UNIFESP
Universidade Federal de São Paulo
1933

Viewing 2D

Profa. Ana Luísa D. Martins Lemos

May 1, 2018

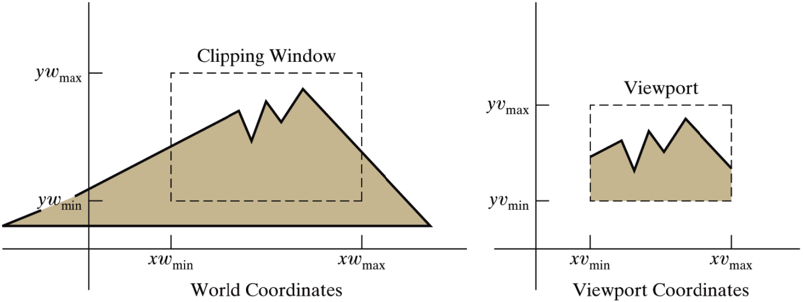
Introdução

- Viewing Pipeline 2D
  - Processo para criar a visão 2D de uma cena, determinando quais partes serão mostradas e suas localizações na tela
  - A imagem é determinada no **sistema de coordenadas do mundo (world coordinates)** cujas partes especificadas (selecionadas) são mapeadas para o **sistema de coordenadas do dispositivo (device coordinates)**
    - Esse mapeamento envolve uma série de translações, rotações e escalas
    - Assim como operações para eliminar as partes da imagem que estão fora da área de visão

Introdução

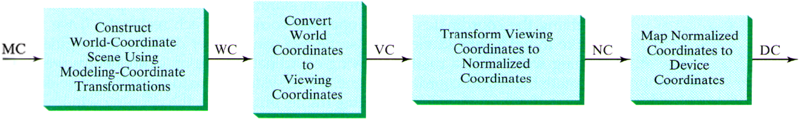
- Janela de Recorte ou Clipping Window
  - Uma seção de uma cena 2D que é selecionada para ser mostrada
    - Tudo o que estiver fora dessa seção será "cortado fora"
- Viewport
  - A *Janela de Recorte* pode ser posicionada dentro de uma janela do sistema usando outra "janela" chamada de **Viewport**
    - Objetos dentro da *Janela de Recorte* (o que será visto) são mapeados para a **Viewport**, que por sua vez é posicionada dentro da janela do sistema (onde serão vistos)
    - **Múltiplas Viewports** podem ser usadas para mostrar diferentes seções da imagem em diferentes posições

Introdução


Clipping Window
yw_{max}
yw_{min}
xw_{min}
xw_{max}
World Coordinates
Viewport
yv_{max}
yv_{min}
xv_{min}
xv_{max}
Viewport Coordinates

Introdução

- Transformação 2D da Visão
  - Mapeamento de uma descrição da cena no sistema de coordenadas do mundo para o sistema de coordenadas do dispositivo



- Para acelerar o processo de **recorte**, sistemas gráficos convertem a descrição dos objetos para **coordenadas normalizadas** (entre 0 e 1 ou entre -1 e 1)

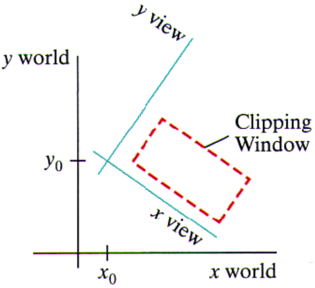
Janela de Recorte

- Embora seja possível criar **Janelas de Recorte** de qualquer formato, a maioria das APIs gráficas somente suporta janelas retangulares alinhadas aos eixos x e y devido ao custo computacional
- Normalmente a **Janela de Recorte** é especificada no **sistema de coordenadas do mundo**

Janela de Recorte

Sistema de Coordenadas

- Normalmente a transformação de visão é definida em um *sistema de coordenadas de visão* dentro do *sistema de coordenadas do mundo*
  - Isso permite especificar uma *Janela de Recorte* retangular em qualquer posição
  - Uma visão das coordenadas do mundo é obtida transferindo a cena para as coordenadas de visão



