

# Computação Gráfica - Recorte

Profa. Mercedes Gonzales  
Márquez

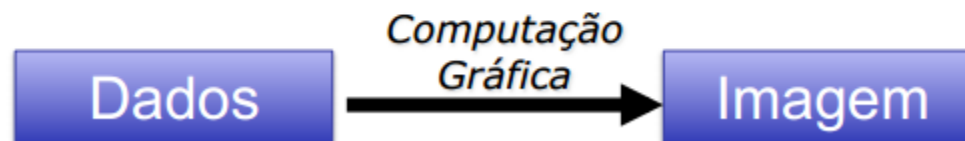
---

# Tópicos

- Recapitulando...
- Conceito de Recorte
- Recorte de Pontos
- Recorte de Segmentos em Regiões Planares
  - Algoritmo de Cohen Sutherland
  - Algoritmo de Cyrus-Beck
- Recorte de Polígonos em Regiões Planares
  - Algoritmo de Sutherland-Hodgeman

## Recapitulando... (Conceito CG)

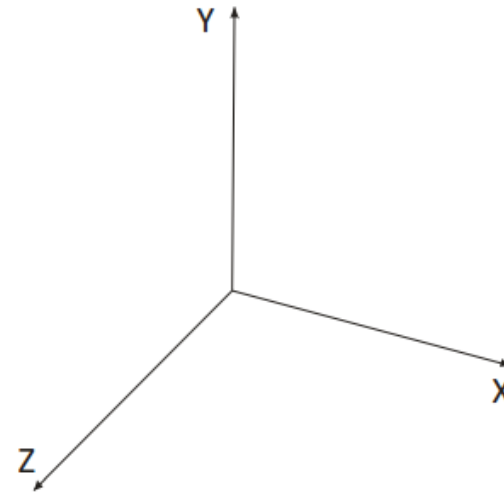
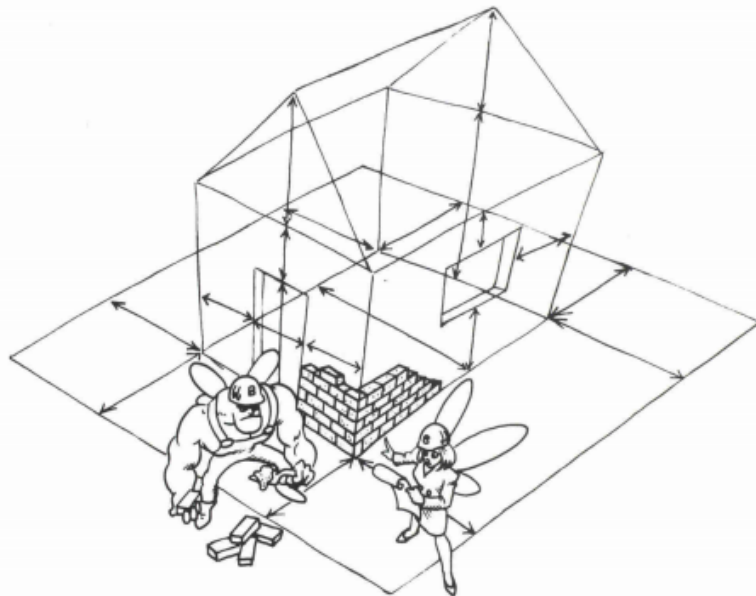
“Computação Gráfica é o conjunto de métodos e técnicas para transformar dados em imagem através de um dispositivo gráfico.”



# Recapitulando... (Modelagem Geométrica)

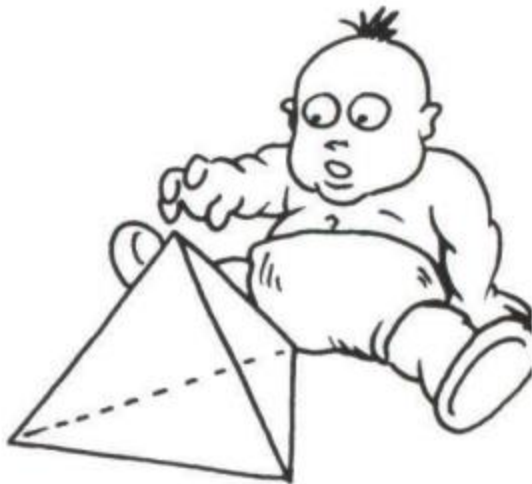
O que é Modelagem Geométrica?

Estruturar e descrever dados geométricos no computador

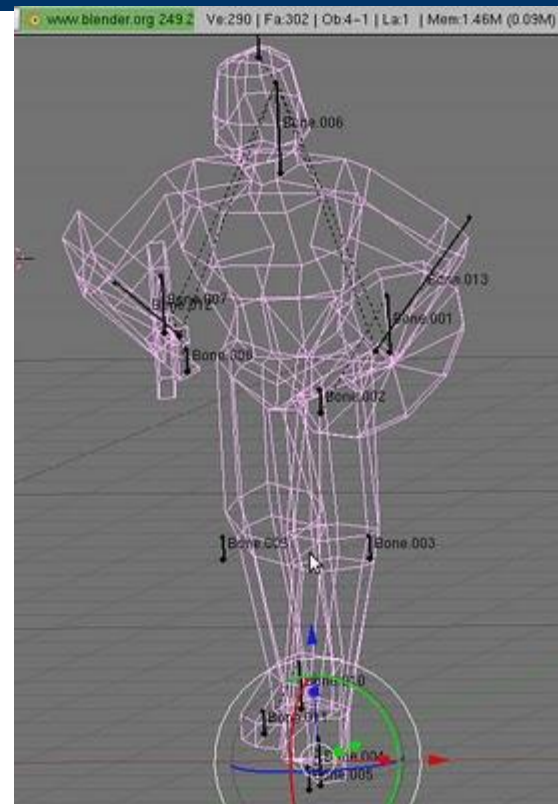
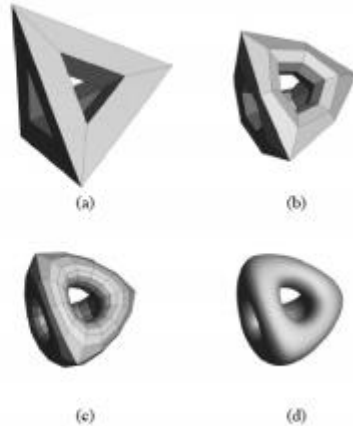
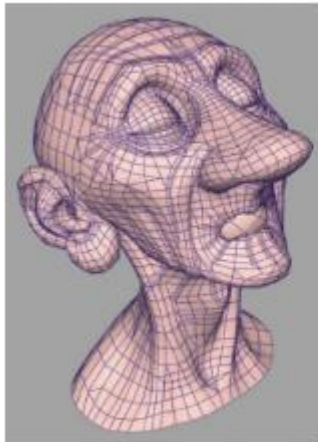


