

INF01047

Recorte 2D



Motivação

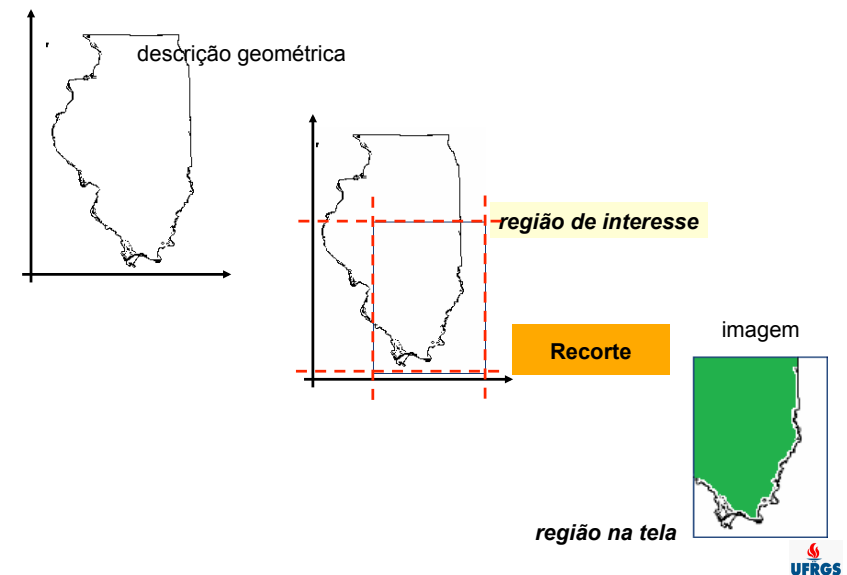
- Nossos dispositivos de saída são limitados
- Nossa visão humana é limitada (não vemos através de paredes 😊)
- No processo de síntese de imagens em CG, aparece a necessidade de *definir regiões de interesse*



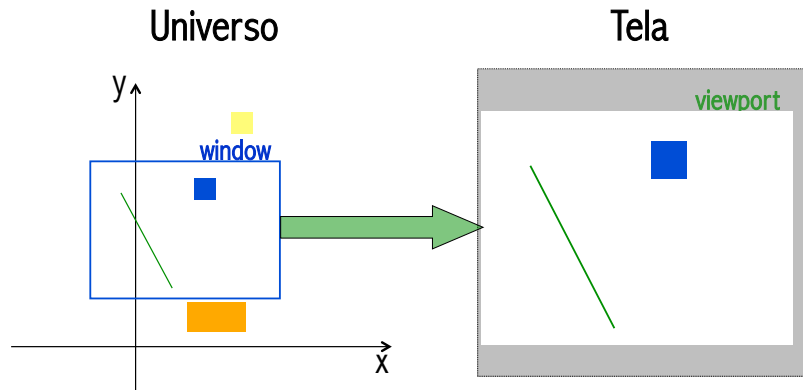
Problema

- Dada uma coleção de objetos numa cena, representados de alguma forma conveniente:
 - 1) definiremos uma região de interesse (usualmente retangular, mas não necessariamente)
 - 2) apenas os objetos (ou partes deles) dentro da região de interesse serão exibidos

