Computação Gráfica - Recorte

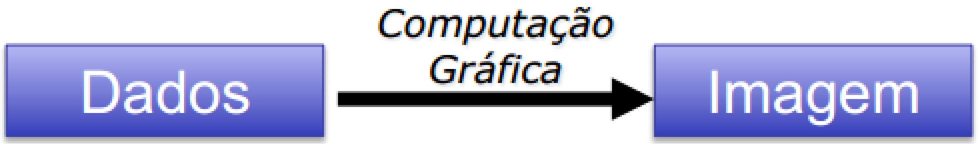
Profa. Mercedes Gonzales Márquez

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)

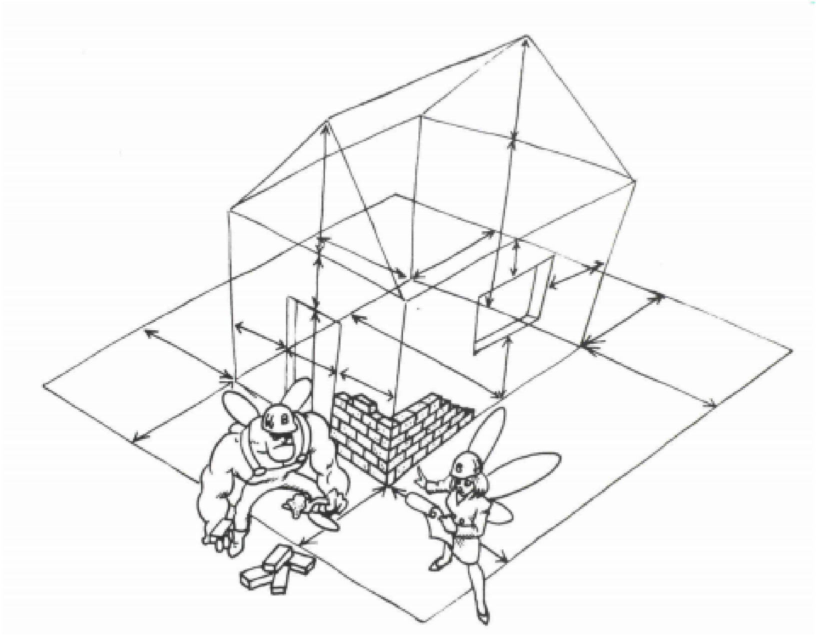
"Computação Gráfica é o conjunto de métodos e técnicas para transformar dados em imagem através de um dispositivo gráfico."

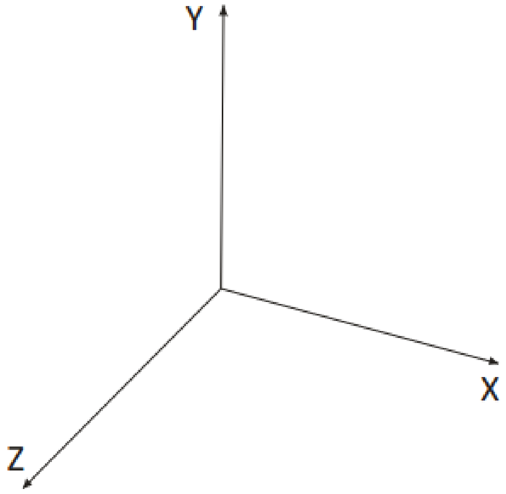


Recapitulando... (Modelagem Geométrica)

O que é Modelagem Geométrica?

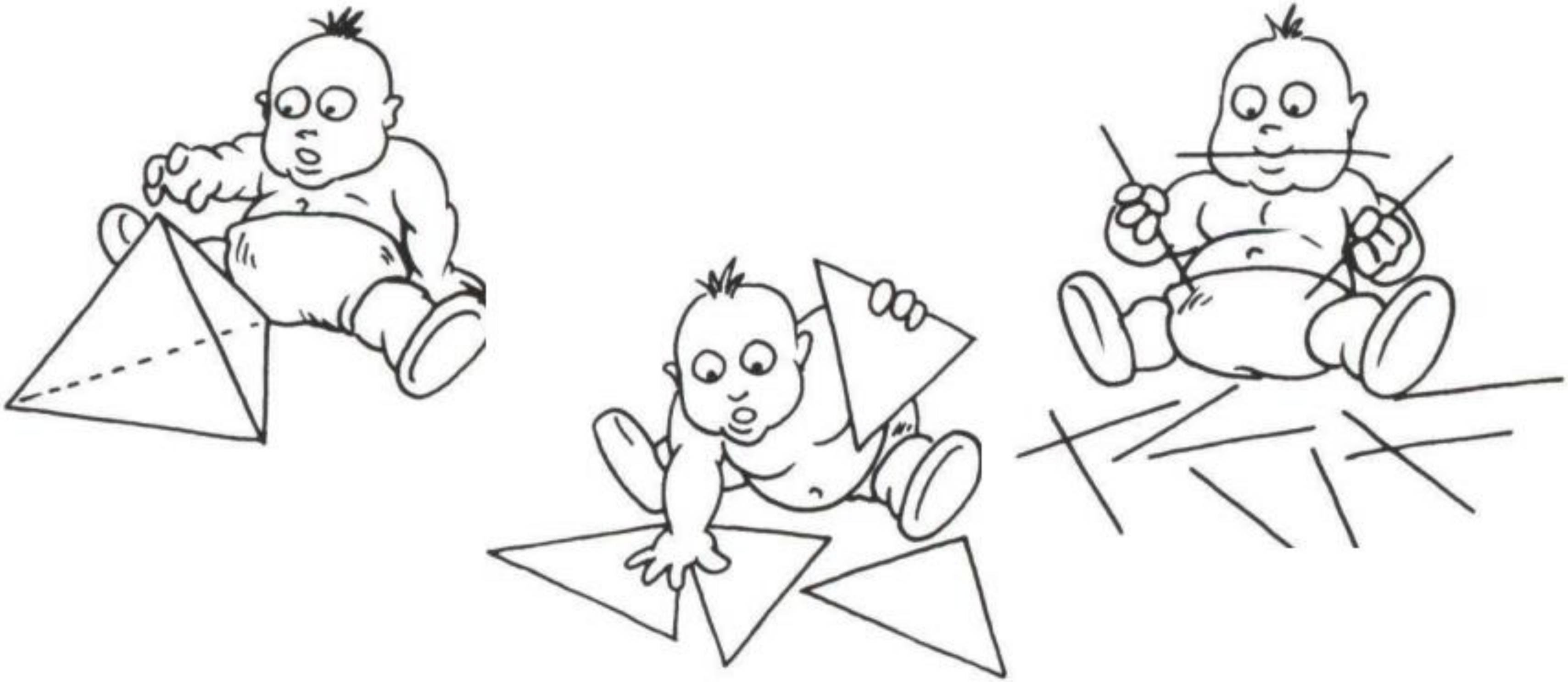
Estruturar e descrever dados geométricos no computador



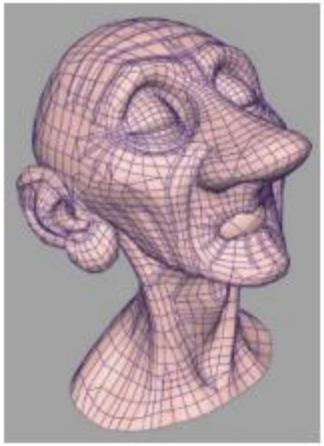


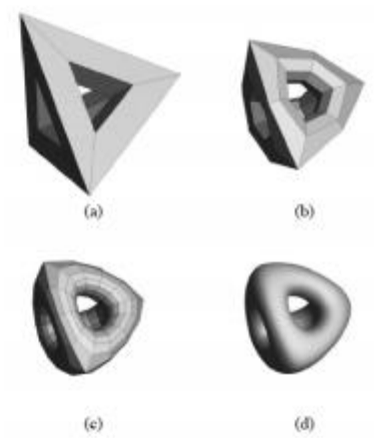
Recapitulando... (Modelagem Geométrica)

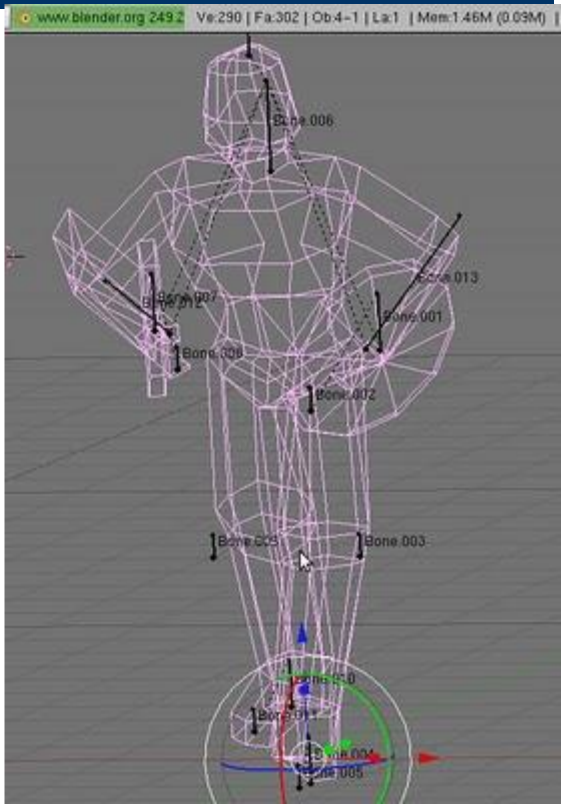
Objetos são definidos por pontos, linhas e planos



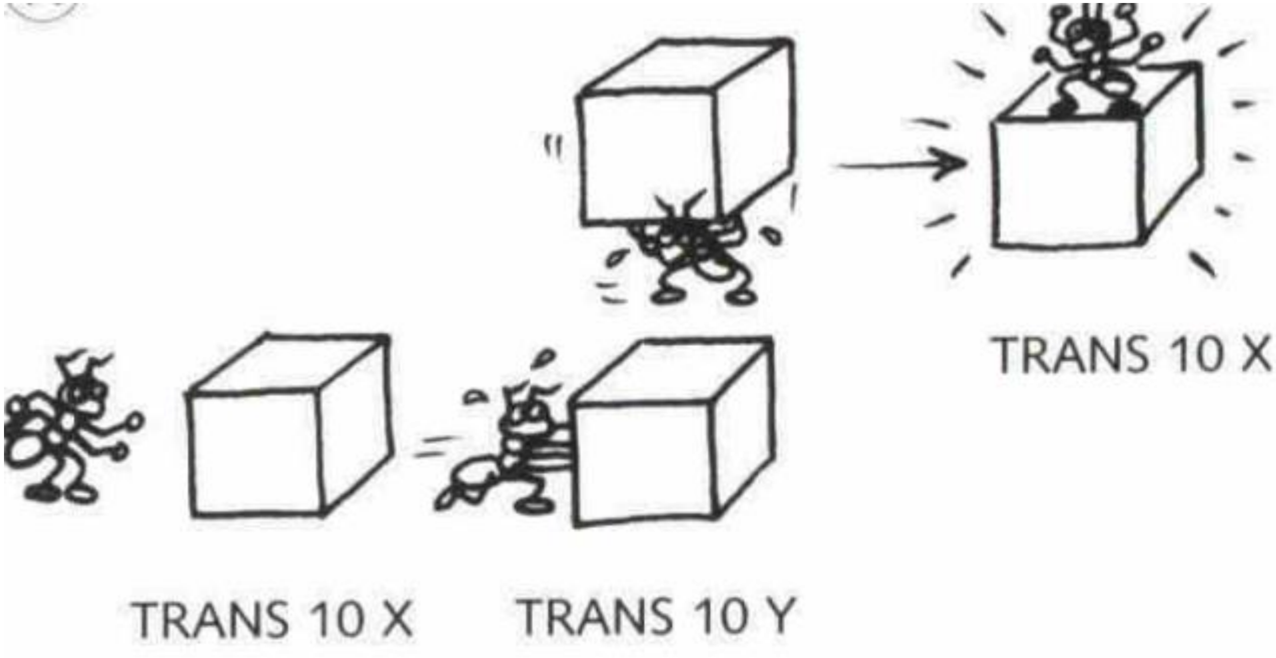
Recapitulando... (Modelagem Geométrica)



