidades: 72 dpi )
72 144 moveto
306 648 lineto
540 144 lineto
closepath
stroke
showpage
```

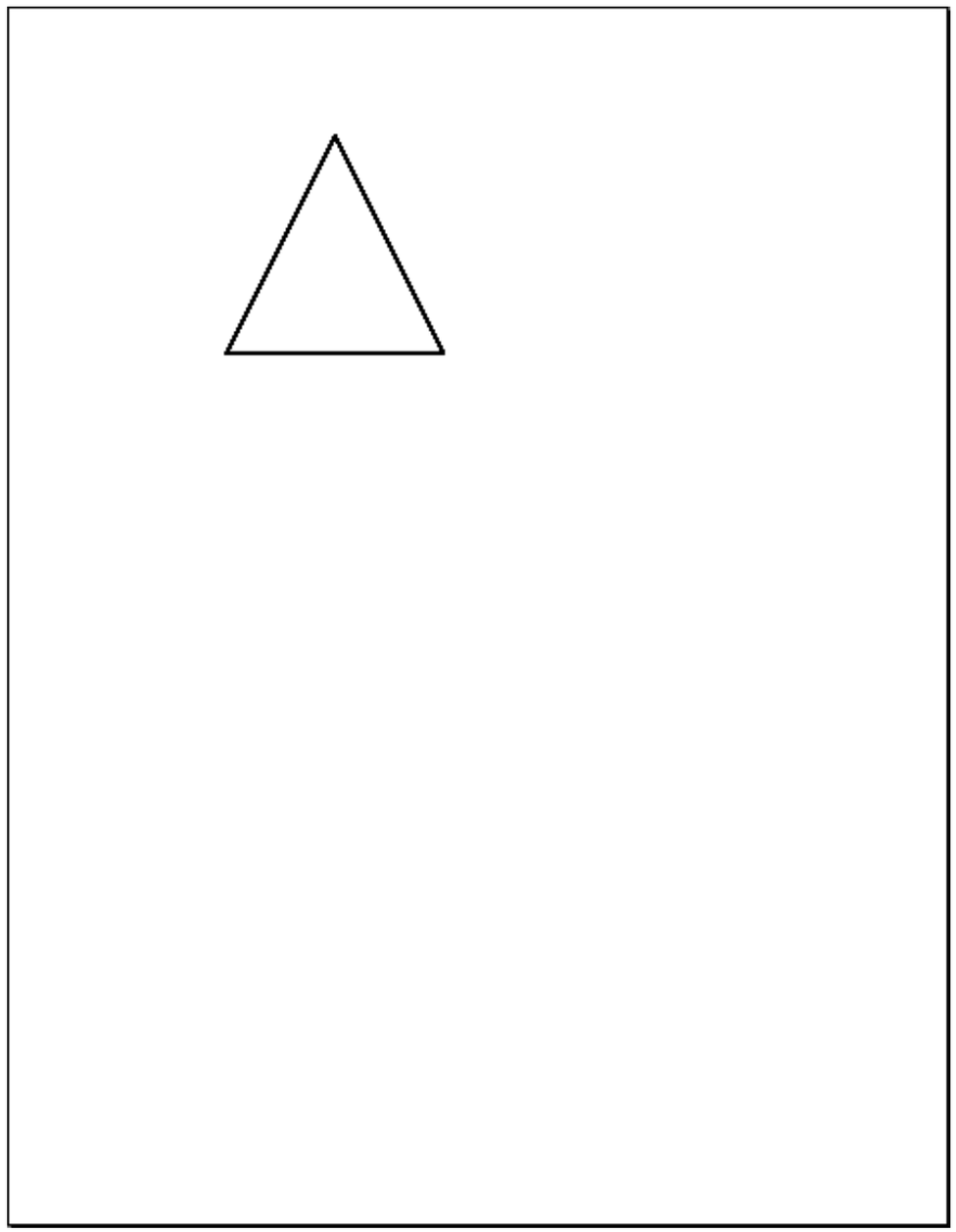
M.Próspero



Programação em PostScript

```
%!PS
% Triangle_1
% Triângulo centrado em página A4
% ( unidades: 72 dpi )
72 144 moveto
306 648 lineto
540 144 lineto
closepath