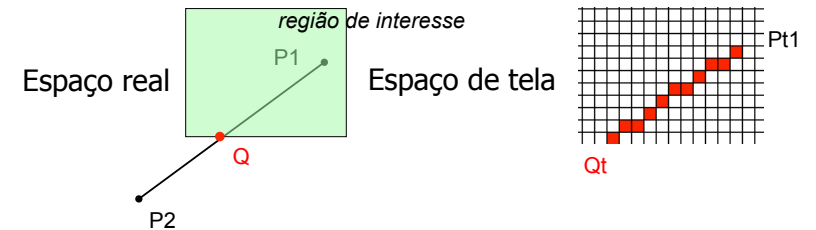
Removendo elementos fora da região



Relação entre recorte e rasterização

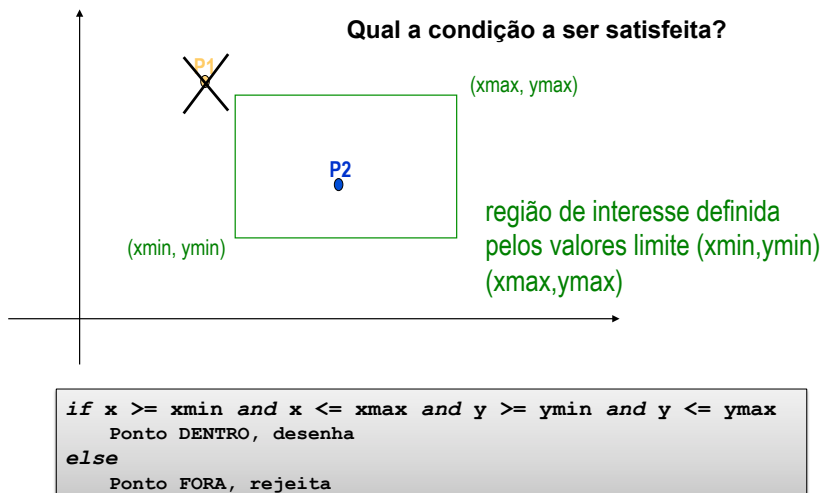


Recorte

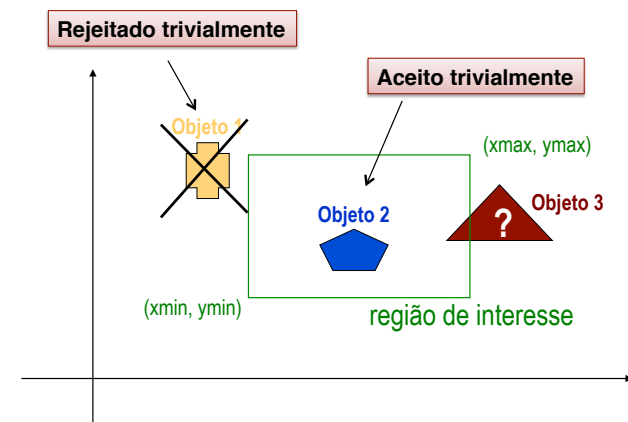


- Algoritmos de recorte permitem eliminar elementos geométricos inteiros ou partes deles
 - Recorte de pontos
 - Recorte de objetos
 - Recorte de linhas

Recorte de pontos



Recorte de objetos

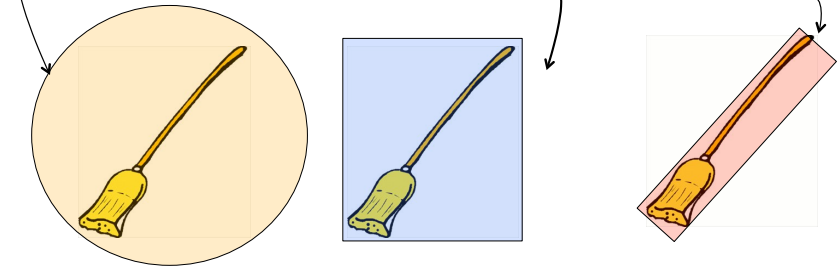


Recorte de objetos: algoritmo

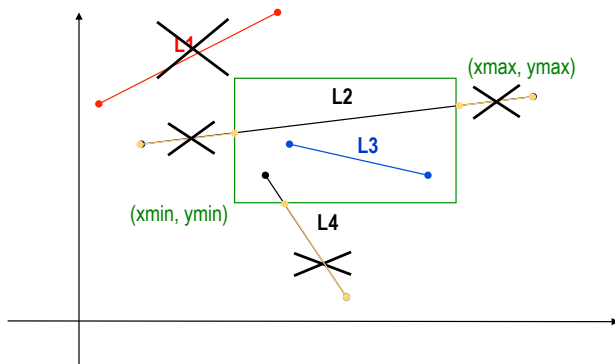
- Determina **ENVELOPE** do objeto
 - (Oxmin, Oymax) → (Oxmax, Oymax)
- Se envelope **DENTRO**
 - Desenha objeto
- Se envelope **FORA**
 - Descarta objeto
- Se envelope intercepta região de interesse
 - Recorta polígono

Envelope de objetos

- “Bounding volumes”
 - Bounding spheres
 - AABB: axis-aligned bounding box
 - OBB: object-aligned bounding box



Recorte de linhas (idéia)



Recorte de linhas: algoritmo?

if P1 = DENTRO and P2 = DENTRO
– Desenha linha P1 → P2

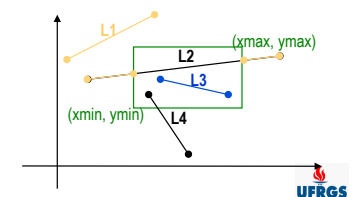
Linha L3

if P1 = DENTRO and P2 = FORA or
P1 = FORA and P2 = DENTRO
– Acha interseção da linha com a região de interesse (qual a
borda?)
– Redefine P2 (ou P1)
– Desenha linha P1 → P2 (ou P2 → P1)

Linha L4

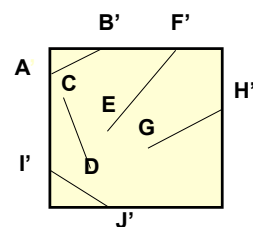
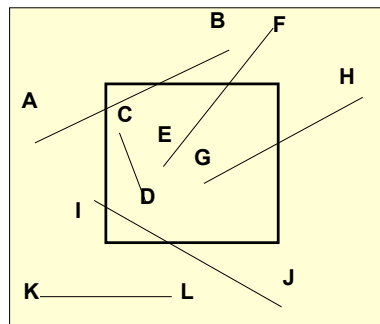
if P1 = FORA and P2 = FORA
???