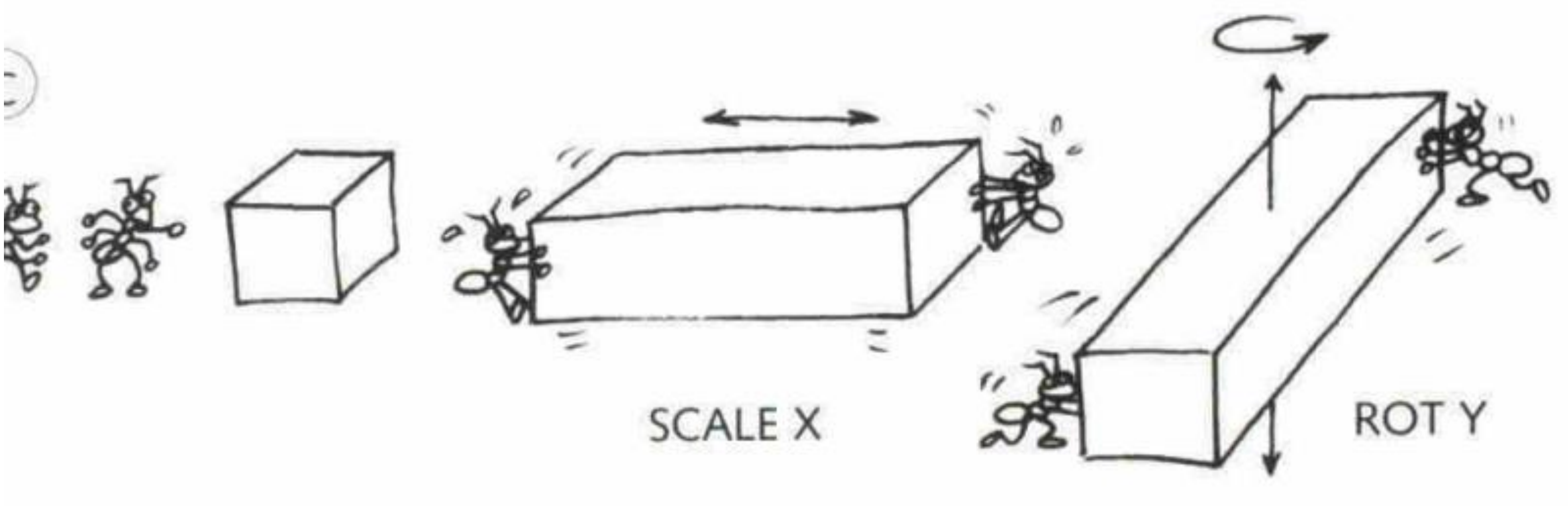




Recapitulando ... (Transformações Geométricas)



Recapitulando ... (Transformações Geométricas)



Recapitulando ...

MODELAGEM GEOMETRICA 3D

+

TRANSFORMAÇÕES GEOMÉTRICAS

=

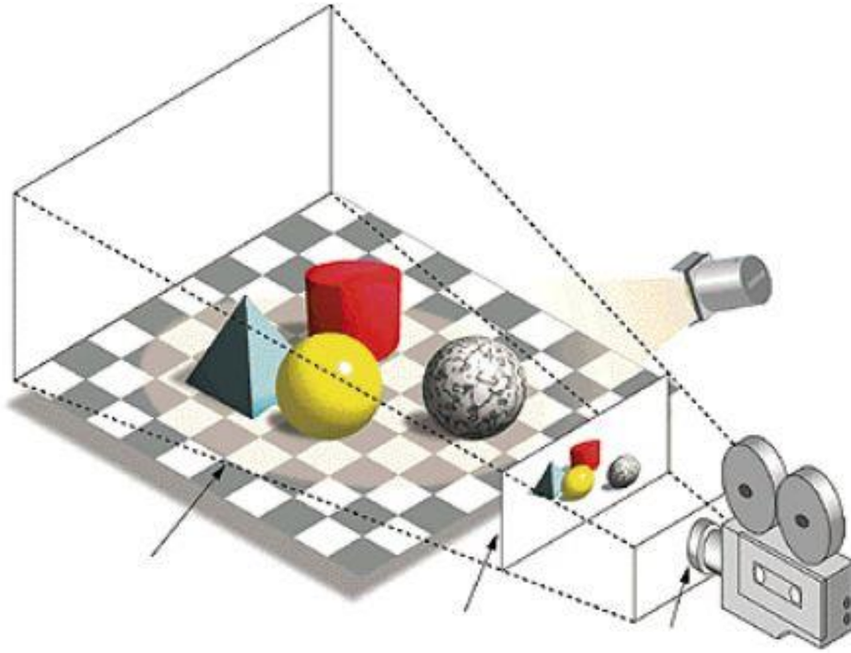
CENÁRIO 3D

Recapitulando ...

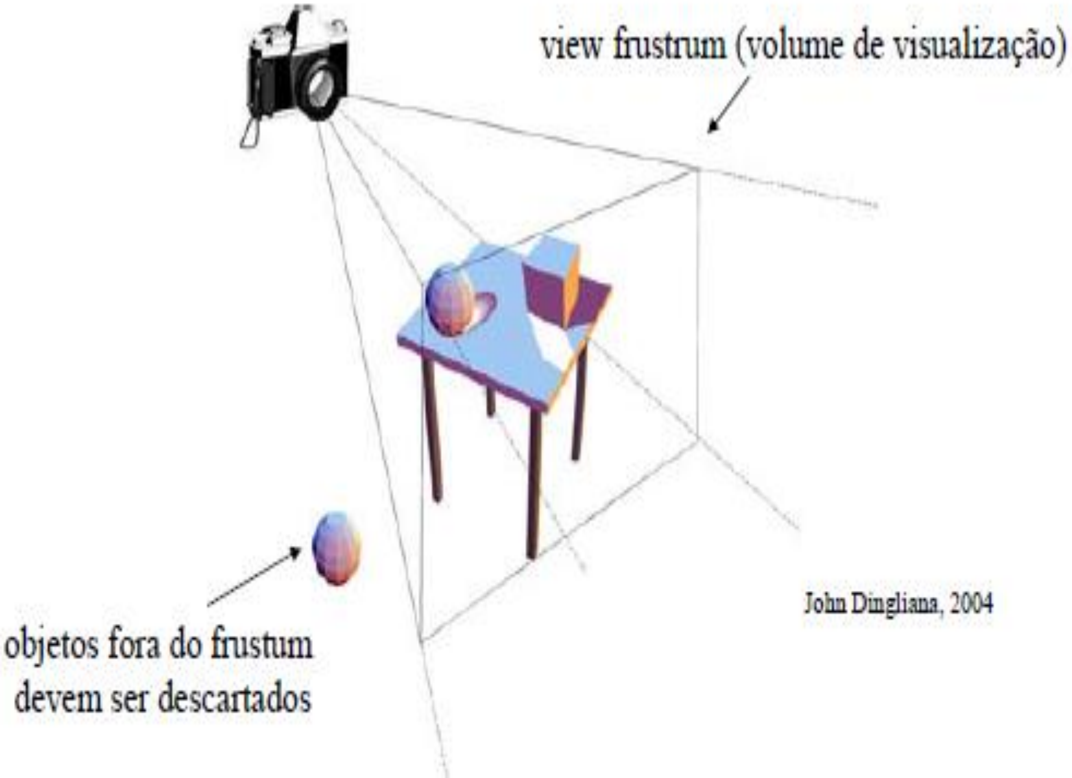
CENÁRIO 3D → IMAGEM

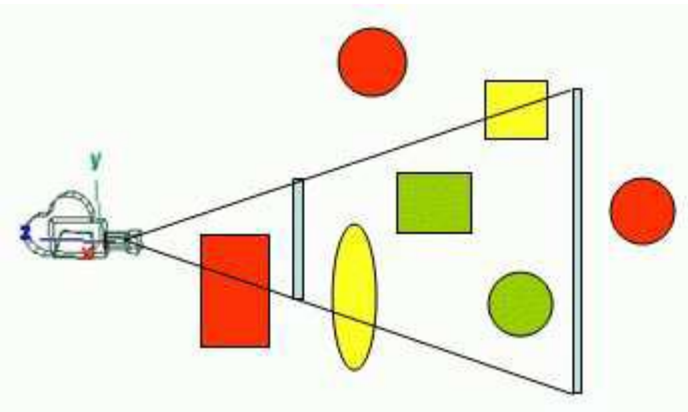
SERÁ NECESSÁRIO:

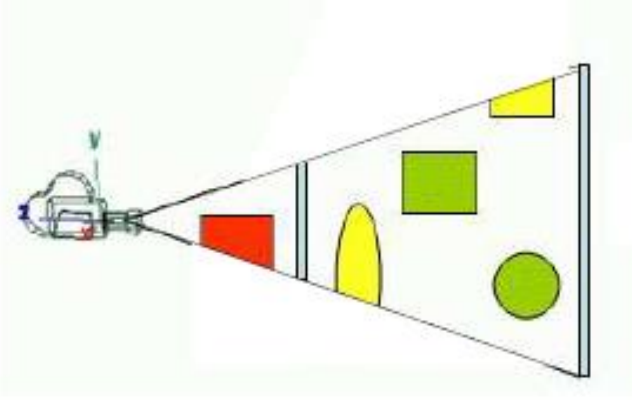
- RECORTE
- PROJEÇÃO
- AMOSTRAGEM
- REMOÇÃO DE SUPERFÍCIES ESCONDIDAS (VISUALIZAÇÃO)
- COLORIZAÇÃO (ILUMINAÇÃO E TEXTURIZAÇÃO)