stroke
showpage
```

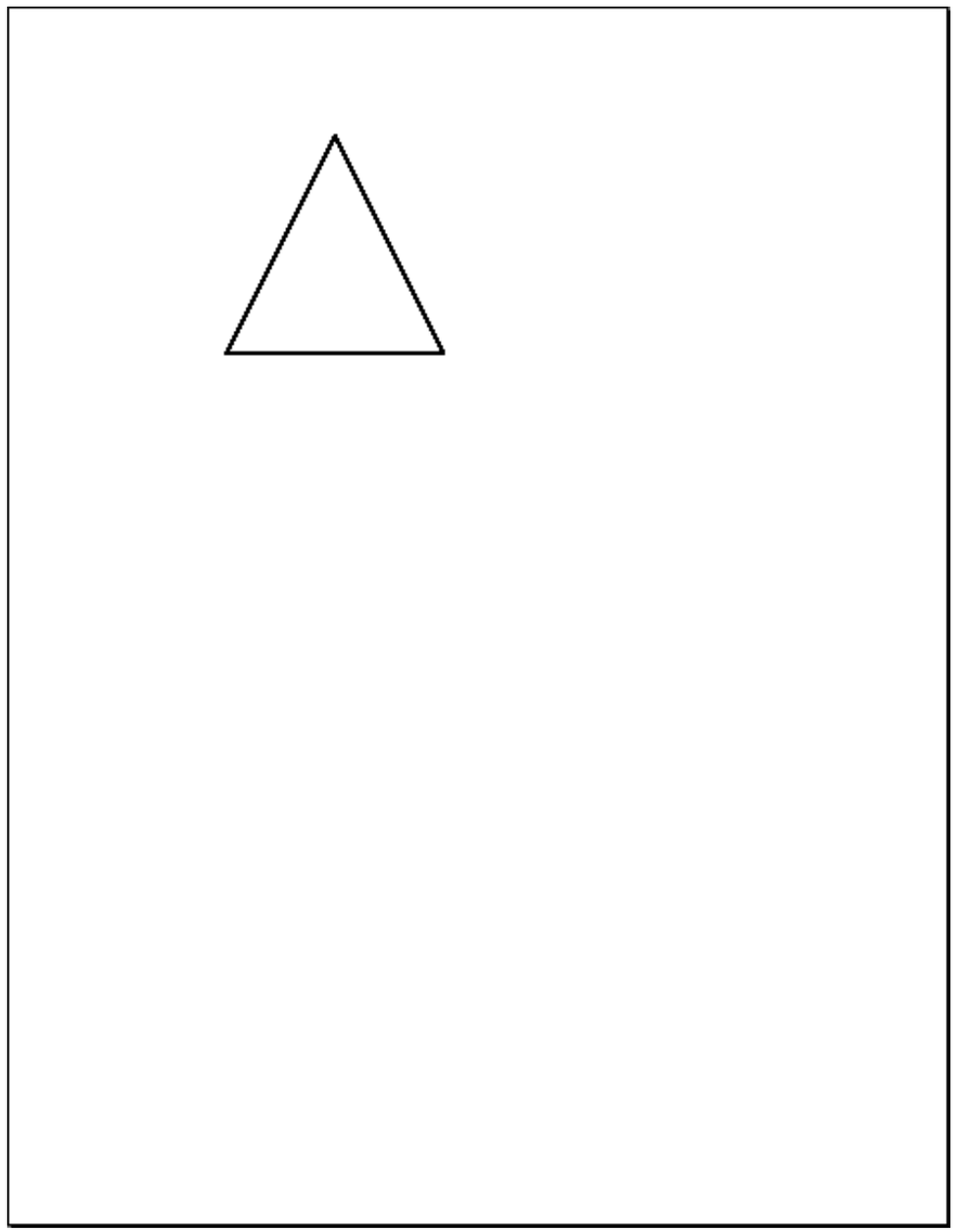
M.Próspero



Programação em PostScript

```
%!PS
% Triangle_2
% Triângulo no topo da página
/cm {28.35 mul} def
5 cm 20 cm moveto
7.5 cm 25 cm lineto
10 cm 20 cm lineto
closepath
stroke
showpage
```

M.Próspero



Programação em PostScript