Linhas L1 e L2



Algoritmos de recorte

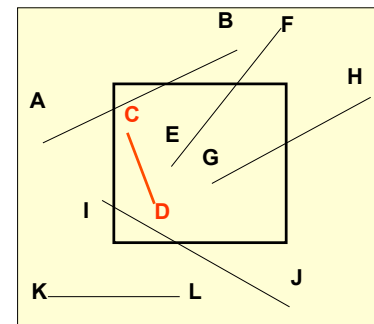
Recorte de linhas



Resultado desejado

Algoritmos de recorte

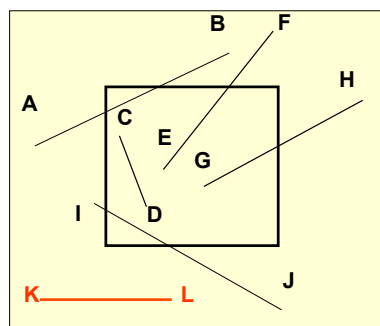
Recorte de linhas: Trivialmente aceito



Pontos estão dentro da região de interesse

Algoritmos de recorte

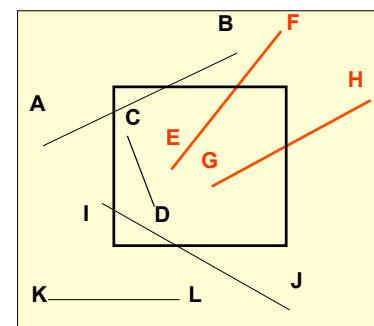
Recorte de linhas: Trivialmente recusado



Os dois pontos estão fora do retângulo e linha não cruza região de interesse

Algoritmos de recorte

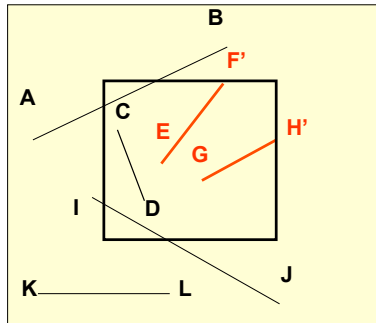
Recorte de linhas: Cálculos de recorte



Um dos pontos da linha está fora e outro está dentro da região de interesse

Algoritmos de recorte

Recorte de linhas: Cálculos de recorte

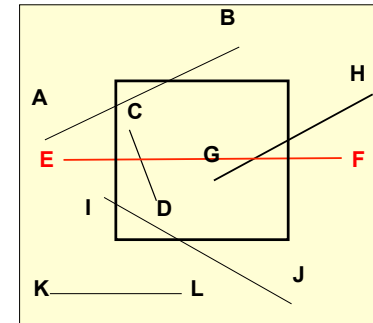


Um dos pontos da linha está fora e outro está dentro da região de interesse

Linha deve ser recortada

Algoritmos de recorte

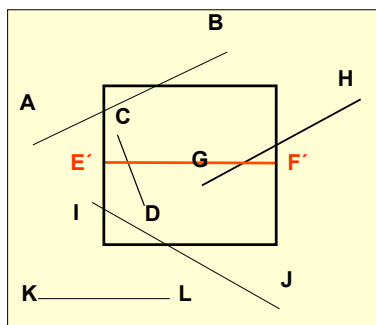
Recorte de linhas: Cálculos de recorte



Os dois pontos estão fora, mas linha cruza região de interesse

Algoritmos de recorte

Recorte de linhas: Cálculos de recorte



Os dois pontos estão fora, mas linha cruza região de interesse

Linha deve ser recortada

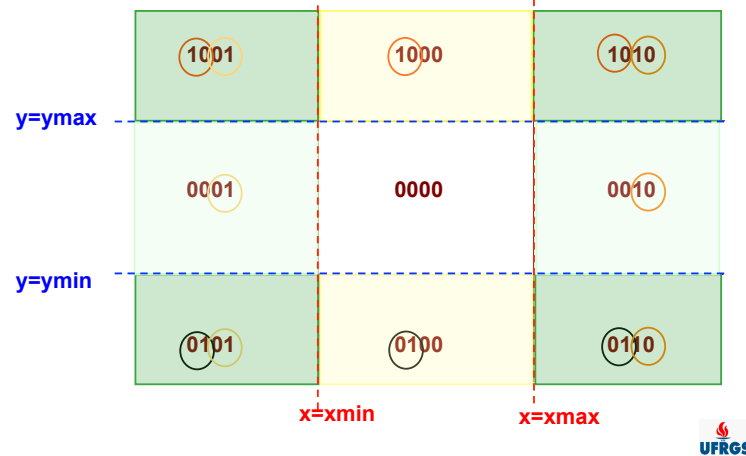
Algoritmo de Cohen e Sutherland (1967)