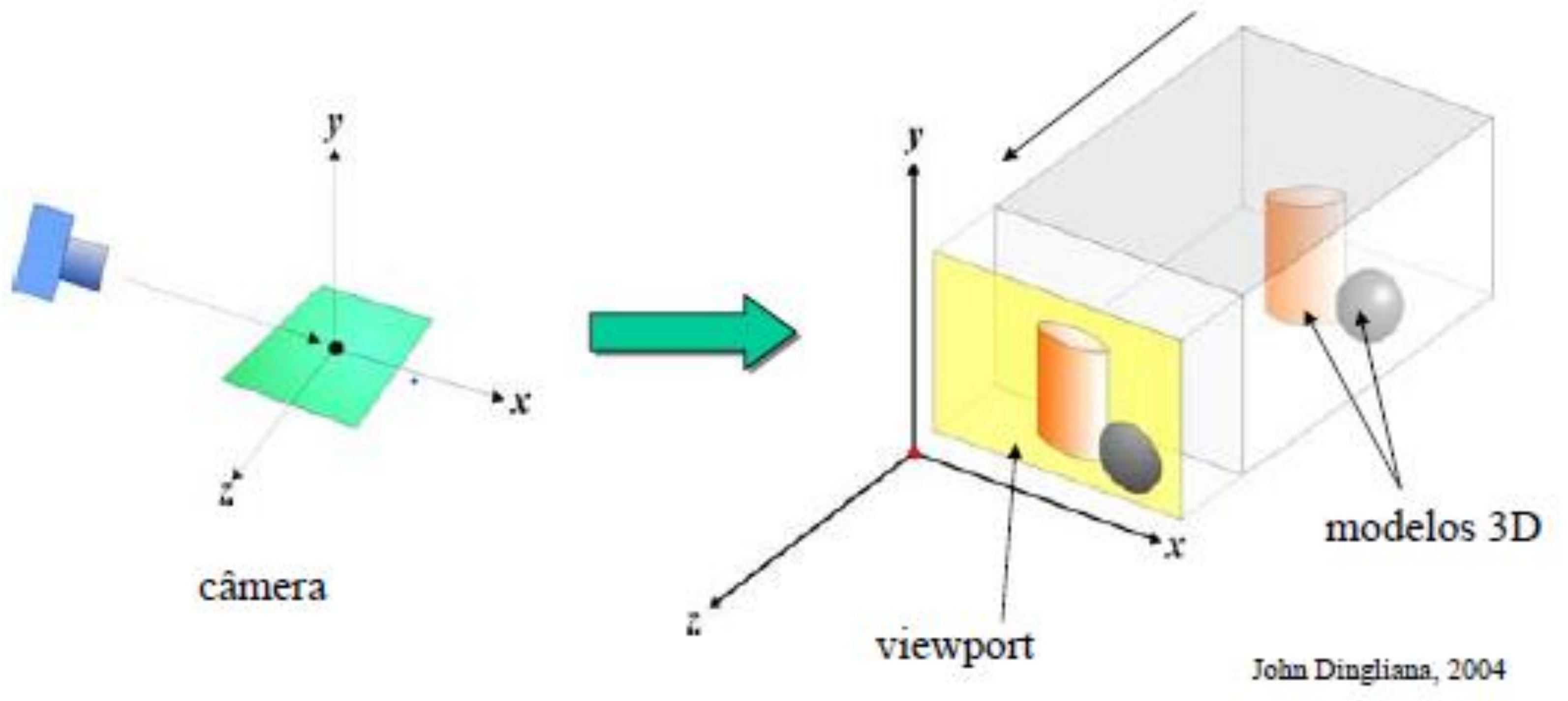
Pipeline gráfico 3D – Recorte



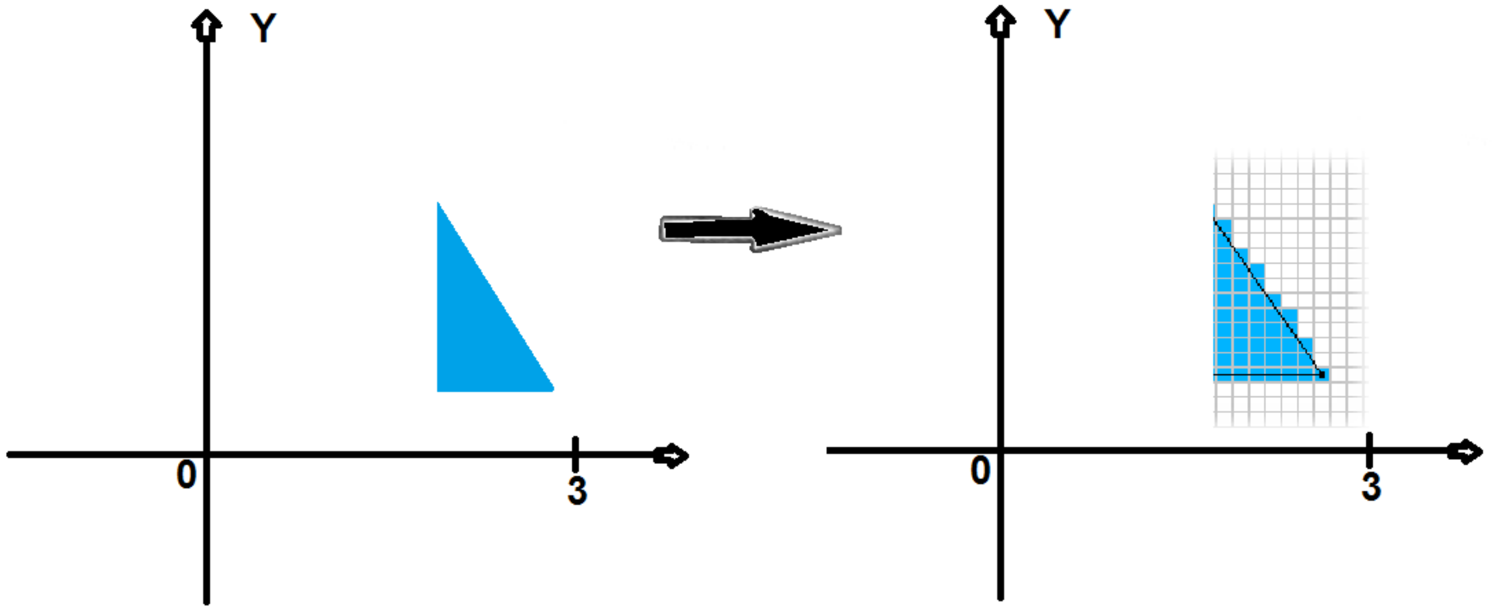




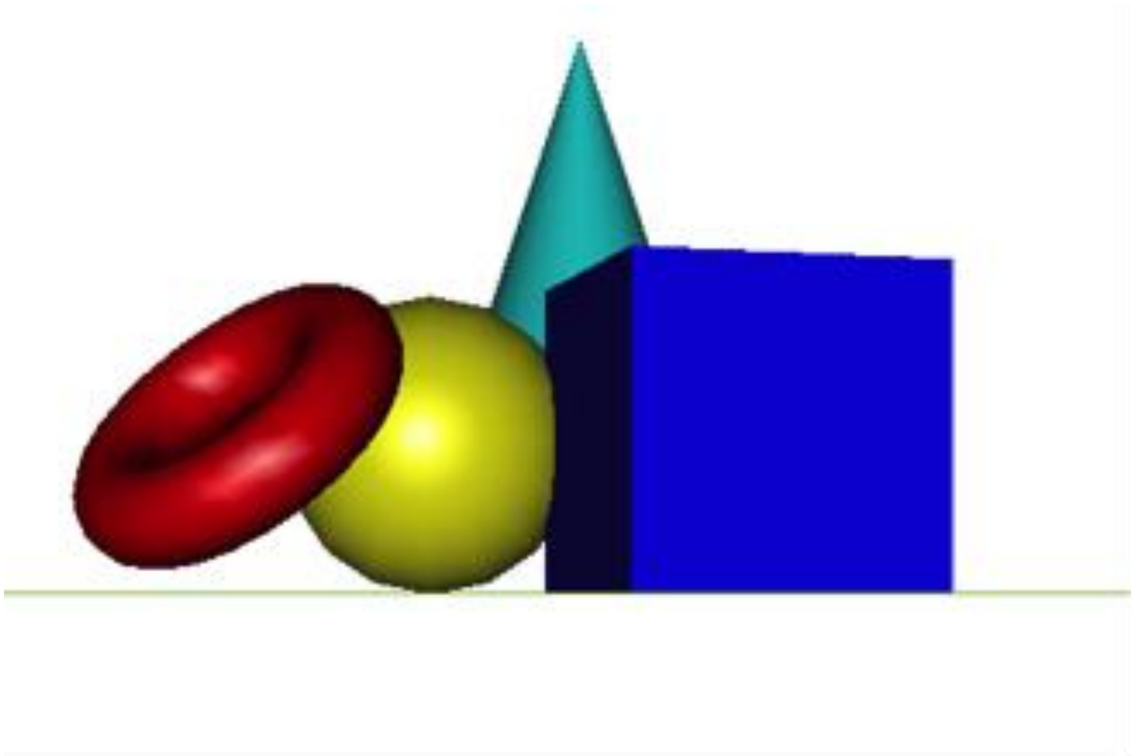
Pipeline gráfico 3D – Projeção e Mapeamento



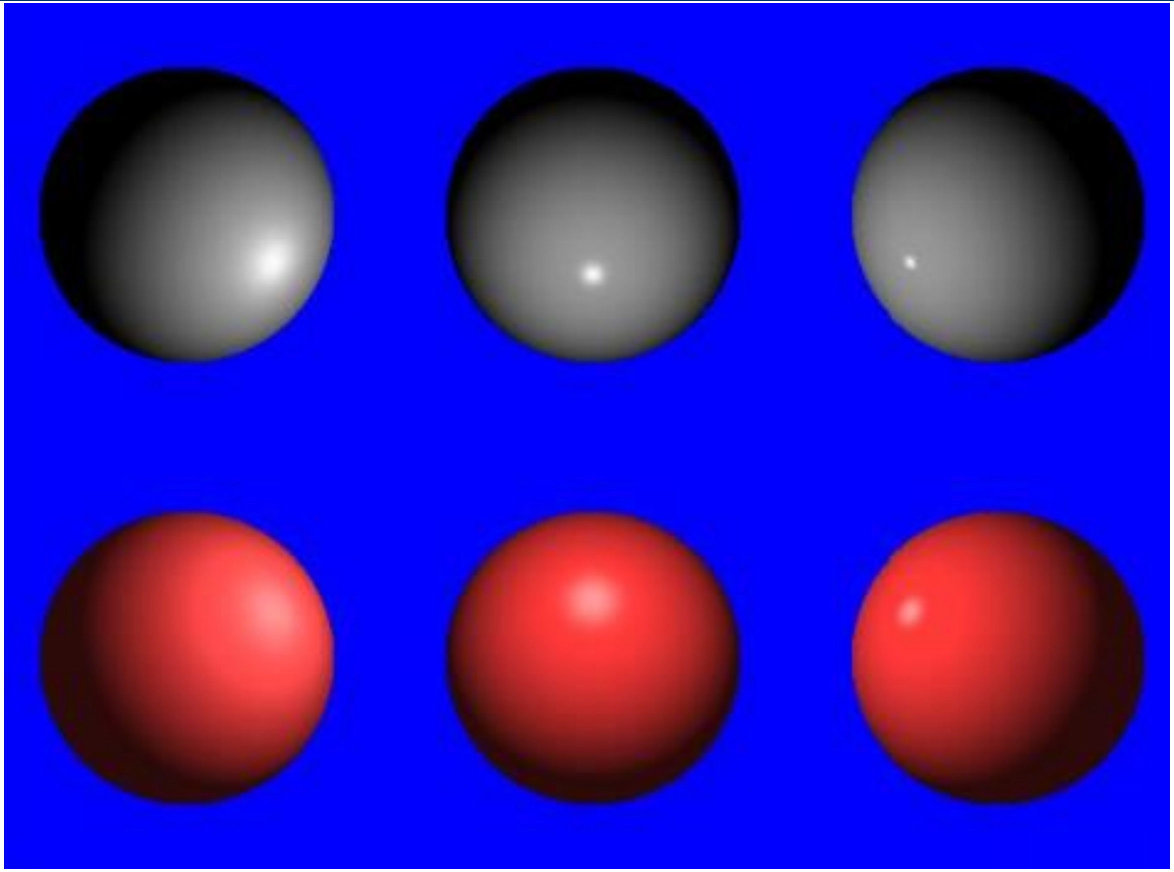
Pipeline gráfico 3D – Conversão Vetorial-Matricia