# Recapitulando... (Modelagem Geométrica)

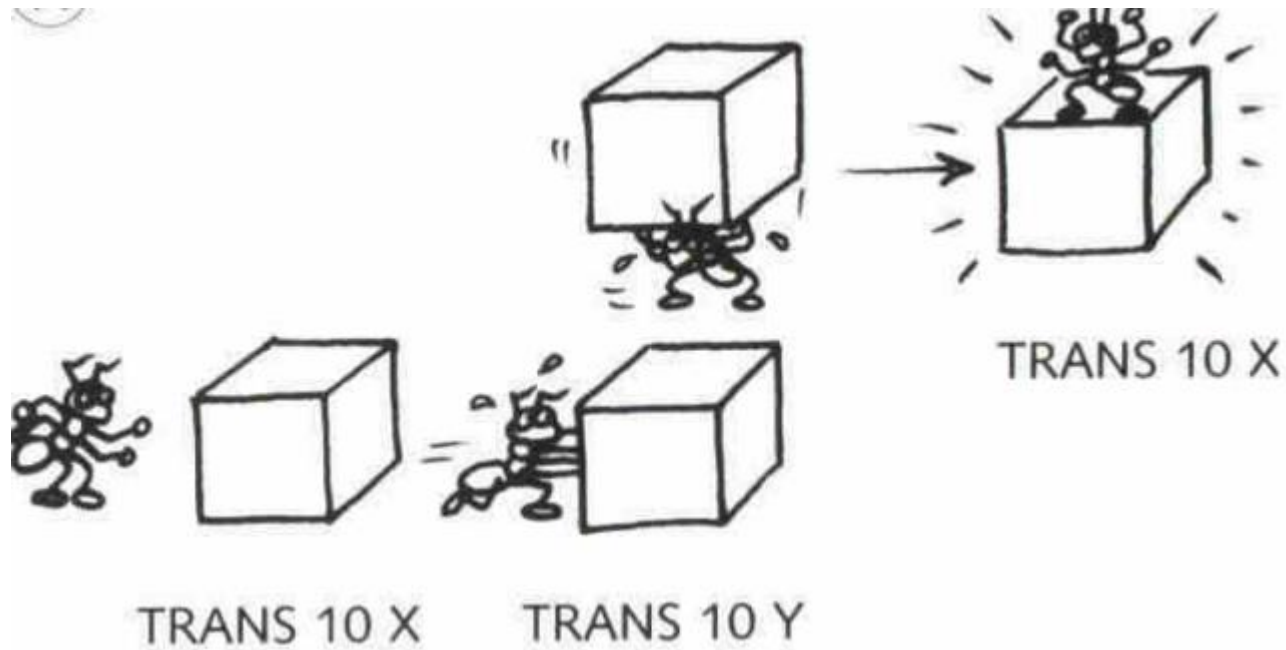
Objetos são definidos por pontos, linhas e planos



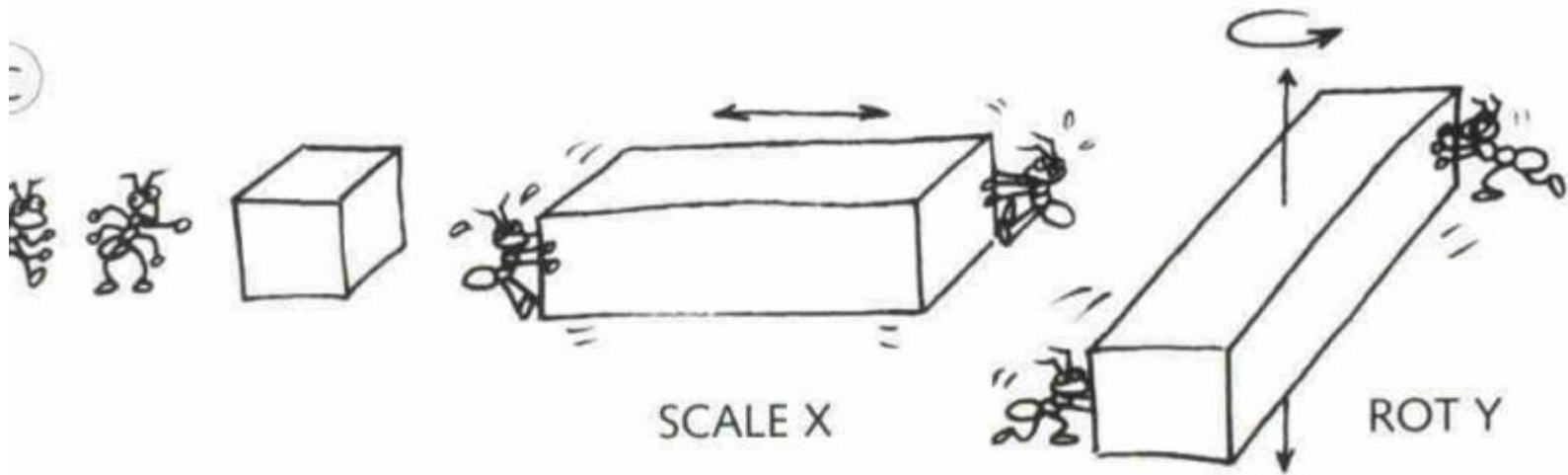
# Recapitulando... (Modelagem Geométrica)



# Recapitulando ... (Transformações Geométricas)



# Recapitulando ... (Transformações Geométricas)





## Recapitulando ...

MODELAGEM GEOMETRICA 3D  
+  
TRANSFORMAÇÕES GEOMÉTRICAS  
=  
CENÁRIO 3D

## Recapitulando ...

CENÁRIO 3D



IMAGEM

SERÁ NECESSÁRIO

- PROJEÇÃO
- RECORTE
- AMOSTRAGEM
- REMOÇÃO DE SUPERFÍCIES ESCONDIDAS (VISUALIZAÇÃO)
- COLORIZAÇÃO (ILUMINAÇÃO E TEXTURIZAÇÃO)