- Objetivos
 - Recorte de Linhas
 - Maneira eficiente de determinar os diferentes casos
 - Permitir a rejeição mais rápida de linhas totalmente fora da região de interesse
 - Facilitar a identificação da borda da região de interesse contra a qual se deve recortar a linha

Algoritmo de Cohen-Sutherland

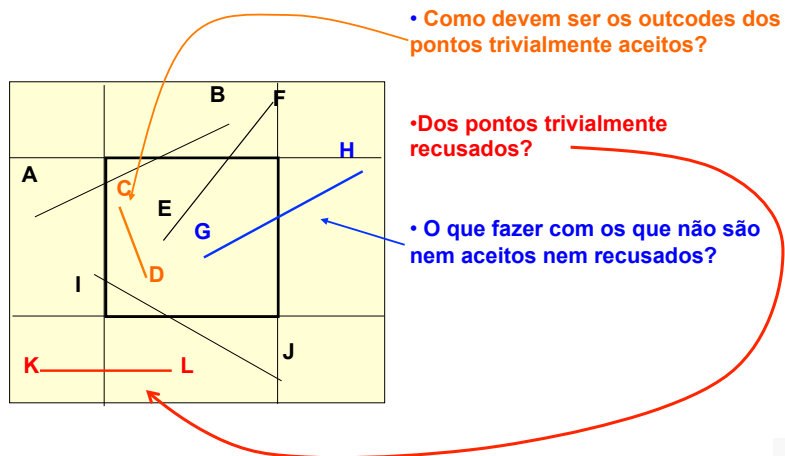
- Divide a região em 9 subespaços
- Atribuição de códigos binários aos espaços



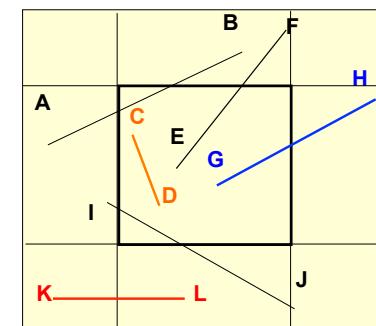
Cohen-Sutherland - Outcodes

If $y > ymax \rightarrow$ seta primeiro bit em 1
 If $y < ymin \rightarrow$ seta segundo bit em 1
 If $x > xmax \rightarrow$ seta terceiro bit em 1
 If $x < xmin \rightarrow$ seta quarto bit em 1

Cohen-Sutherland - Usando os outcodes



Cohen-Sutherland - Usando os outcodes

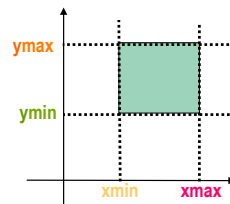


- Como devem ser os outcodes dos pontos trivialmente aceitos?
0000
- Dos pontos trivialmente recusados?
AND bitwise diferente de 0!
- O que fazer com os que não são nem aceitos nem recusados?
Recorte por segmentos

Passos

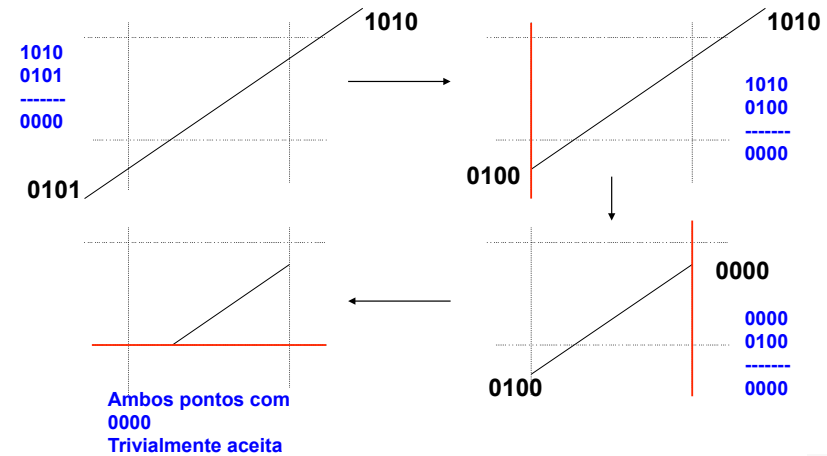
1. Determina código de região para extremidades da linha
2. Analisa o código
3. Caso a linha precise ser recortada, divide em segmentos pelo limite da janela
4. Recorte iterativo até que a linha passe o teste de trivialmente aceita ou trivialmente rejeitada