Pipeline gráfico 3D – Remoção de Superfícies Escondidas



Pipeline gráfico 3D – Iluminação



Pipeline gráfico 3D – Textura

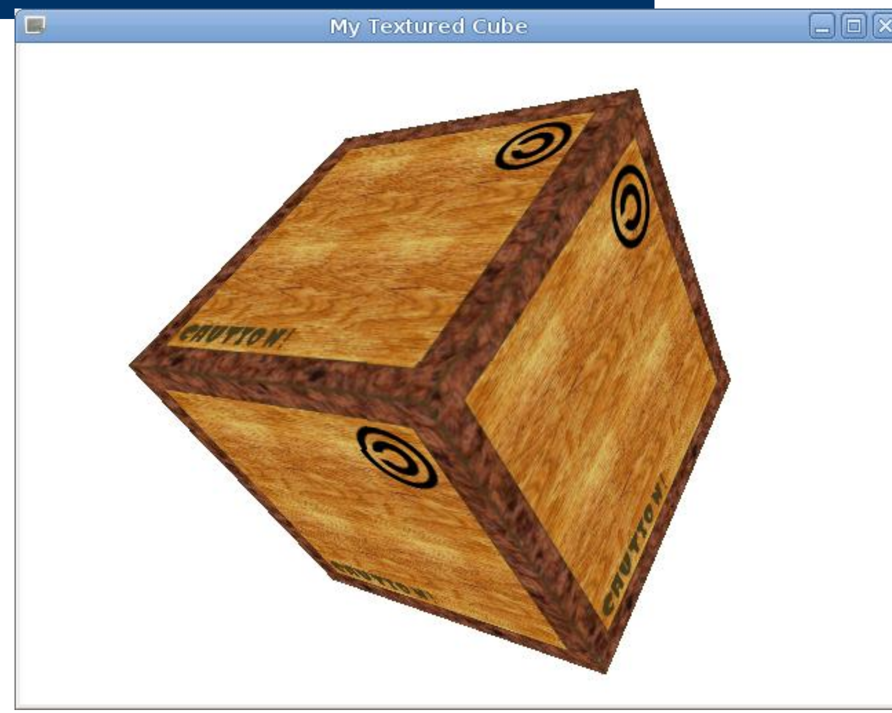
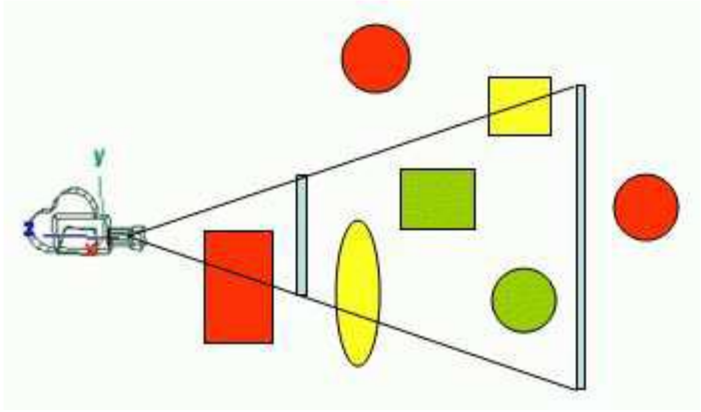


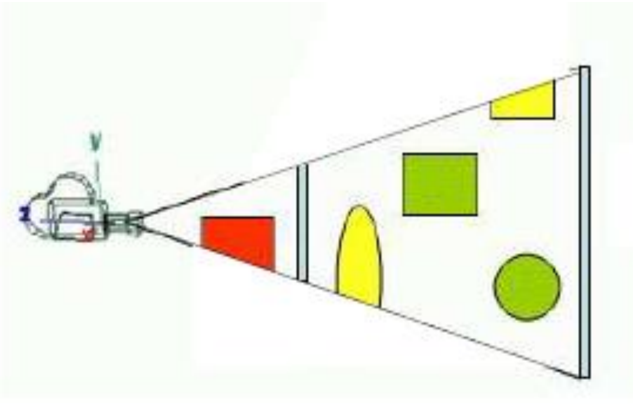
Imagem de textura

Cubo texturizado

Recorte

- A técnica de recorte consiste na remoção das partes que não estejam dentro do volume de visão.
- Somente as formas geométricas contidas no volume de visão devem aparecer (veja um corte do volume de visão)





Recorte

- Remover os pontos que estão fora do volume de visão se reduz, numericamente, a 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) ou (x,y,z)
- Região Recortante:
  - Região Retangular:
    (xmin,xmax,ymin,ymax)
  - Volume Paralelipedal:
    (xmin,xmax,ymin,ymax,zmin,zmax)

Recorte de Pontos

- Objeto: (x,y)
- Região Recortante:
  - Região Retangular: (xmin,xmax,ymin,ymax)
- Um ponto (*x, y), estará dentro do* retângulo de visualizacao, se:

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

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

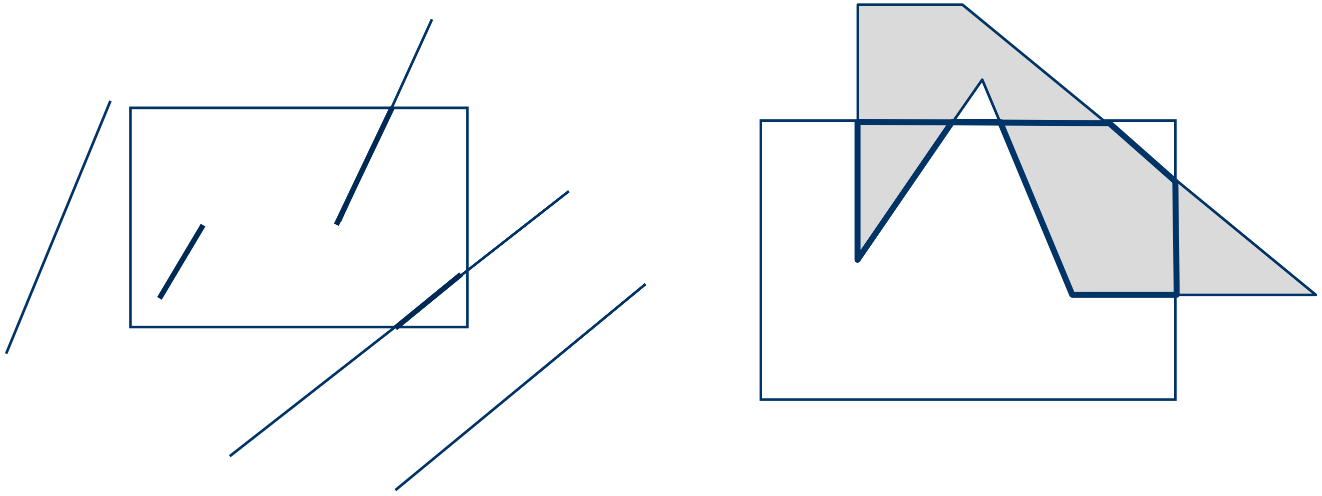
Recorte de Segmentos em Regiões Planares

- Algoritmos de Recorte de Segmentos em Regiões Planares ou Recorte 2D
  - Algoritmo de Cohen Sutherland
  - Algoritmo de Cyrus-Beck
- Objeto: segmentos de reta
- Região Recortante R:
  - Região Retangular:

    (xmin,xmax,ymin,ymax) para o algoritmo Cohen Sutherland
  - Região Convexa

    N pontos (para o algoritmo Cyrus Beck)

Recorte de Segmentos em Regiões Planares ou Recorte 2D

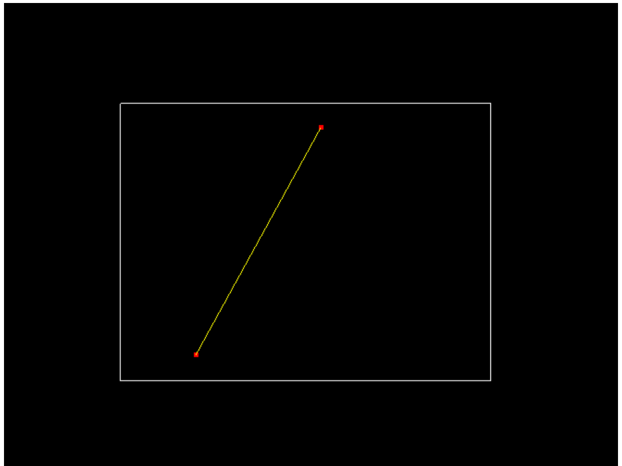


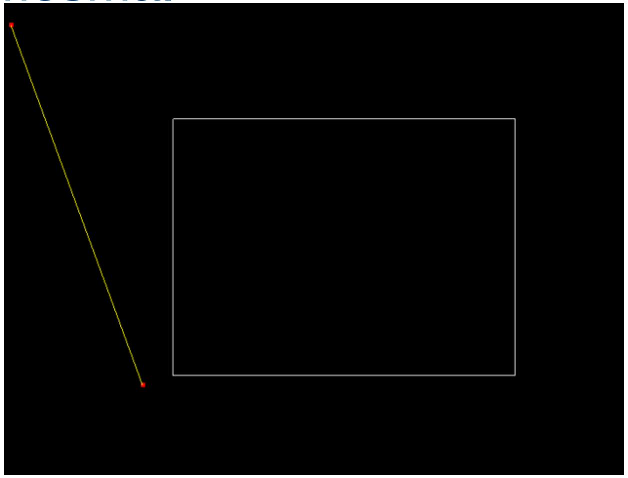
Algoritmo Cohen-Sutherland

- Objeto: segmento de reta
- Região Recortante:
  - Região Retangular definida por (xmin, xmax, ymin, ymax)

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 no sub-espaço (semi-plano) definido a partir de uma aresta da região e para fora da mesma.