# Recapitulando ...

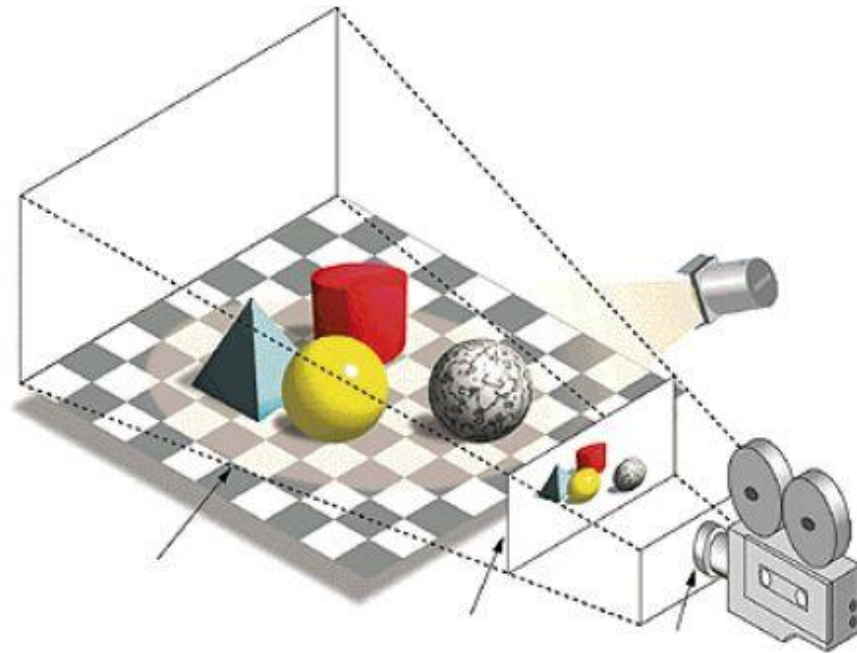
CENÁRIO 3D



IMAGEM

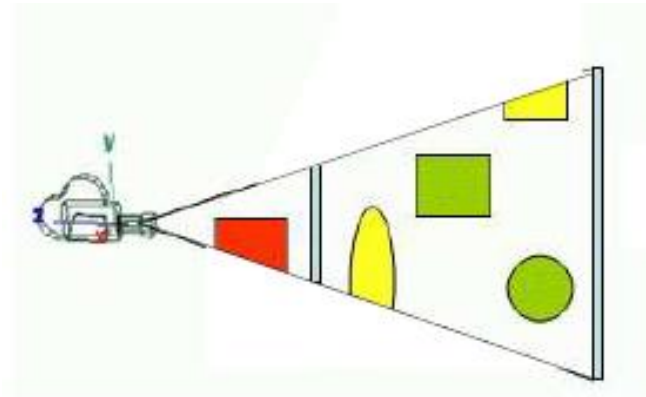
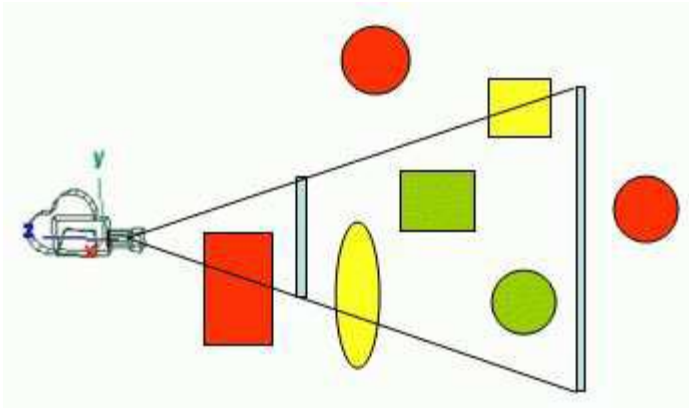
SERÁ NECESSÁRIO:

- RECORTE
- PROJEÇÃO
- AMOSTRAGEM
- REMOÇÃO DE SUPERFÍCIES ESCONDIDAS (VISUALIZAÇÃO)
- COLORIZAÇÃO (ILUMINAÇÃO E TEXTURIZAÇÃO)



# Recorte

- A técnica de recorte consiste na remoção das partes que não estejam dentro do volume de visão.
- Somente as figuras geométricas contidas na janela de visão devem aparecer.



## Recorte

- Remover os pontos que estão fora do volume de visão se reduz, numericamente, em um problema de interseção entre os seus planos limitantes e as figuras geométricas de uma cena e classificação do resultado.

# Recorte de Pontos

- Objeto:  $(x,y,z)$
- Região Recortante:
  - Região Retangular:  
 $(x_{\min},x_{\max},y_{\min},y_{\max})$
  - Volume Paralelepipedal:  
 $(x_{\min},x_{\max},y_{\min},y_{\max},z_{\min},z_{\max})$

# Recorte de Pontos

- Objeto:  $(x,y,z)$
- Região Recortante:
  - Região Retangular:  
 $(x_{min},x_{max},y_{min},y_{max})$
- Um ponto  $(x, y)$ , *estará dentro do retângulo de visualizacao*, se:

$$x_{min} \leq x \leq x_{max}$$

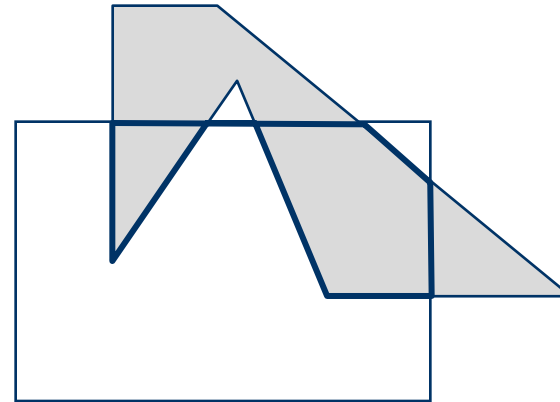
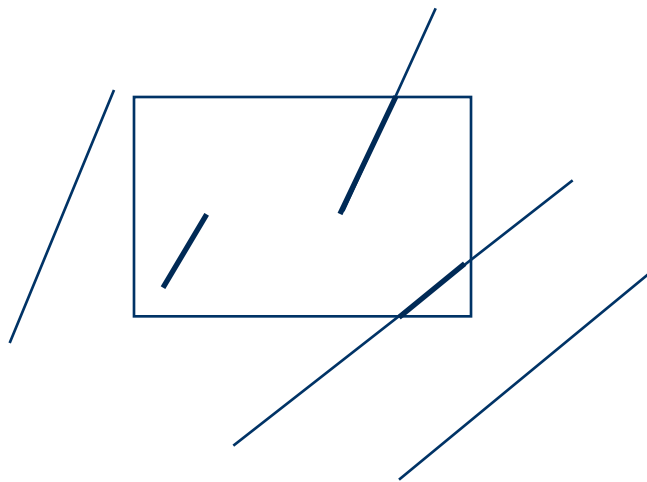
$$y_{min} \leq y \leq y_{max}$$

# Recorte de Segmentos em Regiões Planares

- Objeto: segmentos de reta
- Região Recortante R:
  - Região Retangular:  
(xmin,xmax,ymin,ymax)
- Algoritmos de Recorte de Segmentos em Regiões Planares ou Recorte 2D
  - Algoritmo de Cohen Sutherland
  - Algoritmo de Cyrus-Beck

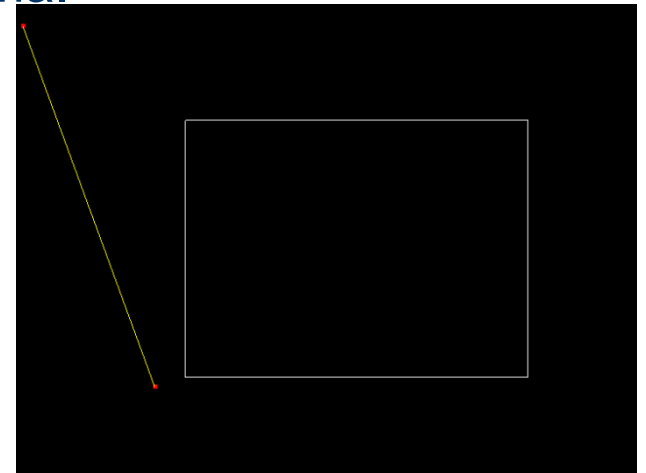
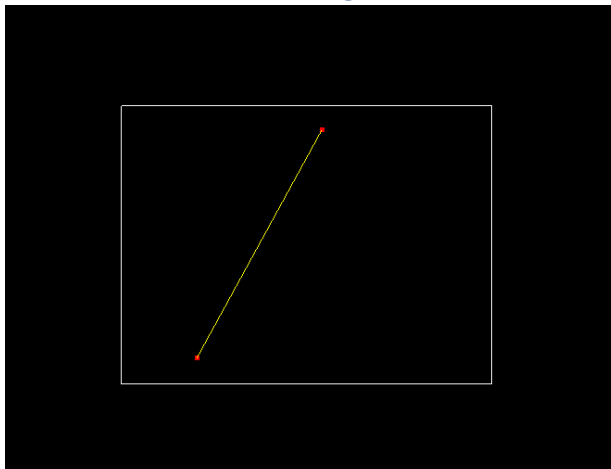


# Recorte de Segmentos em Regiões Planares ou Recorte 2D



# Algoritmo de Cohen-Sutherland

- O algoritmo de Cohen-Sutherland se baseia em dois fatos triviais:
  - um segmento está totalmente contido em uma região, se e somente se, seus dois vértices estiverem contidos nela,
  - um segmento está totalmente fora de uma região, se seus dois vértices estiverem em um semi-plano definido a partir de uma aresta da região e para fora da mesma.