Abaixo: $x1xx$
 $y=ymin$
 Acima: $1xxx$
 $y=ymax$
 À direita: $xx1x$
 $x=xmax$
 À esquerda: $xxx1$
 $x=xmin$



Cohen-Sutherland

- Exemplo (ordem arestas arbitrária xmin,xmax,ymin,ymax)



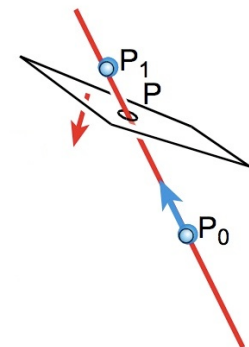
Applet exemplo

<http://www.cs.princeton.edu/~min/cs426/jar/clip.html>

Applet exemplificando o algoritmo

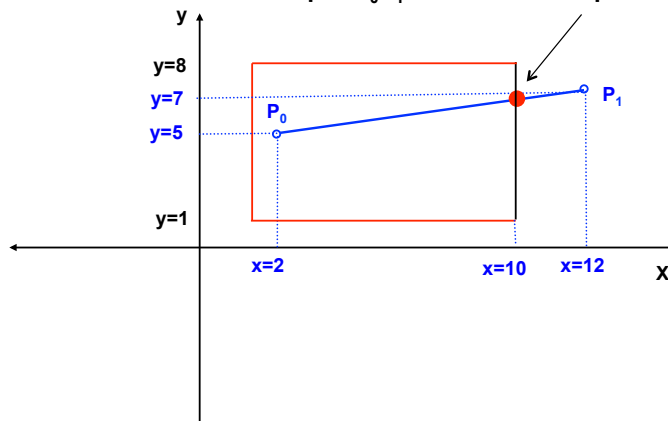
Calculando as intersecções

- Utilizamos a representação paramétrica de uma reta que passa por 2 pontos P_0 e P_1
- $L(t) = (1 - t)P_0 + tP_1$
- O ponto de recorte deve satisfazer a equação das duas retas: a reta sendo recortada e a reta que define a região de interesse



Exemplo

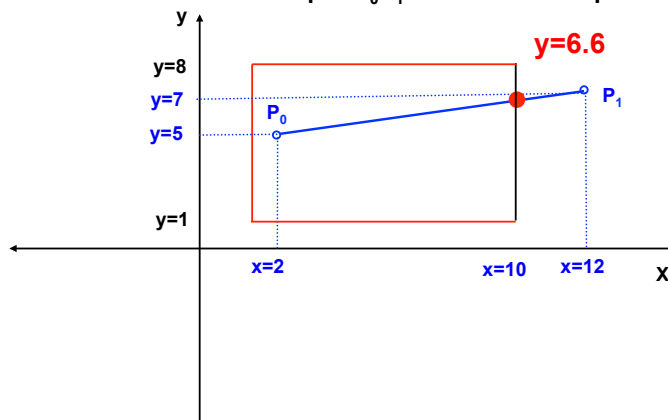
Qual o ponto de intersecção entre o segmento de reta dado por P_0P_1 e a aresta dada por $x=10$?



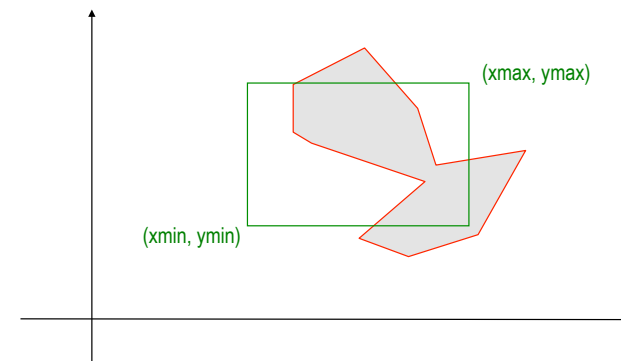
- Eq paramétrica da reta
 - $L(t) = (1 - t) P_0 + t P_1$
 - $Ly(t) = 5(1 - t) + 7t$
 - $Ly(t) = 5 + 2t$
- O ponto de intersecção satisfaz as 2 eqs:
 - $5 + 2t = 1 + 7t$
 - $5 - 1 + 2t - 7t = 0$
 - $4 - 5t = 0$
 - $t = 4/5$
- Substituindo em qq equação
 - $5 + 2 \cdot 4/5$
 - $5 + 8/5$
 - $33/5$
 - 6.6**