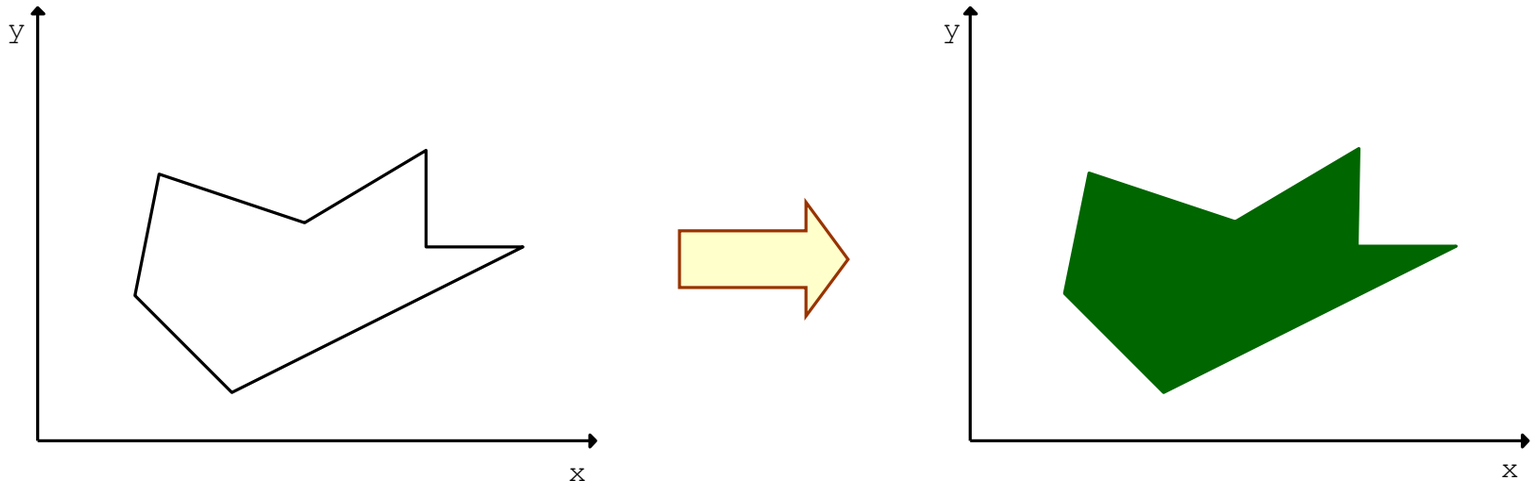
```
%!PS
% Triangle_3
% A mesma figura que Triangle_2
/cm {28.35 mul} def
5 cm 20 cm moveto
7.5 cm 25 cm lineto
10 cm 20 cm lineto
5 cm 20 cm lineto
stroke
showpage
```

M.Próspero

POLÍGONOS

FILL AREA

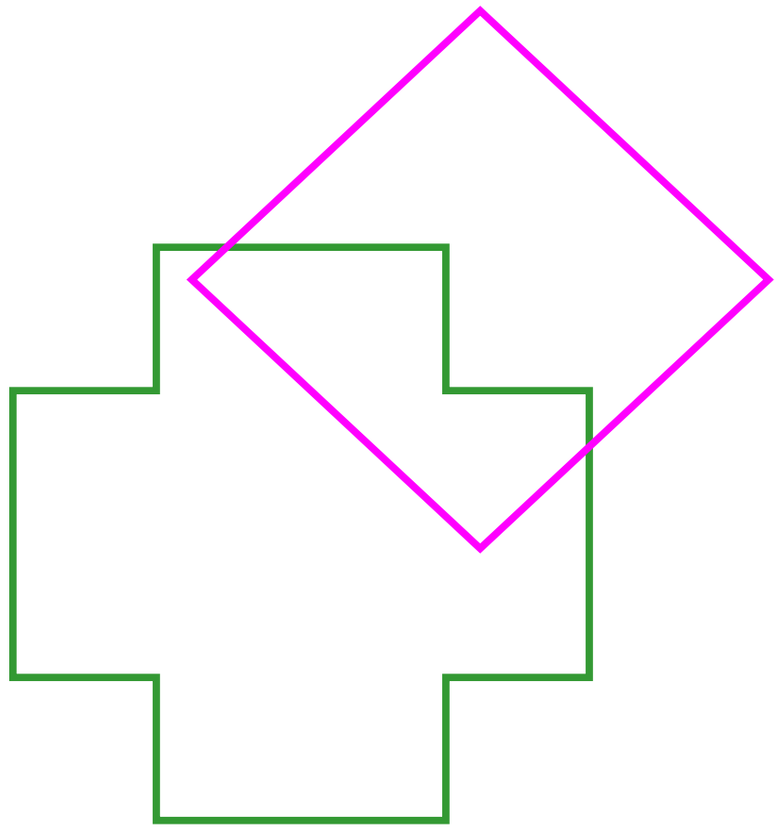