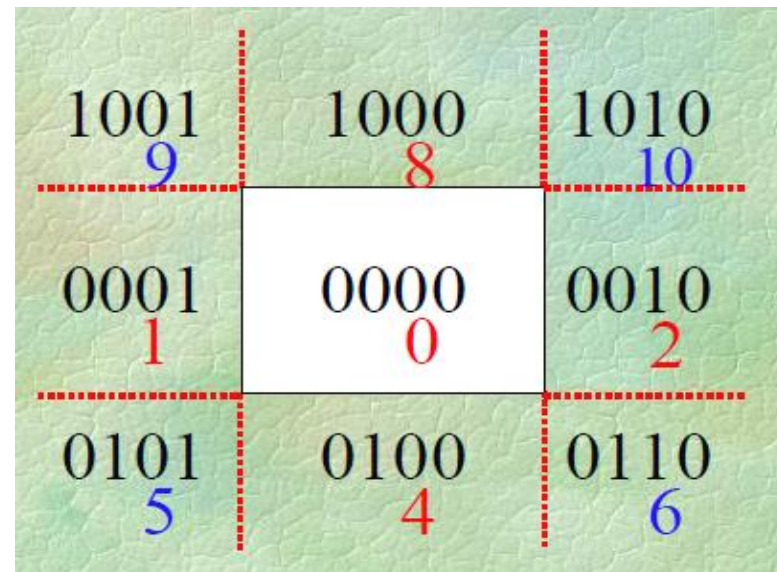


# Algoritmo de Cohen-Sutherland

- Temos 4 sub-espacos, um a partir de cada aresta da região recortante R.
  - Subespaço superior:  $y > y_{max}$
  - Subespaço inferior:  $y < y_{min}$
  - Subespaço direito:  $x > x_{max}$
  - Subespaço esquerdo:  $x < x_{min}$

# Algoritmo de Cohen-Sutherland

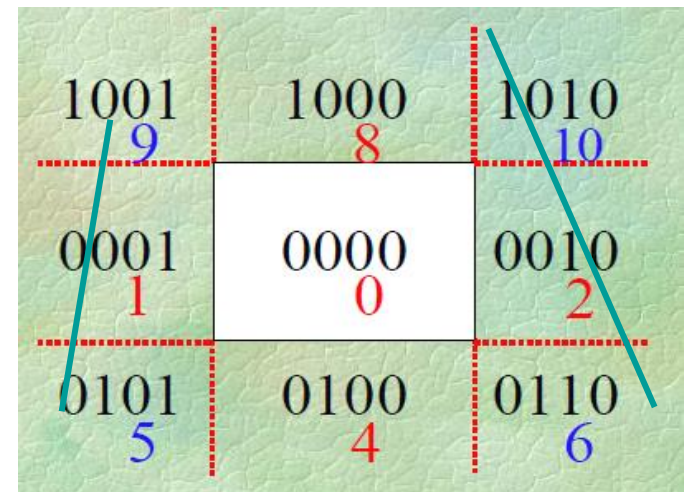
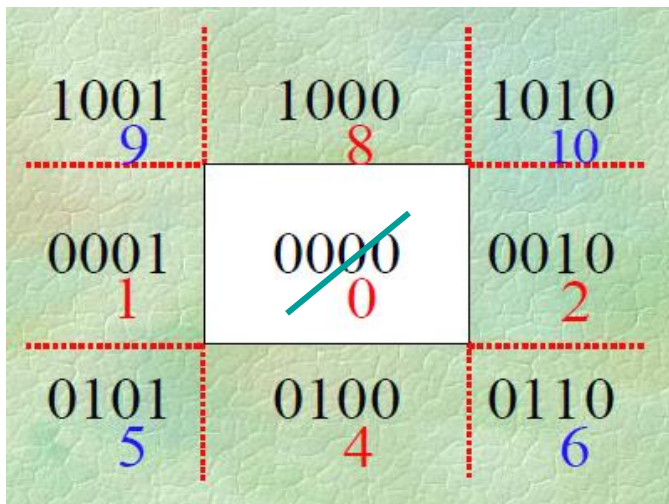
- Cria-se um código de 4 bits para cada vértice do segmento de acordo com sua disposição em relação à região R.
- O bit é 0, se o vértice estiver dentro do subespaço e 1 se estiver fora.
- Subespaço superior:  $y > y_{max}$  (1o bit)
- Subespaço inferior:  $y < y_{min}$  (2o bit).
- Subespaço direito:  $x > x_{max}$  (3o bit)
- Subespaço esquerdo:  $x < x_{min}$  (4o bit)



# Algoritmo de Cohen-Sutherland

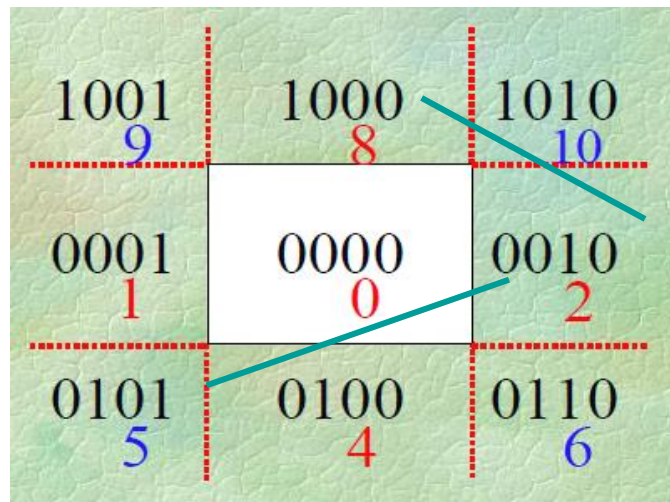
- Se aplicarmos uma operação lógica AND, bit a bit, entre os códigos dos dois vértices de um segmento teremos as seguintes possíveis situações:

- O código dos vértices é 0000: o segmento está contido na região.
- O resultado é diferente de zero: o segmento está totalmente fora da região.



# Algoritmo de Cohen-Sutherland

3. o resultado é 0000 embora os códigos dos vértices não sejam: é indecidível a pertinência do segmento à região.



- O segmento pode estar fora da região ou
- O segmento pode estar parcialmente contido na região.

# Algoritmo de Cohen-Sutherland

- Na situação 3, o segmento  $P1P2$  é subdividido (e descartado), da direção do ponto exterior  $P1$  para interior, até que todos fiquem decidíveis.
- Para subdividir, substituímos os valores  $xmin$ ,  $xmax$ ,  $ymin$  e  $ymax$  em uma das seguintes equações da reta a fim de obter as interseções sucessivamente.