Exemplo

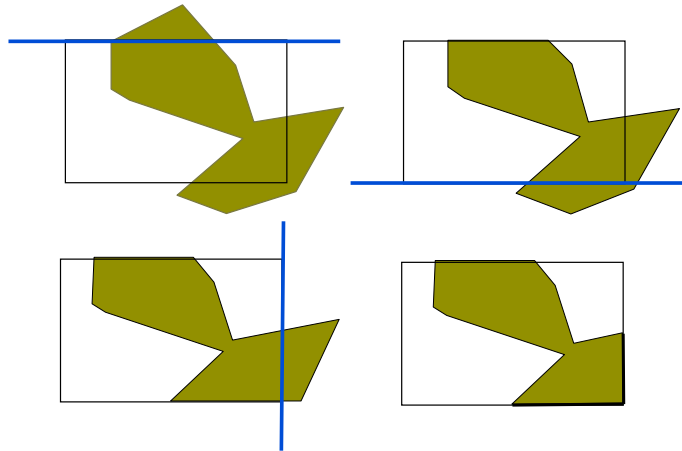
Qual o ponto de intersecção entre o segmento de reta dado por P_0P_1 e a aresta dada por $x=10$?



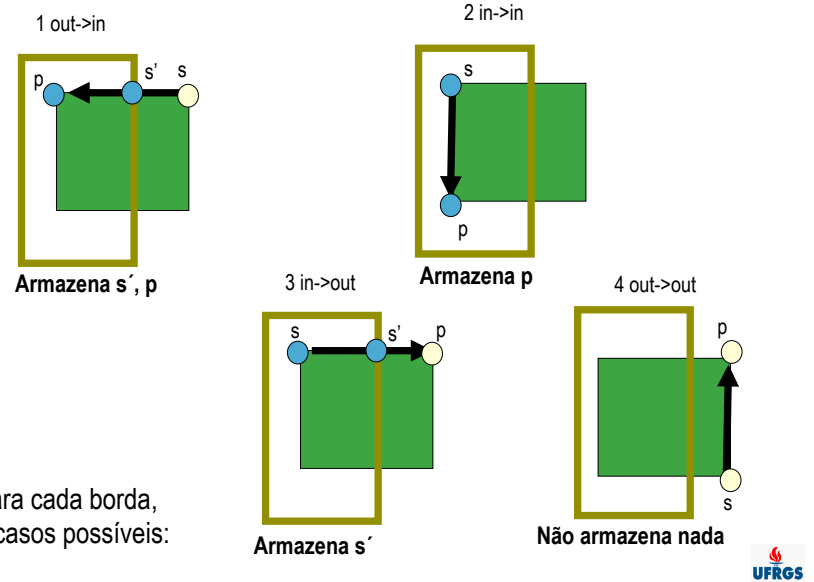
Recorte de polígono: algoritmo?



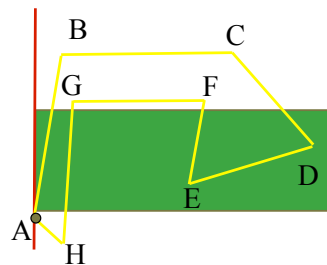
Sutherland-Hodgman (1974)



Sutherland-Hodgman



Sutherland- Hodgman



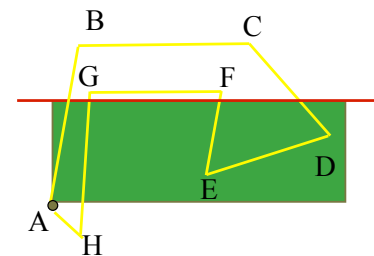
Recorte contra a **borda esquerda**

Lista inicial de vértices
A,B,C,D,E,F,G,H

Considerando que todos estão dentro (caso 2) a lista permanece a mesma

Lista de vértices
A,B,C,D,E,F,G,H

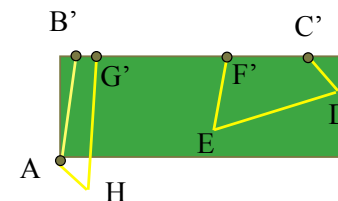
Sutherland- Hodgman



Recorte contra a **borda superior**

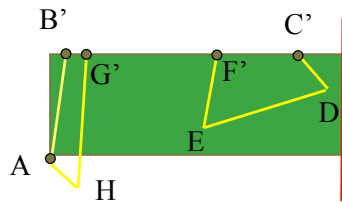
Lista de vértices
A,B,C,D,E,F,G,H

AB->B' (caso 3)
BC->nenhum (caso 4)
CD->C'D (caso 1)
DE->E (caso 2)
EF->F' (caso 3)
FG-> nenhum (caso 4)
GH->G'H (caso 1)
HA->A (caso 2)