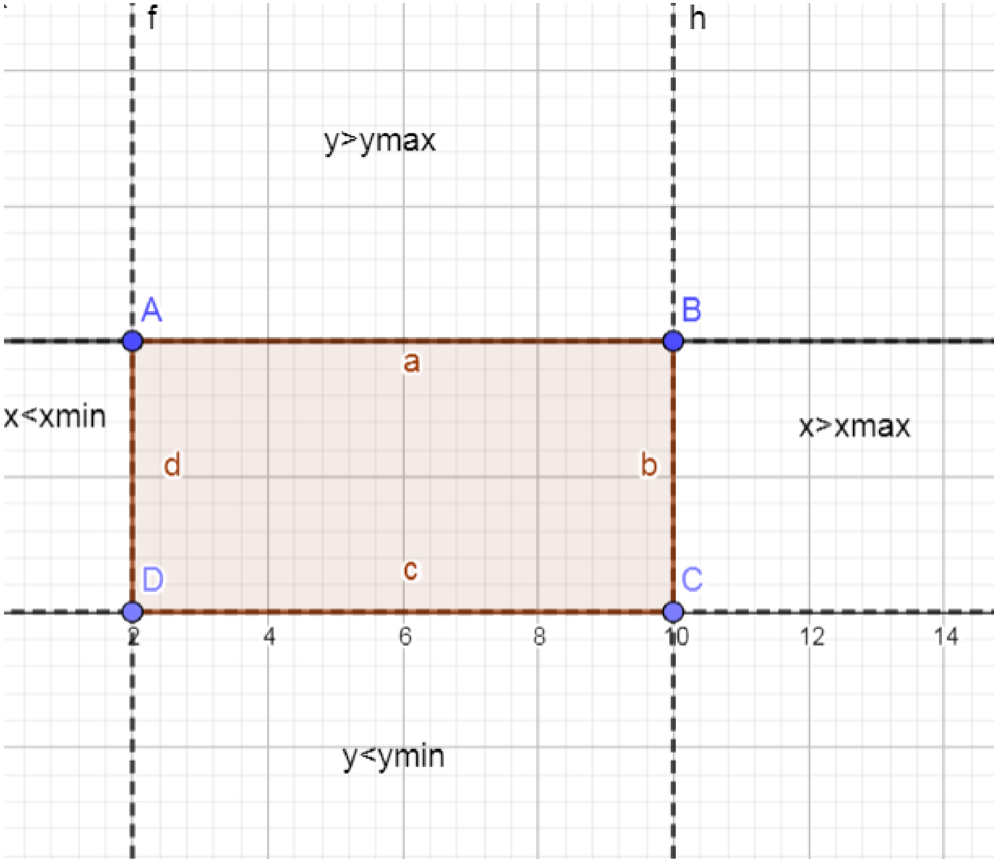
Algoritmo de Cohen-Sutherland

- Temos 4 sub-espaços, um a partir de cada aresta da região recortante R.

- Subespaço superior: $y>ymax$
- Subespaço inferior: $y<ymin$
- Subespaço direito: $x>xmax$
- Subespaço esquerdo: $x<xmin$

Algoritmo de Cohen-Sutherland

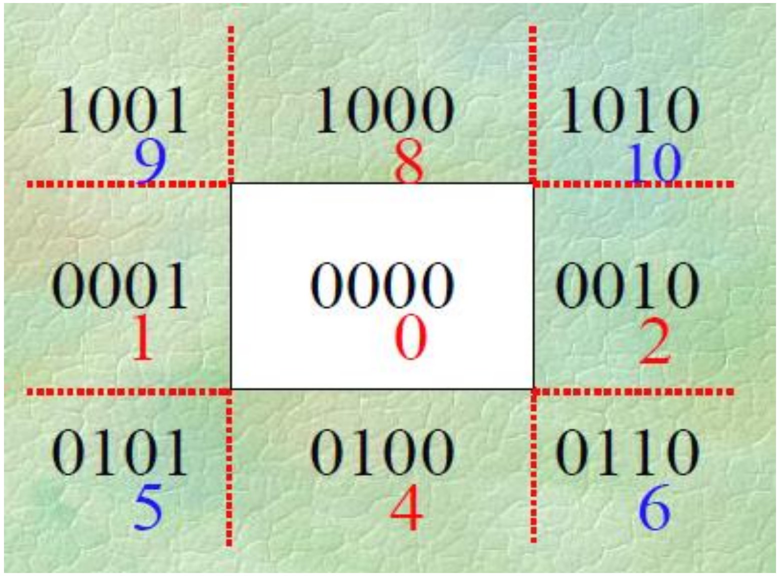
- Exemplo: xmin=2, xmax=10ymin=0, ymax=4
- Tenho 4 semiplanos fora da região, definidos a partir das suas arestas.
- x<xmin
- x>xmax
- y<ymin
- y>ymax



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 1, se o vértice estiver dentro do subespaço e 0 se estiver fora.

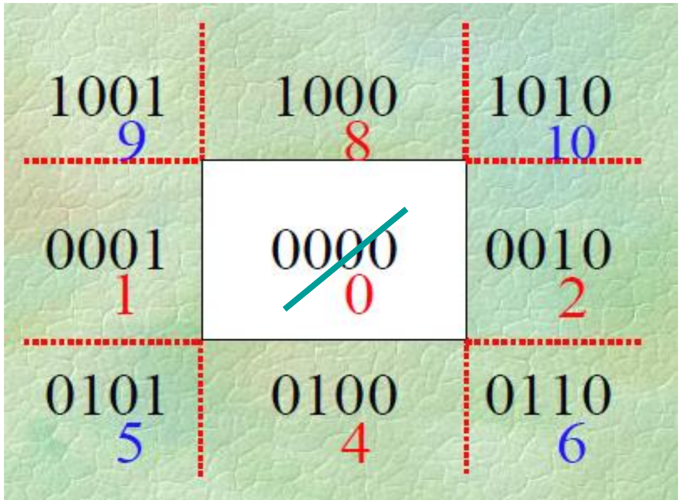
  - Subespaço superior: y>ymax (1o bit)
  - Subespaço inferior: y<ymin (2o bit).
  - Subespaço direito: x>xmax (3º bit)
  - Subespaço esquerdo: x<xmin (4º bit)



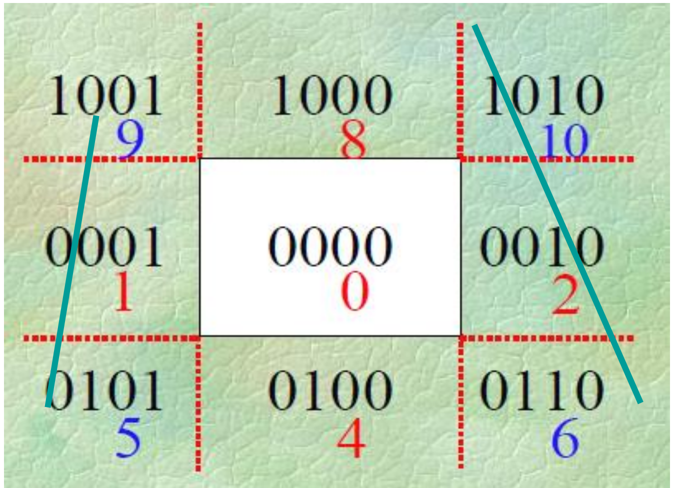
Algoritmo de Cohen-Sutherland

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

1. O código dos vértices é 0000: o segmento está contido na região.

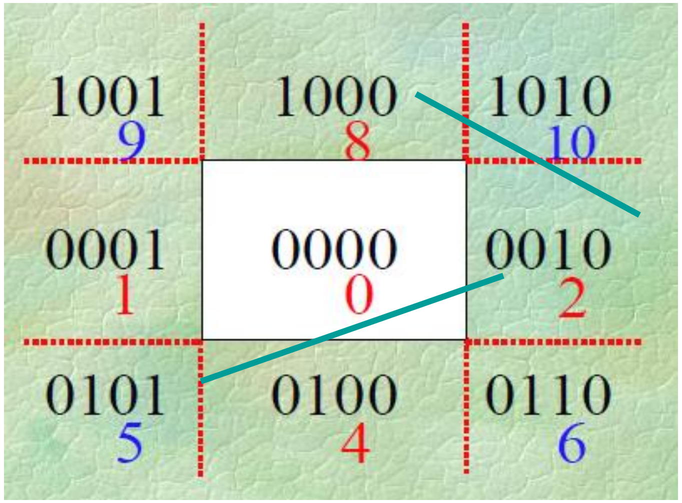


2. O resultado é diferente de zero: o segmento está totalmente fora da região.



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, substituimos os valores xmin, xmax, ymin e ymax em uma das seguintes equações da reta a fim de obter as interseções sucessivamente.

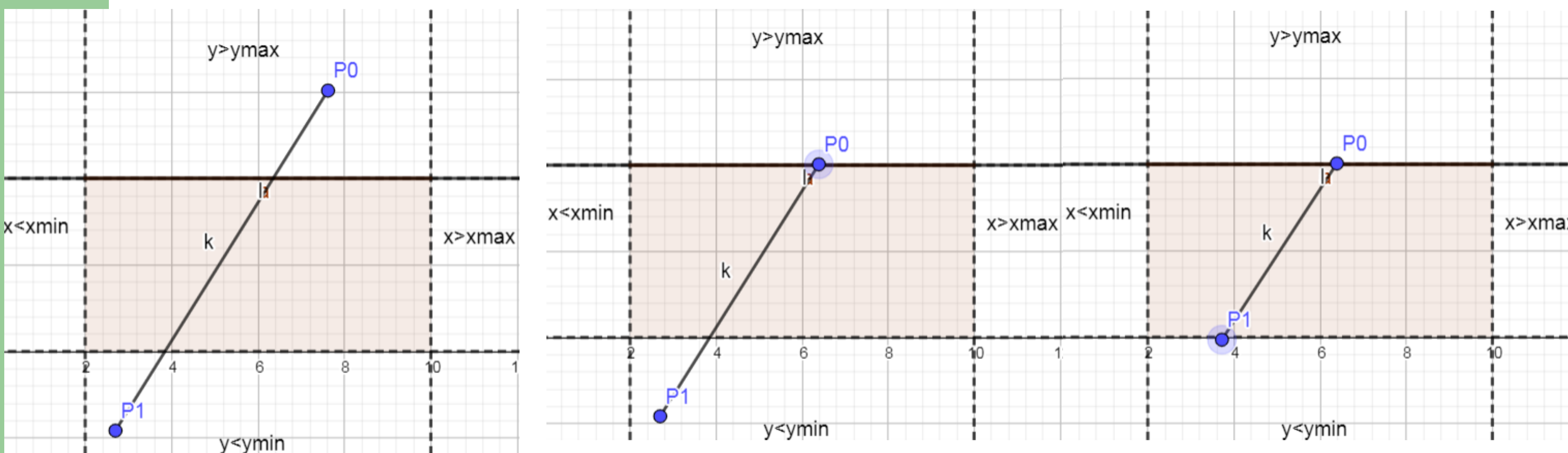
Usamos a equação da reta dada por dois pontos P0 e P1

y em função de x : $y-y0=((y1-y0)/(x1-x0))(x-x0)$

ou

x em função de y : $x-x0=(x1-x0)/(y1-y0)(y-y0)$

Algoritmo de Cohen-Sutherland



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;    //1000
  if (y < ymin) code += 4;    //0100
  if (x > xmax) code += 2;    //0010
  if (x < xmin) code += 1;    //0001