# Algoritmo de Cohen-Sutherland

```
unsigned char code(double x, double y, double xmin,
                  double xmax, double ymin, double ymax){
    unsigned char code=0;
    if (y > ymax) code += 8;
    if (y < ymin) code += 4;
    if (x > xmax) code += 2;
    if (x < xmin) code += 1;

    return code;
}
```



# Algoritmo de Cohen-Sutherland

```
void CohenSutherlandLineClip(double x0, double y0, double x1, double y1,
double xmin, double xmax, double ymin, double ymax)
{
    unsigned char outcode0, outcode1, outcodeOut;
    double x, y;    boolean accept = FALSE, done = FALSE;

    outcode0 = code(x0, y0, xmin, xmax, ymin, ymax);
    outcode1 = code(x1, y1, xmin, xmax, ymin, ymax);

    do {
        if (outcode0 == 0 && outcode1 == 0) {
            accept = TRUE;    done = TRUE;                /* trivial draw and
exit */
        } else if((outcode0 & outcode1) != 0) {
            done = TRUE;                /* trivial reject and
exit */
        } else {
            /* discart an out part
*/
```

# Algoritmo de Cohen-Sutherland

```
    outcodeOut = (outcode0 != 0) ? outcode0 : outcode1;          /* pick
an out vertice */
    if (outcodeOut & 8) {                                         /*
discart top */
        x = x0 + (x1 - x0) * (ymax - y0) / (y1 - y0); y = ymax;
    } else if(outcodeOut & 4) {                                     /*
discart bottom */
        x = x0 + (x1 - x0) * (ymin - y0) / (y1 - y0); y = ymin;
    } else if(outcodeOut & 2) {                                     /*
discart right */
        y = y0 + (y1 - y0) * (xmax - x0) / (x1 - x0); x = xmax;
    } else if(outcodeOut & 1) {                                     /*
discart left */
        y = y0 + (y1 - y0) * (xmin - x0) / (x1 - x0); x = xmin;
    }
}
```

# Algoritmo de Cohen-Sutherland

```
if (outcodeOut == outcode0) {  
    x0 = x; y0 = y; outcode0 = code(x0, y0, xmin, xmax, ymin, ymax);  
} else {  
    x1 = x; y1 = y; outcode1 = code(x1, y1, xmin, xmax, ymin, ymax);  
}  
  
}  
} while (!done);  
  
if (accept) DrawLineReal(x0, y0, x1, y1);  
}
```

# Algoritmo de Cohen-Sutherland

1. Acrescente no programa CohenSutherland.cpp, a opção para que a janela recortante seja interativamente redimensionada. Considere apenas a modificação do canto superior esquerdo (SE) e do canto inferior direito (ID) da janela recortante. Os outros dois cantos deverão acompanhar os valores de SE e ID.

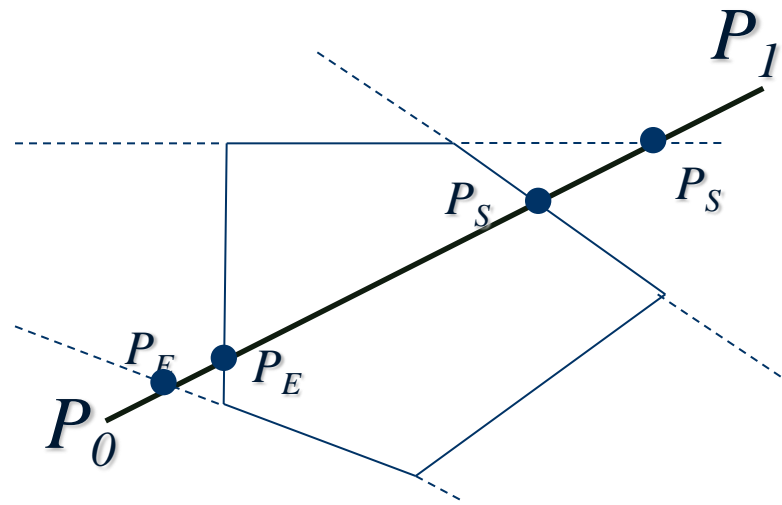
# Algoritmo de Cyrus-Beck

- Objeto: segmentos de reta
- Região Recortante:
  - Convexa
- Princípios Básicos:
  - Uso do vetor normal das arestas da região de recorte para determinar de forma simples as interseções com o segmento.
  - Uso do mesmo vetor normal para determinar se o segmento está entrando ou saindo da região de recorte.

# Algoritmo Cyrus-Beck

1. Determinar os valores do parâmetro  $t$  no qual o segmento  $P_0P_1$  intersesta cada uma das  $n$  arestas da região de recorte. As retas suporte do segmento ou das arestas poderão ser necessárias.
2. Classificar as interseções: Classifique as interseções como PE (Potencialmente entrando) ou PS (Potencialmente saindo)
3. Fazer o recorte se existir: Ache o PE com maior  $t$  (PEm) e o PS com menor  $t$  (PSM)  
-Se  $PEm < PSM$  recorte nesses 2 pontos

# Algoritmo Cyrus-Beck

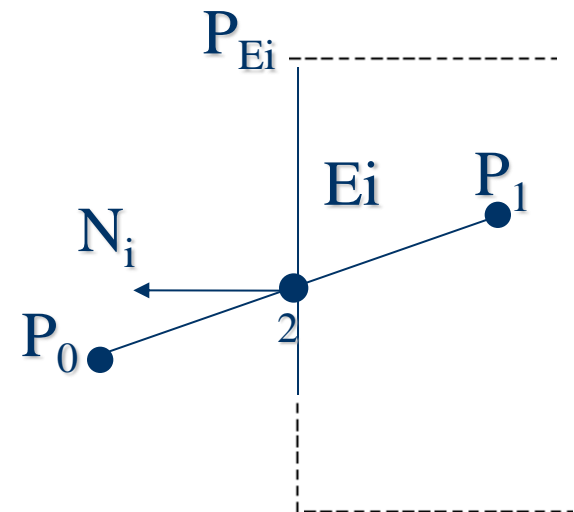


# Algoritmo Cyrus-Beck – Vetor normal

- Seja o vetor normal  $N_i$  (apontando para fora) da aresta  $E_i$  da região de recorte e seja  $P_{E_i}$  um ponto sobre  $E_i$ .
- Seja  $P(t) = P_0 + t(P_1 - P_0)$ ,  $0 \leq t \leq 1$ , a equação do segmento  $S$  unindo  $P_0$  e  $P_1$ .
- Encontramos a interseção do segmento  $P(t)$  com a aresta  $E_i$ , fazendo:

$$\langle N_i, P(t) - P_{E_i} \rangle = 0$$

- Onde  $N_i$  e  $P_{E_i}$  são respectivamente o vetor normal e um ponto da aresta  $E_i$ .





# Determinando a interseção

- Então haverá interseção do segmento com a aresta  $E_i$

- Se  $P_0$  não coincidir com  $P_1$ .

$$P_0 \neq P_1$$

- Se o segmento não for paralelo à aresta

$$\langle N_i, P_1 - P_0 \rangle \neq 0$$

- Se o ponto de interseção estiver dentro do segmento.

$$0 \leq t \leq 1$$

- Determinação do  $t$  no qual haverá interseção.