Nova Lista de vértices
A,B',C',D,E,F',G',H

Sutherland- Hodgman



Recorte contra a **borda direita**

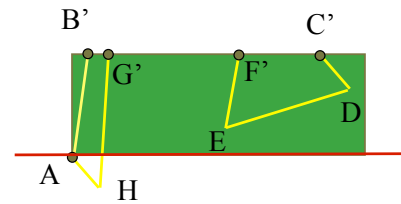
Lista de vértices
A,B',C',D,E,F',G',H

AB'→B' (caso 2)
B'C'→C' (caso 2)
C'D→D (caso 2)
DE→E (caso 2)
EF'→F' (caso 2)
F'G'→G' (caso 2)
G'H→H (caso 2)
HA→A (caso 2)

Lista de vértices permanece a mesma



Sutherland- Hodgman

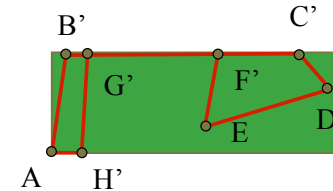


Recorte contra a **borda inferior**

Lista de vértices
A,B',C',D,E,F',G',H

AB'→B' (caso 2)
B'C'→C' (caso 2)
C'D→D (caso 2)
DE→E (caso 2)
EF'→F' (caso 2)
F'G'→G' (caso 2)
G'H→H' (caso 3)
HA→H'A (caso 1)

Lista final de vértices
A,B',C',D,E,F',G',H'



Recorte 2D - Resumo

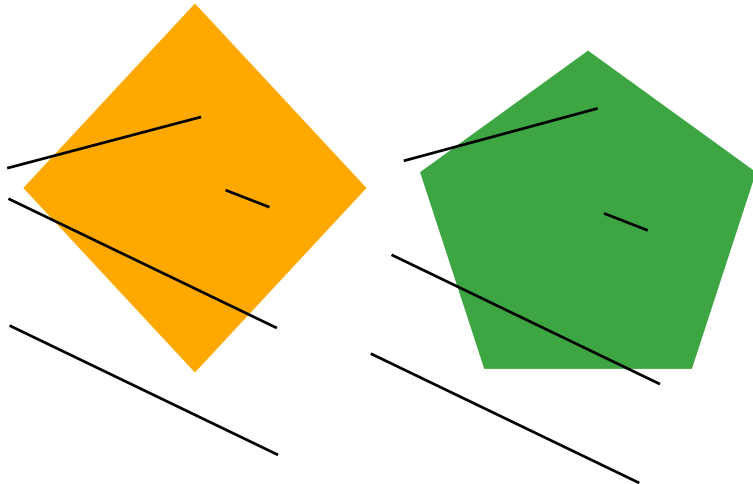
- Básico
 - Determina partes que estão dentro e fora da window
 - Redefine objeto, eliminando as partes que estão fora
- Algoritmos específicos
 - para objetos gráficos diferentes (pontos, linhas, círculos, etc.)
 - buscam otimização de procedimentos



Recorte



Área de interesse não é um retângulo!



Material Suplementar

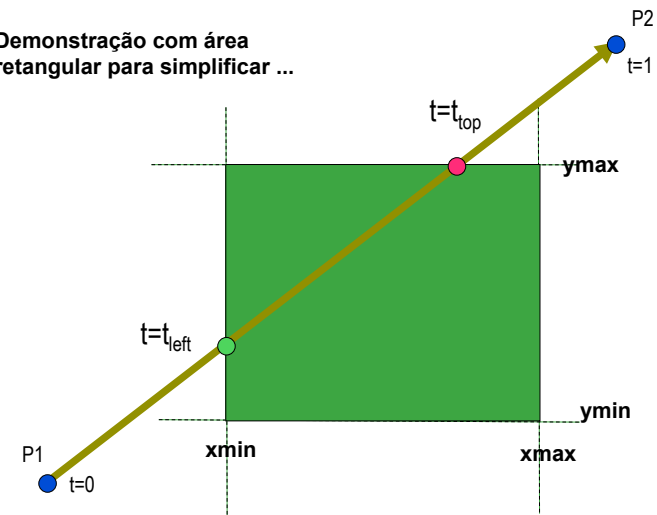
- Os slides a seguir apresentam um algoritmo para recorte no caso quando a área de interesse não é retangular

Cyrus-Beck (1978): idéia básica

- Segmento a ser recortado:
 $P(t) = P1 + t(P2 - P1) \quad 0 \leq t \leq 1$
 $t = (P(t) - P1) / (P2 - P1)$
- Intersecção com bordas da janela:
 $t_{left} = (x_{min} - x1) / (x2 - x1)$, borda esquerda
 $t_{right} = (x_{max} - x1) / (x2 - x1)$, borda direita
 $t_{top} = (y_{max} - y1) / (y2 - y1)$, borda superior
 $t_{bottom} = (y_{min} - y1) / (y2 - y1)$, borda inferior
- Se $0 \leq t_{_} \leq 1$ para qualquer $t_{_}$, este ponto pertence ao segmento
– Mas, o segmento é visível?