  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

Usamos a equação da reta dada por dois pontos

y-y0=((y1-y0)/(x1-x0))(x-x0)

```
    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);
}
```

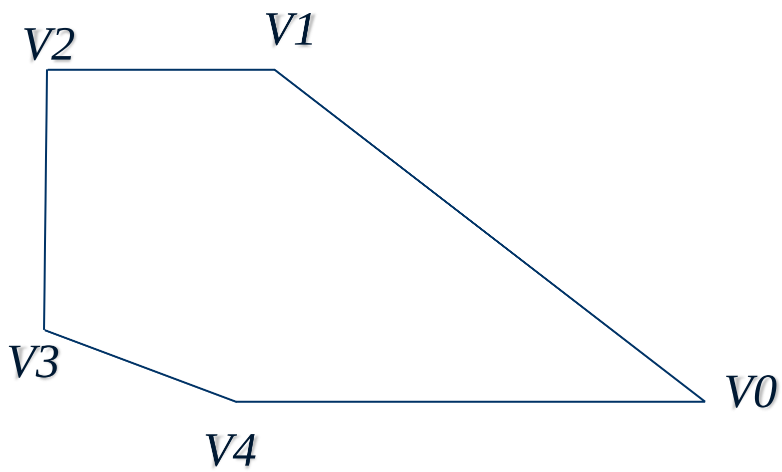
Algoritmo de Cohen-Sutherland

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: segmento de reta
- Região Recortante:
  - Região Convexa de n pontos

    No exemplo n=5



Algoritmo de Cyrus-Beck

- Princípios Básicos:

  - Uso do vetor normal das arestas da região de recorte para determinar de forma simples as interseções com o segmento.

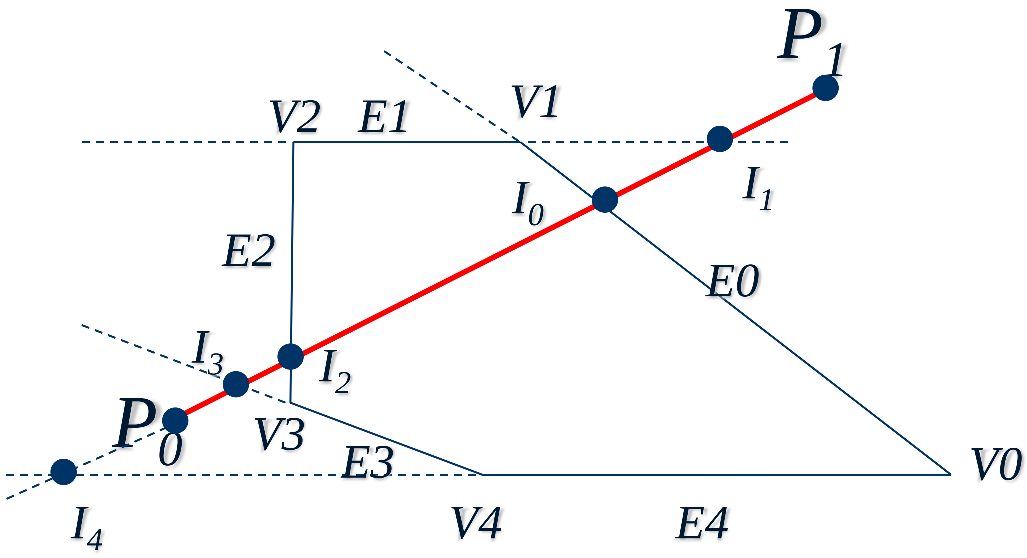
  - Uso do mesmo vetor normal para determinar se o segmento está entrando ou saindo da região de recorte.

Algoritmo Cyrus-Beck

1. O segmento de reta é representado pela sua equação paramétrica P(t)=P0+t(P1-P0), 0<=t<=1, e são determinadas as interseções deste segmento com cada uma das n arestas da região de recorte. Para isso as retas suporte do segmento ou das arestas poderão ser necessárias.

Algoritmo Cyrus-Beck

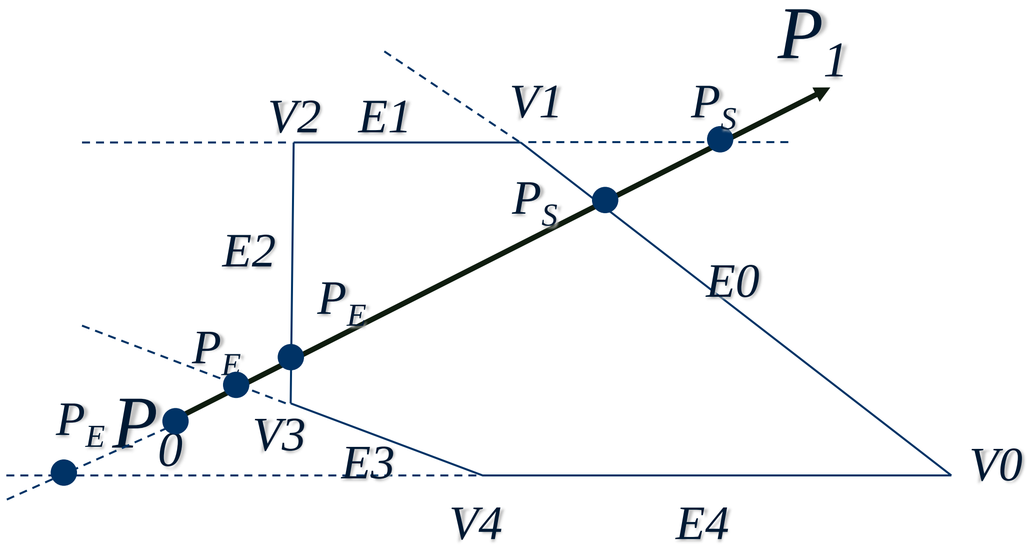
1º Passo



Algoritmo Cyrus-Beck

2º Passo: Classificar as interseções: Considerando o P0P1 como segmento orientado, classifique as interseções como PE (Potencialmente entrando) ou PS (Potencialmente saindo)

Algoritmo Cyrus-Beck

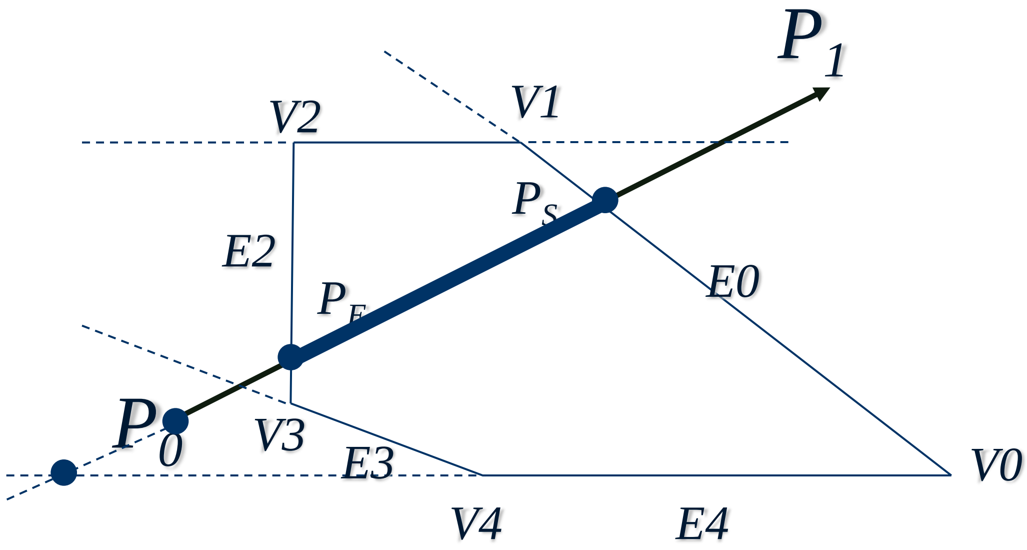


Algoritmo Cyrus-Beck

3. Fazer o recorte, se existir: Para isso determine o PE com maior t (PEM) e o PS com menor t (PSm)

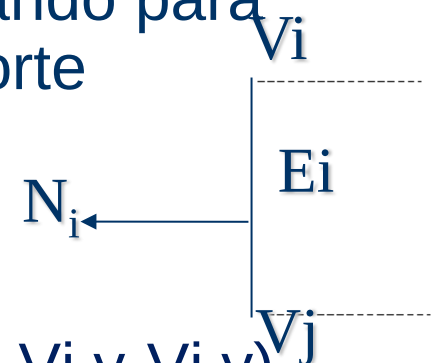
- Se PEM<PSm recorte nesses 2 pontos

Algoritmo Cyrus-Beck



Algoritmo Cyrus-Beck – Vetor normal

- Determinamos o vetor normal Ni (apontando para fora) da aresta Ei=ViVj da região de recorte



- Calculamos o vetor ViVj = (Vj.x-Vi.x, Vj.y-Vi.y)
- Para determinar o vetor ortogonal a ViVj, deve-se cumprir a seguinte igualdade: <N,ViVj>=0
- Assim <(n.x,n.y),(Vj.x-Vi.x,Vj.y-Vi.y)>=0
  n.x(Vj.x-Vi.x)+n.y(Vj.y-Vi.y)=0

E fazendo n.y=1 temos que n.x= -(Vj.y-Vi.y)/(Vj.x-Vi.x)

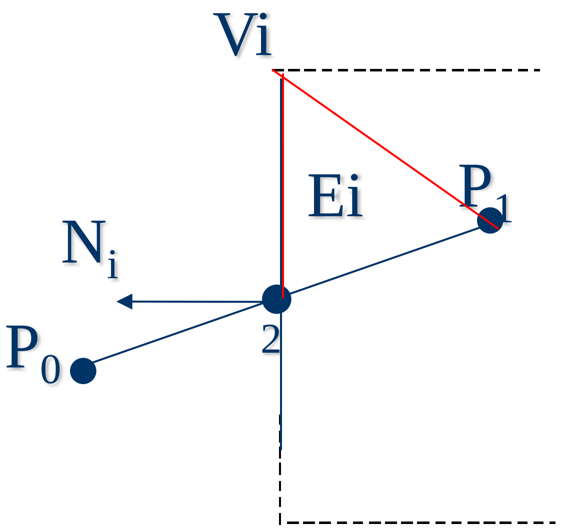
- Vetor normal então é: (-(Vj.y-Vi.y)/(Vj.x-Vi.x),1)

Determinando interseção entre o segmento e a aresta

- Seja P(t)=P0+t(P1-P0), 0<=t<=1, a equação do segmento S unindo P0 e P1.
- O ponto P(t) que ligado a Vi gerar um vetor ortogonal a Ni será o ponto que cumprir a seguinte condicao:

$$\boxed{\langle N_i, P(t) - V_i \rangle = 0}$$

- Esse ponto P(t) será justamente o ponto de interseção do segmento *P(t)* com a aresta Ei.