- Condição para a interseção

$$\langle N_i, P(t) - P_{E_i} \rangle = 0$$

$$\langle N_i, P_0 + t(P_1 - P_0) - P_{E_i} \rangle = 0$$

$$\langle N_i, P_0 - P_{E_i} \rangle + \langle N_i, t(P_1 - P_0) \rangle = 0$$

$$t = \frac{\langle N_i, P_0 - P_{E_i} \rangle}{-\langle N_i, P_1 - P_0 \rangle}$$

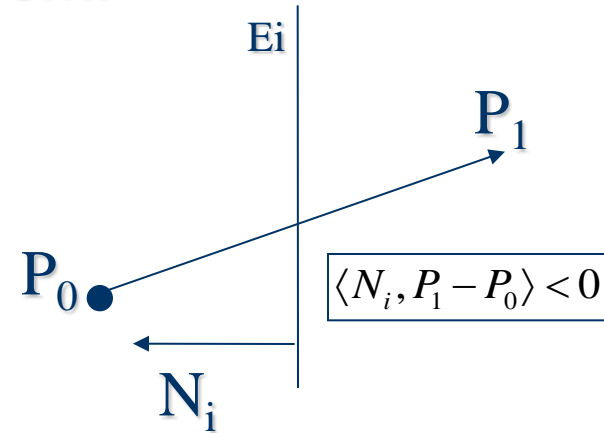
# Classificando a interseção

- Classificamos as interseções restantes em:

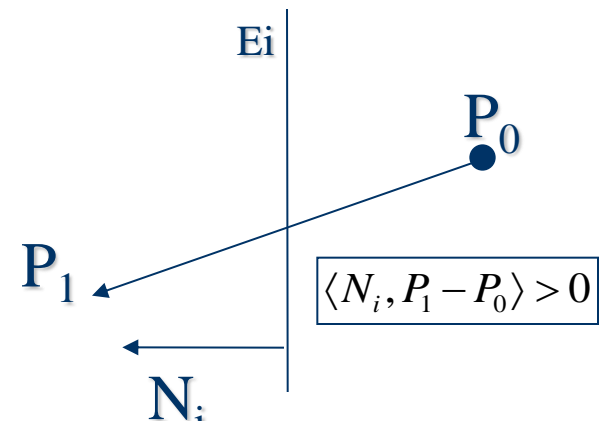
- Potencialmente Entrando (PE)

- Potencialmente Saindo (PL)

- $\langle N_i, P_1 - P_0 \rangle < 0$  implica PE  $\rightarrow$



- $\langle N_i, P_1 - P_0 \rangle > 0$  implica PL  $\rightarrow$

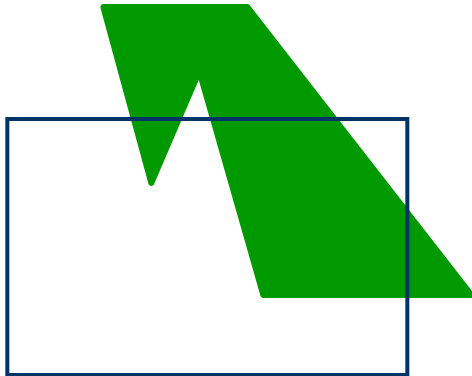


# Algoritmo Cyrus-Beck - Exercícios

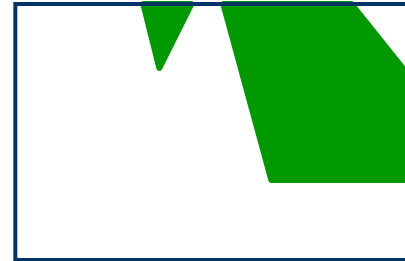
1. Entenda o programa cyrus-beck disponível no site da disciplina.
2. Modifique a interface dada como tarefa no slide 28 para incluir a opções de recorte com o algoritmo Cyrus-beck quando a região recortante for um polígono de  $n$  arestas. Ofereça as duas alternativas de recorte caso a região for um retângulo.

# Recorte de Polígonos

Antes do recorte



Depois do recorte



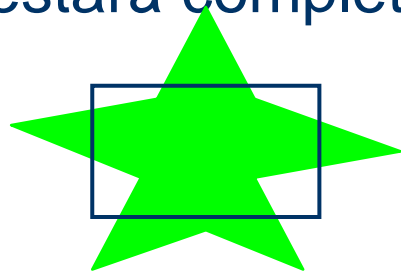
# Algoritmo de Sutherland-Hodgeman

- Princípio Básico:

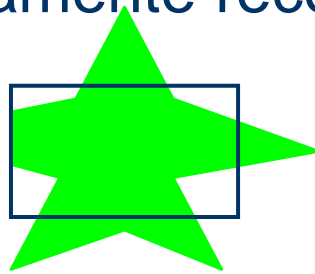
- Considerar individualmente cada aresta da região recortante.

- Recortar o polígono pela equação da aresta.

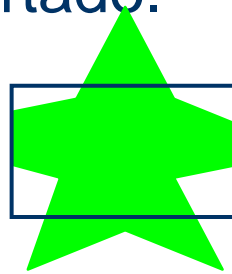
- Depois de fazer isso para todas as arestas, o polígono estará completamente recortado.