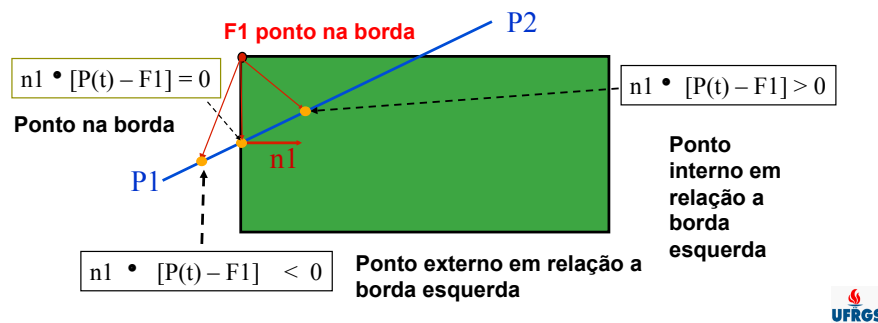
Cyrus-Beck (1978)

Demonstração com área retangular para simplificar ...



Cyrus-Beck (1978)

- Utilização de vetores normais internos a cada aresta da região de recorte
- Classificação dos pontos em relação às normais internas



Cyrus-Beck (1978)

Determinação de t sem calcular diretamente o ponto de intersecção

$$n_i \cdot [P_1 + (P_2 - P_1)t - F_i] = 0$$

$D = P_2 - P_1$, direção do segmento a ser recortado

$$n_i \cdot (P_1 + Dt - F_i) = 0$$

$$n_i \cdot (P_1 - F_i) + n_i \cdot Dt = 0$$

$$n_i \cdot (P_1 - F_i) = -(n_i \cdot Dt)$$

$$n_i \cdot (P_1 - F_i) = -(n_i \cdot D) t$$

Cyrus-Beck

- Cálculo de t para cada aresta da região convexa

$$t_i = -\frac{(P_1 - F_i) \cdot n_i}{D \cdot n_i}$$

F_i é substituído para cada limite da janela de recorte

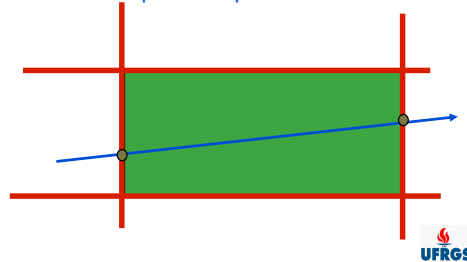
Cyrus-Beck

- O algoritmo busca os valores de t para os extremos do segmento “dentro da janela”
- Considera a orientação da linha sendo recortada para buscar t_{min} e t_{max}
- Avalia os valores de t obtidos para decidir se o segmento final deve ser “rejeitado” ou “aceito”

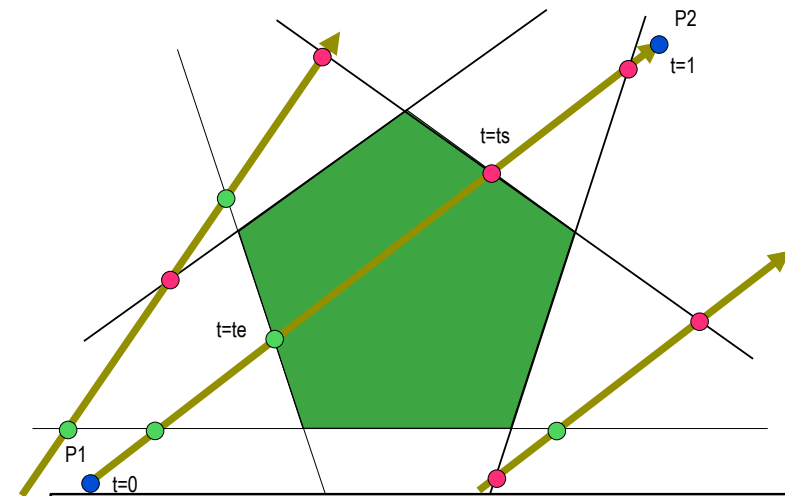
Cyrus-Beck (1978)

$t_{min} = 0$; $t_{max} = 1$

- Para cada aresta da borda:
 - calcula o t
 - verifica se deve atualizar o t_{min} ou o t_{max} (usando a orientação do segmento)
- Ao final, se $t_{min} < t_{max}$, o segmento é visível
- Descrição completa em
 - Rogers, David. Procedural Elements for Computer Graphics.



Cyrus-Beck



Vantagem: pode ser utilizado para recorte contra qualquer figura geométrica convexa