Determinando a interseção

- Determinação do t no qual haverá interseção.

  \- Condição para a interseção

$$\boxed{\langle N_i, P(t) - V_i \rangle = 0}$$

$$\boxed{\begin{aligned}
&\langle N_i, P_0 + t(P_1 - P_0) - V_i \rangle = 0 \\
&\langle N_i, P_0 - V_i \rangle + \langle N_i, t(P_1 - P_0) \rangle = 0 \\
&\langle N_i, t(P_1 - P_0) \rangle = -\langle N_i, P_0 - V_i \rangle \\
&t\langle N_i, (P_1 - P_0) \rangle = -\langle N_i, P_0 - V_i \rangle \\
&t = \frac{\langle N_i, P_0 - V_i \rangle}{-\langle N_i, P_1 - P_0 \rangle}
\end{aligned}}$$

Determinando a interseção

- Então haverá interseção do segmento com a aresta Ei

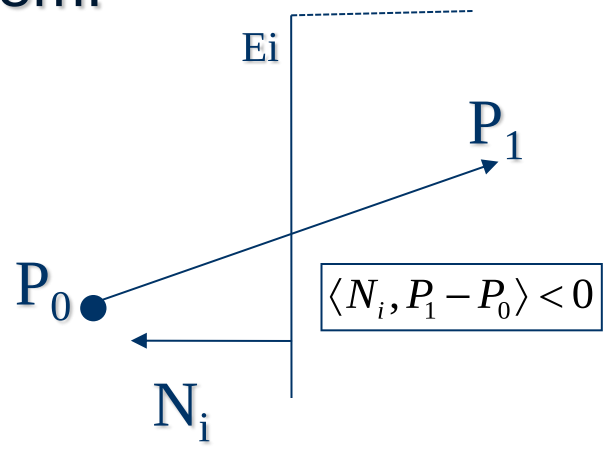
  - Se P0 não coincidir com P1. $\boxed{P_0 \neq P_1}$

  - Se o segmento não for paralelo à aresta $\boxed{\langle N_i, P1 - P0 \rangle \neq 0}$

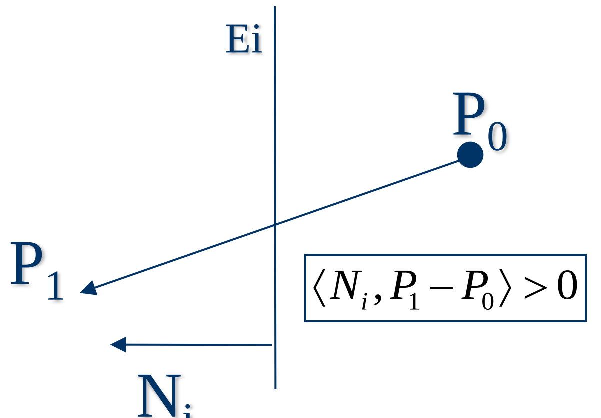
  - Se o ponto de interseção estiver dentro do segmento. $\boxed{0 \leq t \leq 1}$

Classificando a interseção

- Classificamos as interseções restantes em:
  - Potencialmente Entrando (PE)
  - Potencialmente Saindo (PS)



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



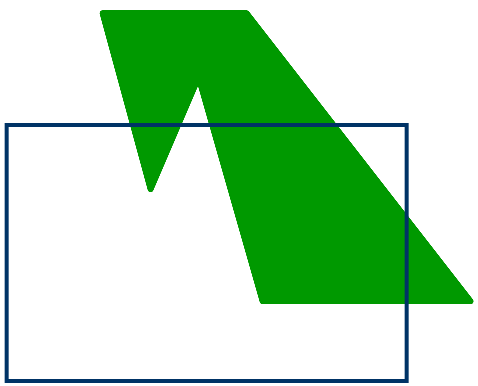
- $\boxed{\langle N_i, P_1 - P_0 \rangle > 0}$ implica PS $\rightarrow$

Cyrus Beck - Exercícios

1. Entenda o programa cyrus-beck disponível no site da disciplina.
2. Modifique a interface dada como tarefa no slide 31 para incluir a opção 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

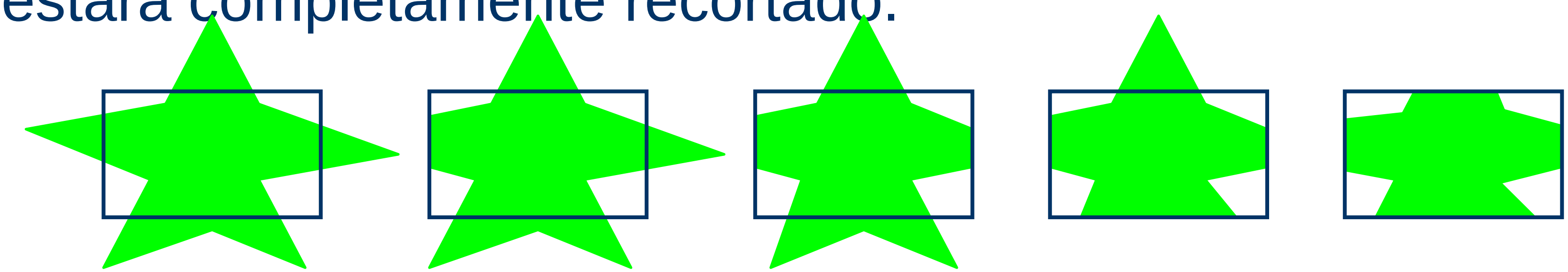
Algoritmo de Sutherland-Hodgeman

- 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.



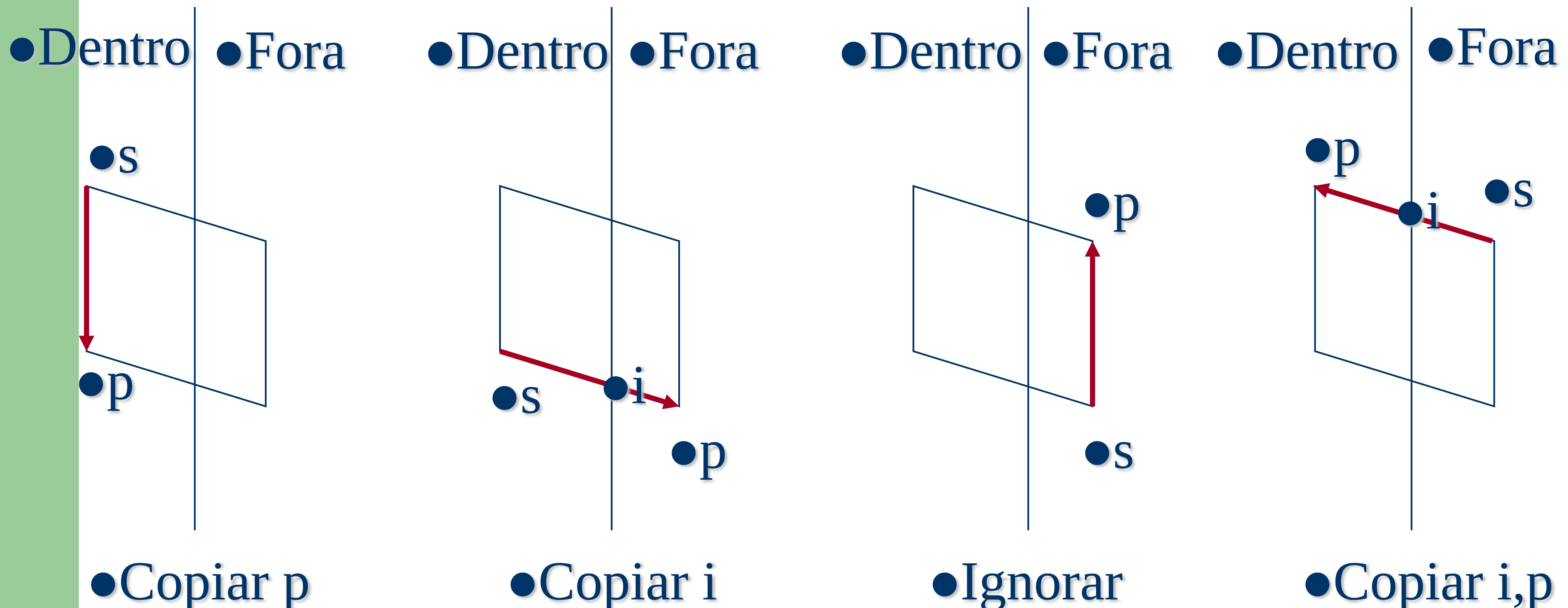
Polígono Original | Recorte esquerdo | Recorte direito | Recorte inferior | Recorte Superior

Algoritmo de Sutherland-Hodgeman

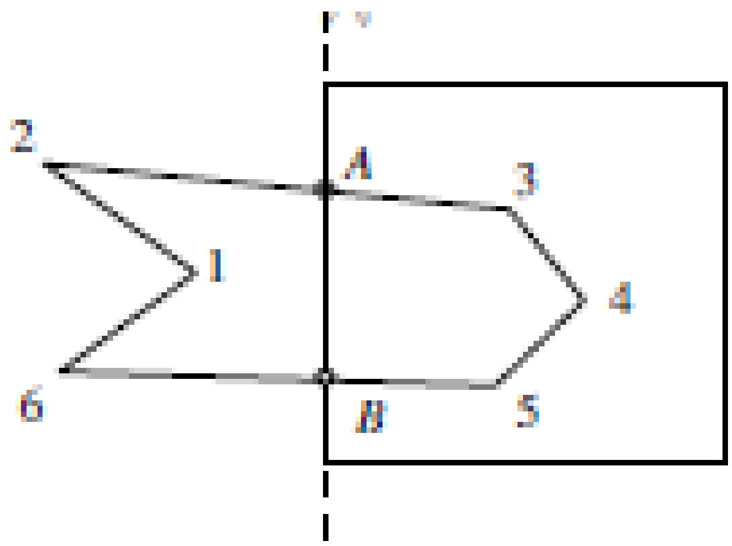
- Entrada/Saída do algoritmo
  - Entrada:
    - Região de recorte retangular
    - lista ordenada de vértices do polígono
  - Saída: lista dos vértices recortados, com alguns vértices originais (possivelmente) e outros novos (possivelmente)