Objetivo: **Preenchimento de um polígono definido pela <u>sequência</u> dos respetivos <u>vértices</u> (pontos 2D), independentemente dos valores preexistentes na memória de refrescamento.**



M.Próspero

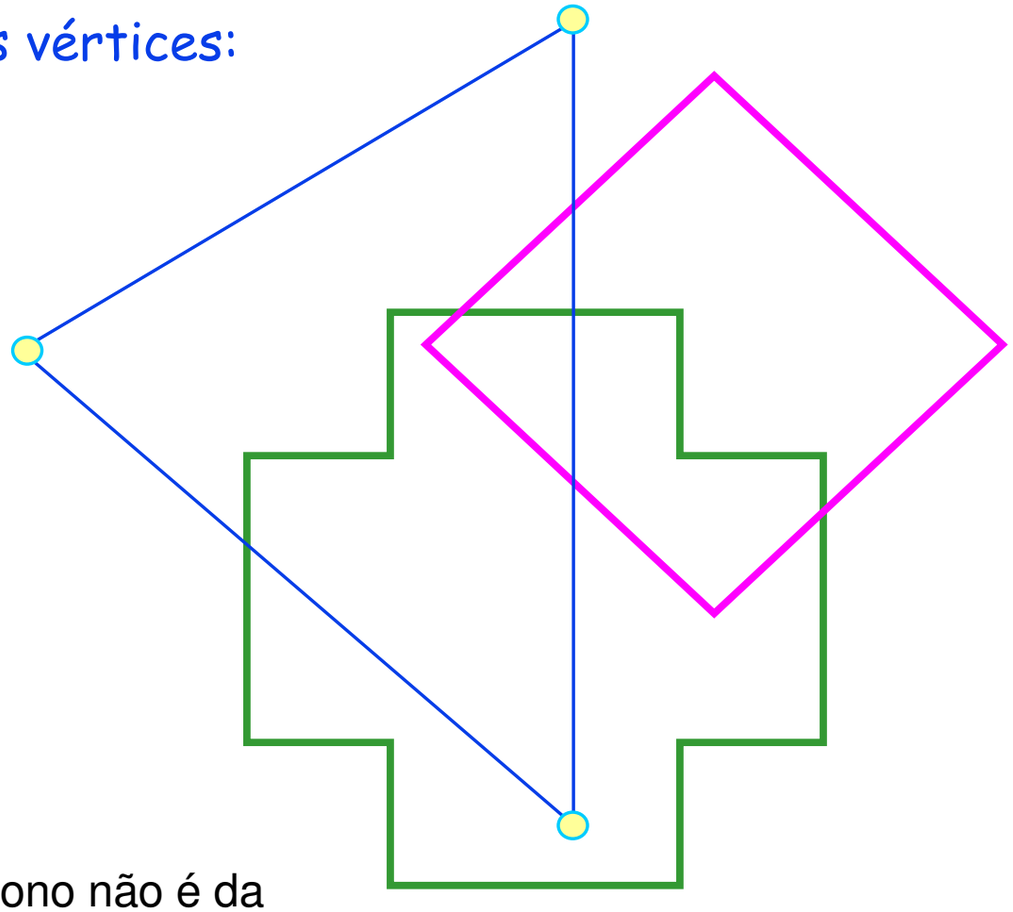
FILL AREA

Exemplo de ecrã numa situação inicial:



FILL AREA