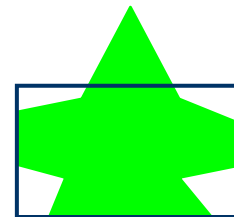
Polígono Original



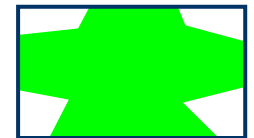
Recorte esquerdo



Recorte direito



Recorte inferior



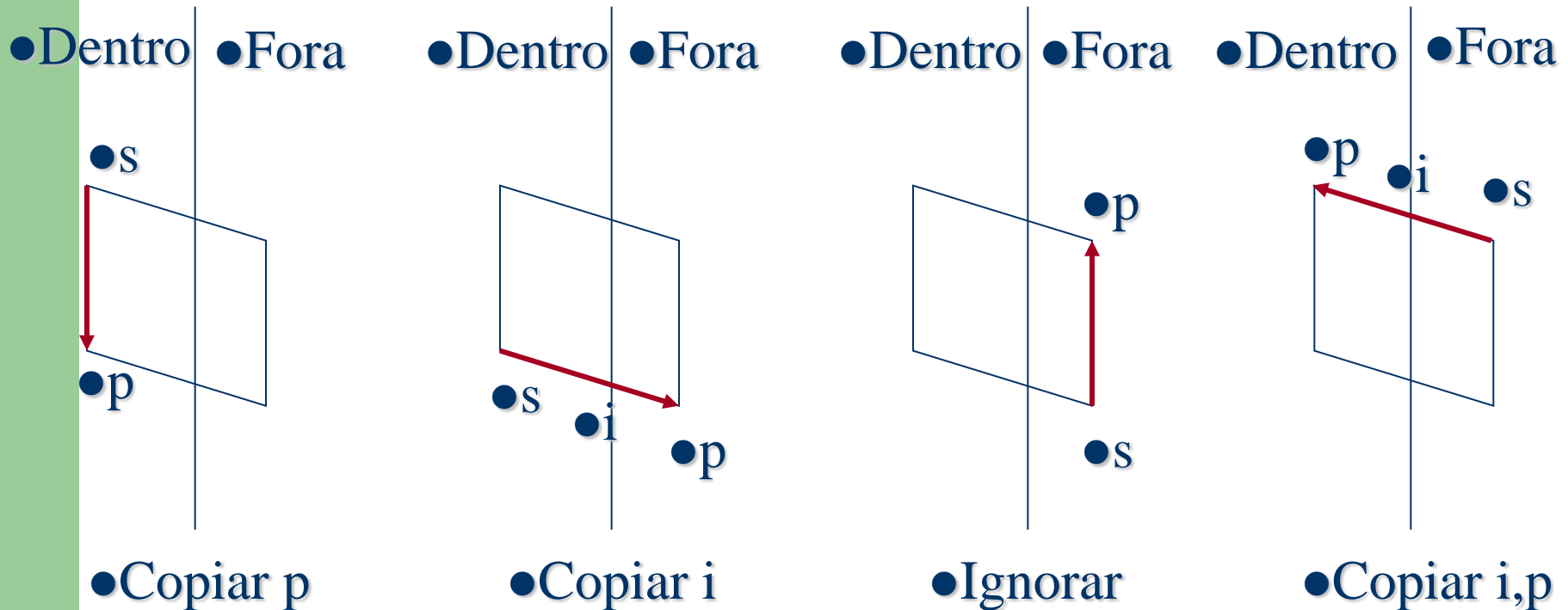
Recorte Superior

# Algoritmo de Sutherland-Hodgeman

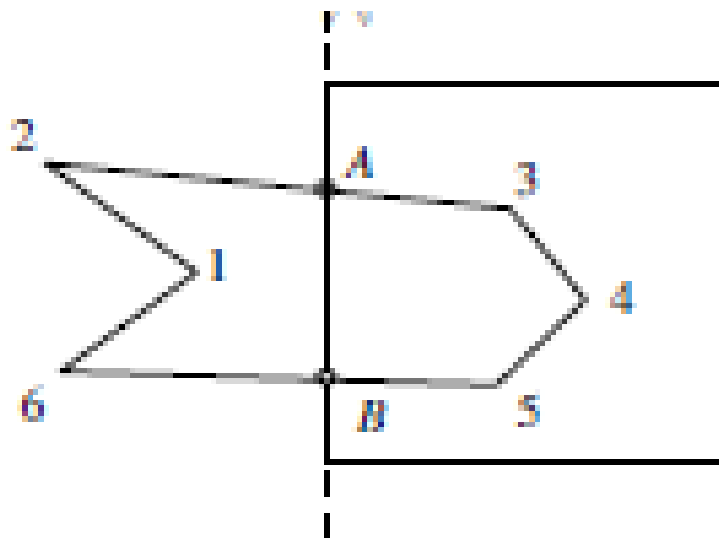
- Entrada/Saída do algoritmo
  - Entrada:
    - Região de recorte retangular
    - lista ordenada de vértices do polígono
  - Saída: lista dos vértices recortados, com alguns vértices originais (possivelmente) e outros novos (possivelmente)

# Algoritmo de Sutherland-Hodgeman

- Aresta de s a p se enquadra em um dos 4 casos:



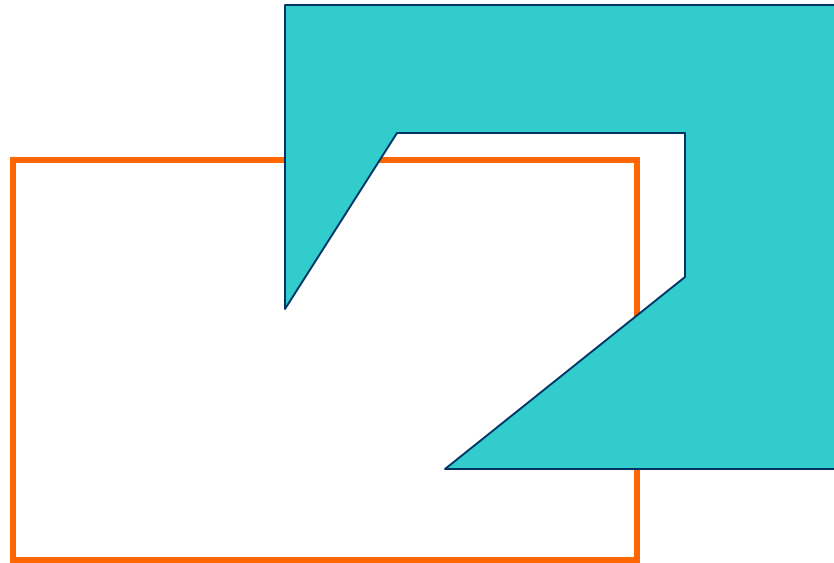
# Algoritmo de Sutherland-Hodgeman



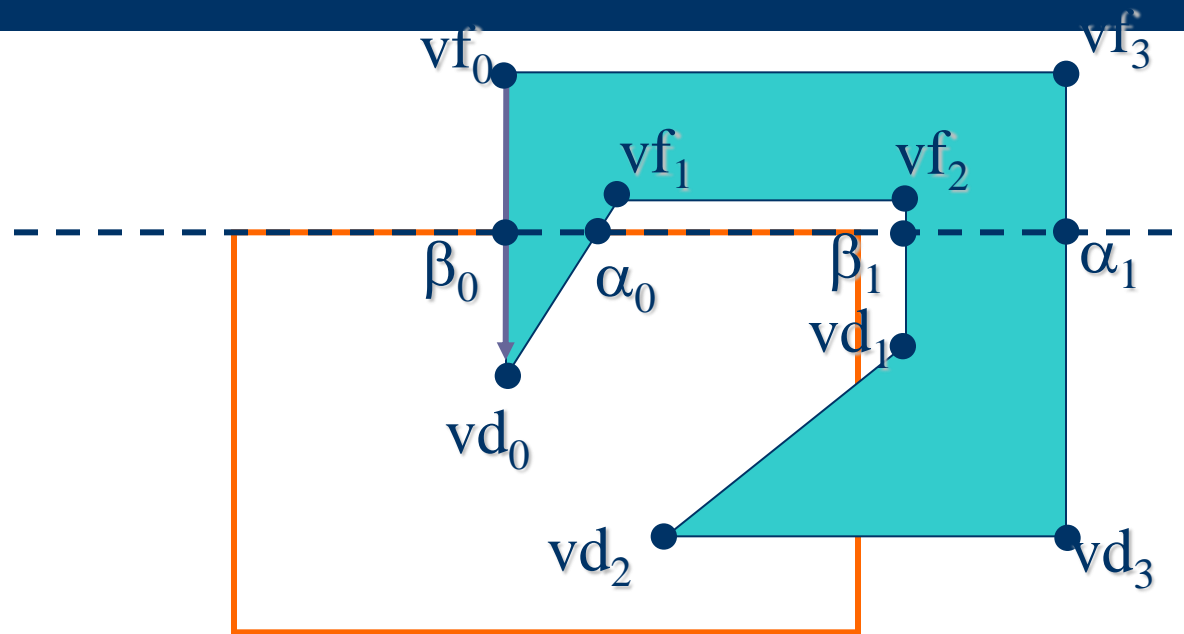
S	P	Ação
1	2	x
2	3	store A,3
3	4	store 4
4	5	store 5
5	6	store B
6	1	x