Algoritmo de Sutherland-Hodgeman

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

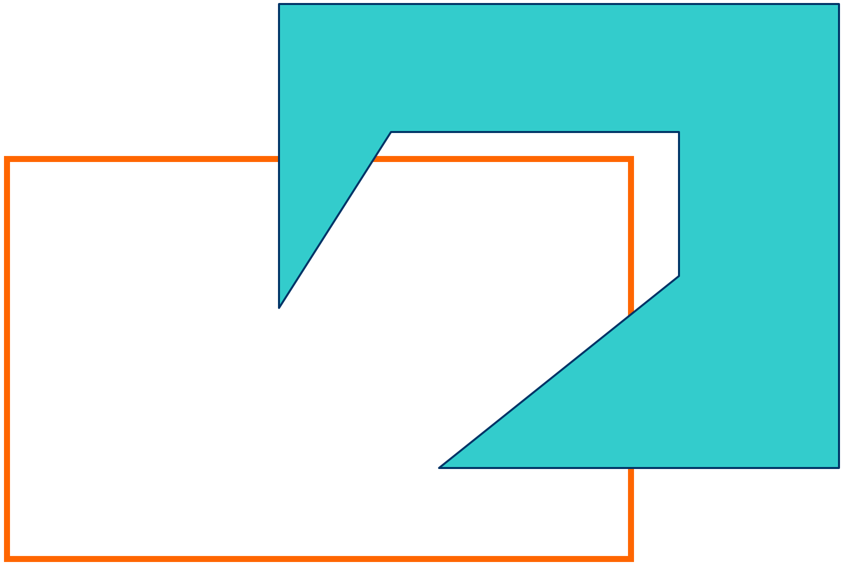


Algoritmo de Sutherland-Hodgeman

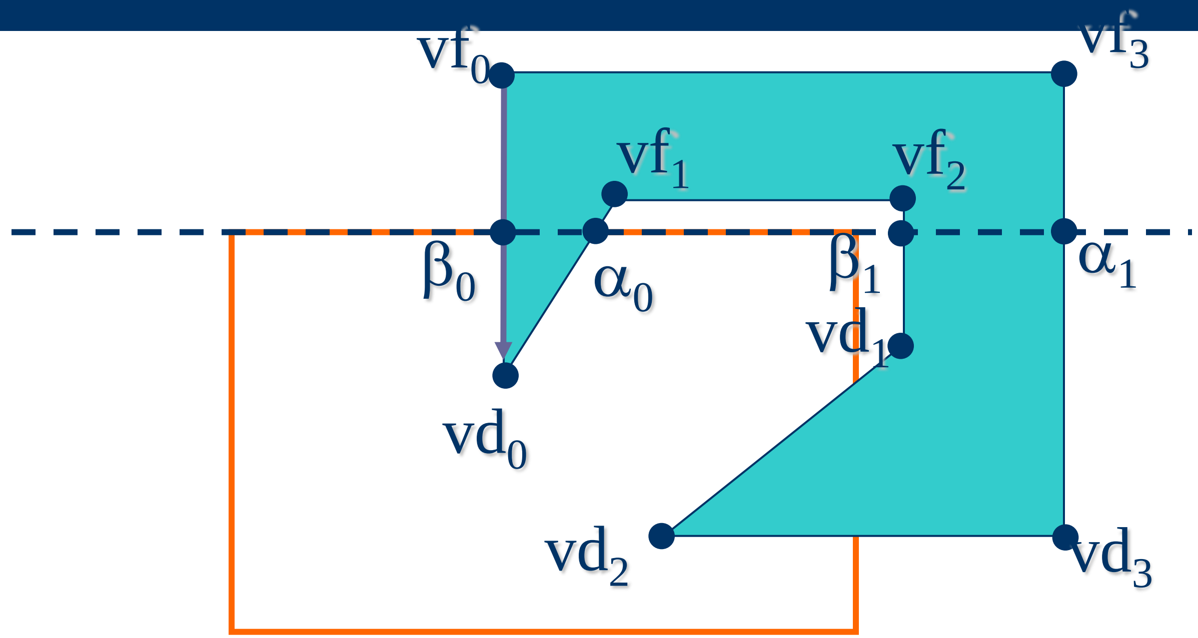


S	P	Ação
1	2	x
2	3	store A,3
3	4	store 4
4	5	store 5
5	6	store B
6	1	x

Algoritmo de Sutherland-Hodgeman



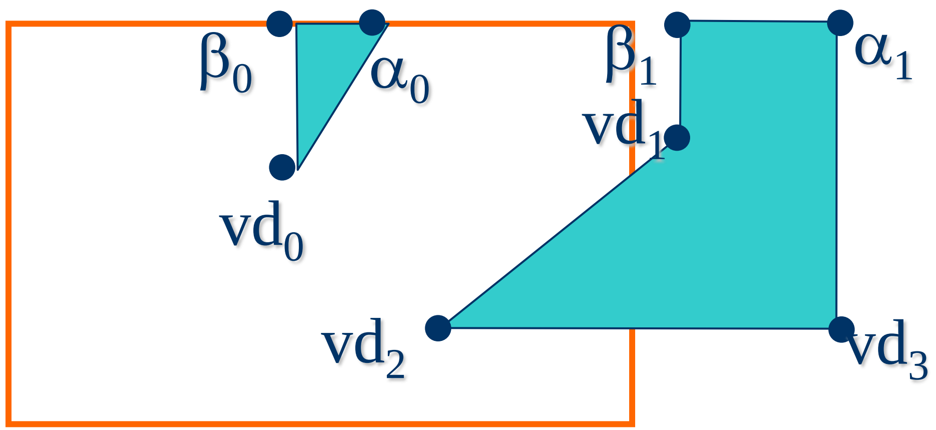
Algoritmo de Sutherland-Hodgeman



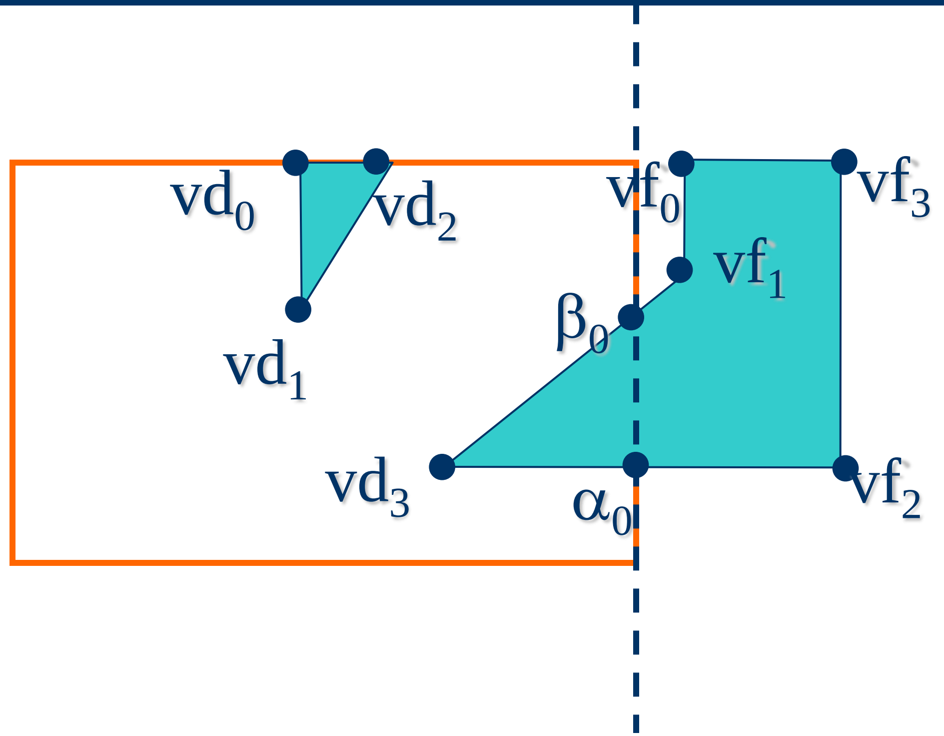
$vf_0\ \beta_0\ vd_0\ \alpha_0\ vf_1\ vf_2\ \beta_1 vd_1\ vd_2\ vd_3\ \alpha_1\ vf_3 vf_0$

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

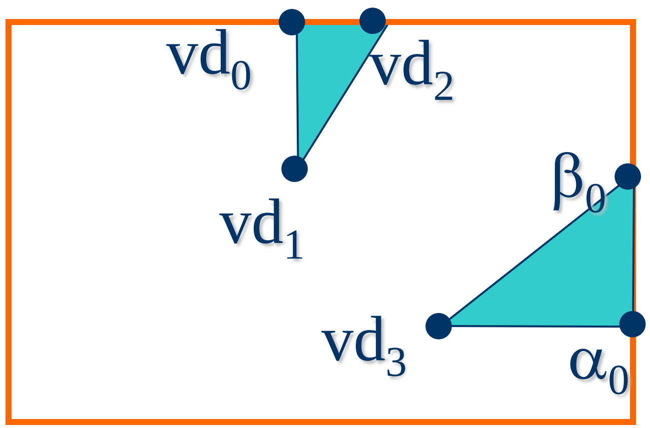
Algoritmo de Sutherland-Hodgeman



Algoritmo de Sutherland-Hodgeman

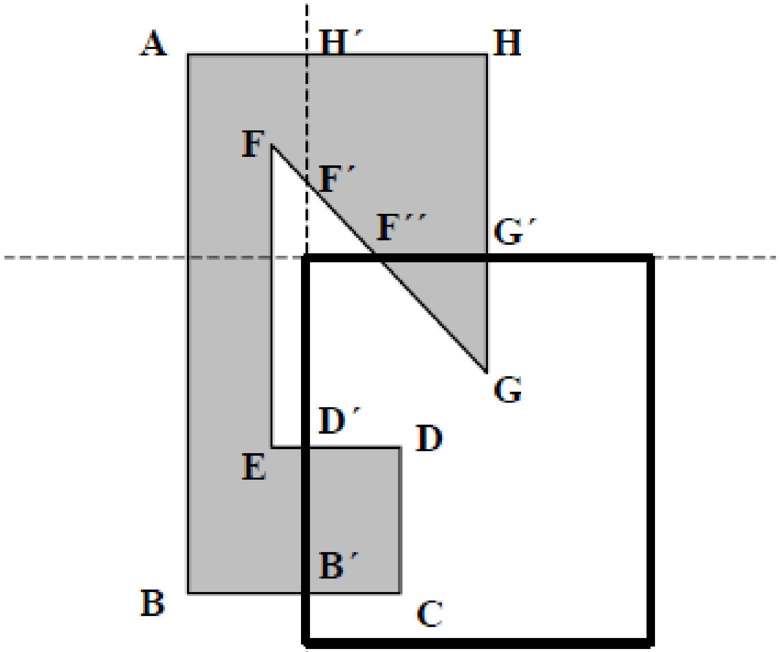


Algoritmo de Sutherland-Hodgeman



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 45 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.