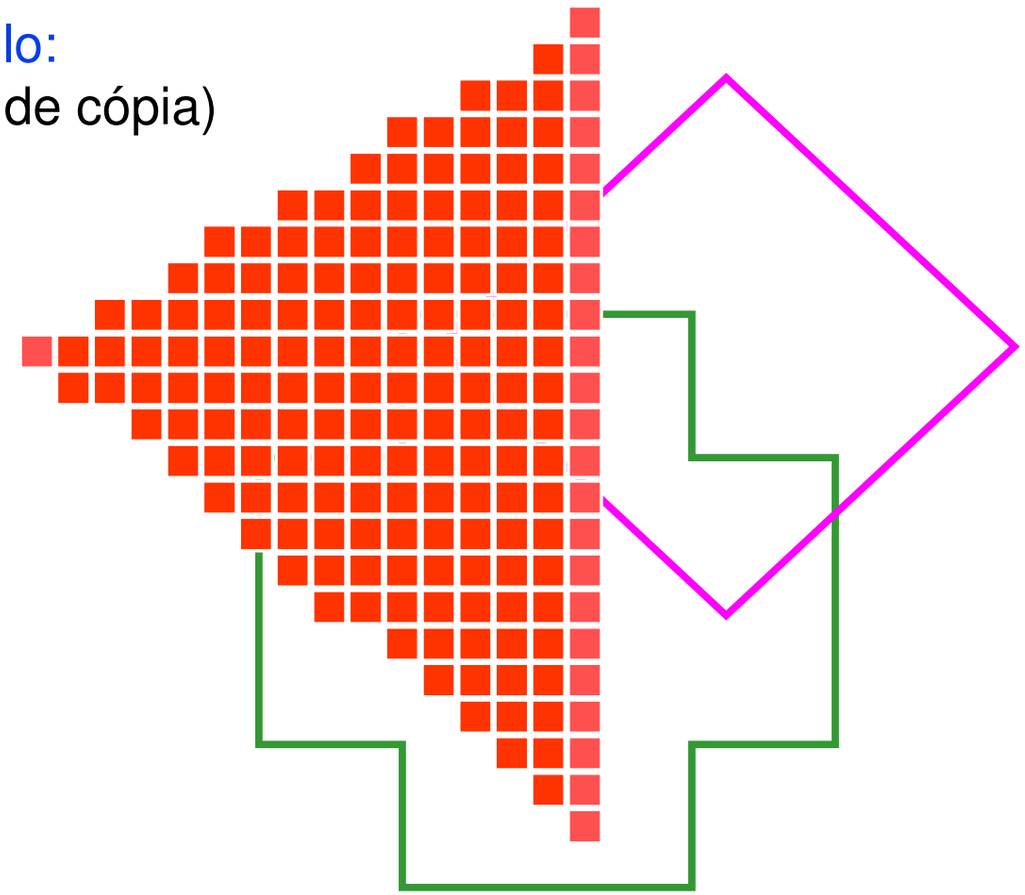
Novo polígono, definido pelos seus vértices:



NB: O desenho da fronteira do polígono não é da responsabilidade do algoritmo de FILL_AREA .

FILL AREA

Preenchimento do triângulo:
(em baixa resolução e em modo de cópia)



FILL AREA

Resultado do preenchimento:
(na resolução do ecrã)