# Algoritmo de Sutherland-Hodgeman



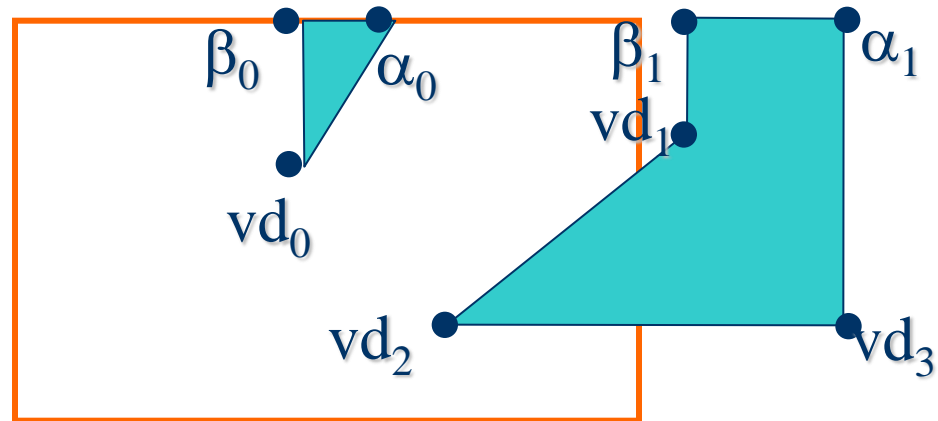
# Algoritmo de Sutherland-Hodgeman



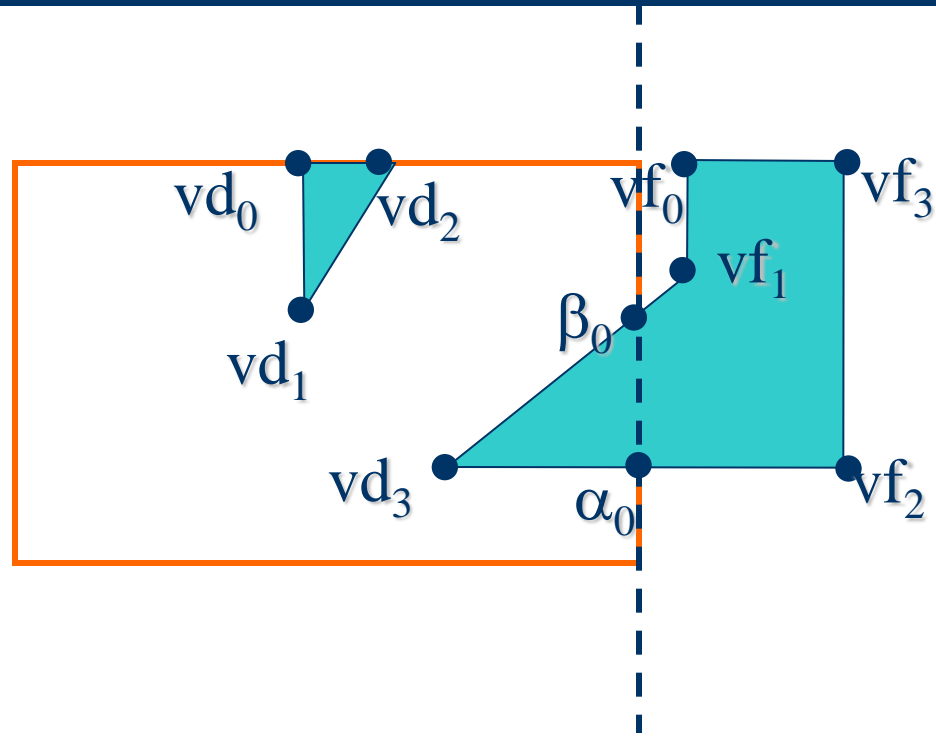
$vf_0 \beta_0 vd_0 \alpha_0 vf_1 vf_2 \beta_1 vd_1 vd_2 vd_3 \alpha_1 vf_3 vf_0$

$\beta_0 vd_0 \alpha_0 \beta_0 \quad e \quad \beta_1 vd_1 vd_2 vd_3 \alpha_1 \beta_1$

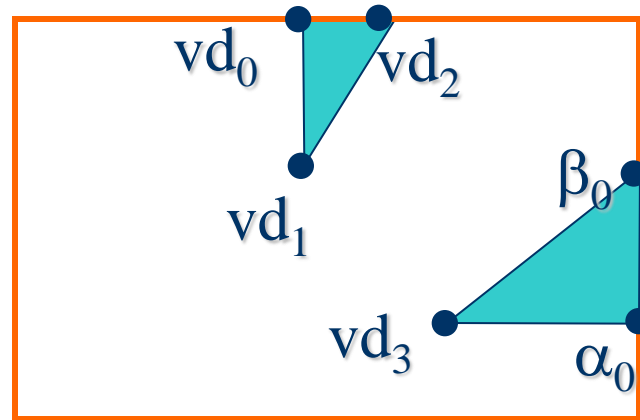
# Algoritmo de Sutherland-Hodgeman



# Algoritmo de Sutherland-Hodgeman

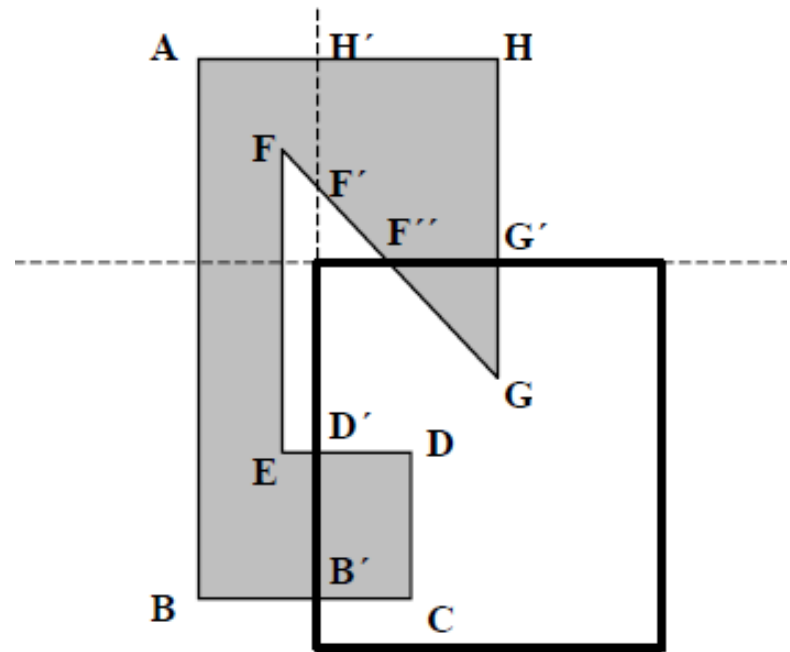


# Algoritmo de Sutherland-Hodgeman



# Algoritmo de Sutherland-Hodgeman

- Exercício: (1) Mostre passo a passo como seria a saída do recorte do seguinte polígono.



- (2) Você poderia rodar o exemplo acima usando o programa SutherlandHodgman.cpp postado no site? Quais modificações deverão ser feitas?

# Algoritmo de Sutherland-Hodgeman

- (3) Estenda o algoritmo Sutherland-Hodgeman para que considere o recorte de qualquer polígono (inclusive polígonos côncavos).
- (4) Estenda a interface dada como tarefa no slide 35 incluindo a opção de recorte de polígonos no menu já existente. Considere a captura interativa dos pontos da janela recortante, assim como a sequência ordenada dos vértices do polígono.