Janela de Recorte

Sistema de Coordenadas

- Escolhe-se uma origem $P_0(x_0, y_0)$ no sistema de coordenadas de visão e uma orientação usando um vetor V que dá a direção y_{view}
  - V é chamado de **view-up vector** 2D
- Outra abordagem é definir um ângulo de rotação relativo a x ou y e a partir desse obter o **view-up vector**

Janela de Recorte

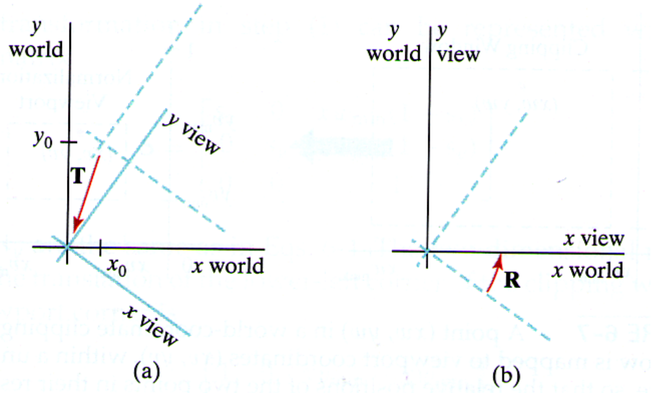
Sistema de Coordenadas

- Uma vez estabelecido o sistema de coordenadas de visão, é possível transformar a descrição dos objetos em uma cena usando translações e rotações para sobrepor os diferentes sistemas de coordenadas
  - Translado a origem P_0 para a origem do sistema de coordenadas do mundo
  - Rotaciono o sistema de visão para alinhá-lo com o sistema de coordenadas do mundo
- Essa conversão, entre coordenadas do mundo em coordenadas de visão é dada por

$$M_{WC,VC} = R.T$$

Janela de Recorte

Sistema de Coordenadas



Viewport

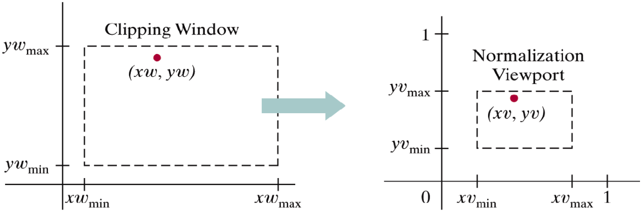
Normalização e Transformações

- Em alguns sistemas, a normalização e a transformação *window-viewport* são combinadas em uma única operação
  - Nesse caso as coordenadas da *viewport* são definidas entre 0 e 1
  - Após o recorte, o quadrado unitário contendo a *viewport* é mapeado para o dispositivo de saída
- Em outros sistemas a normalização e as rotinas de recorte são aplicadas antes das transformações de *viewport*
  - Nesse caso as coordenadas do *viewport* são as coordenadas da tela

Mapeamento

Janela para Viewport Normalizada

- Considerando uma *viewport* com as coordenadas entre 0 e 1, temos que mapear a descrição dos objetos para esse espaço normalizado usando transformações que mantenham a posição relativa de um ponto como foi definida na *Janela de Recorte*



- O ponto (xw, yw) é mapeado para (xv, yv)

Mapeamento

Janela para Viewport Normalizada

- Para transformar um ponto no sistema de coordenadas do mundo para um ponto na *viewport*, temos que fazer

$$\frac{xv - xv_{\min}}{xv_{\max} - xv_{\min}} = \frac{xw - xw_{\min}}{xw_{\max} - xw_{\min}}$$

$$\frac{yv - yv_{\min}}{yv_{\max} - yv_{\min}} = \frac{yw - yw_{\min}}{yw_{\max} - yw_{\min}}$$

- Resolvendo para a posição (xv, yv) na *viewport* temos

$$xv = s_x.xw + t_x$$

$$yv = s_y.yw + t_y$$

Mapeamento

Janela para Viewport Normalizada

- Onde os fatores de escala são

$$s_x = \frac{xv_{\max} - xv_{\min}}{xw_{\max} - xw_{\min}}$$

$$s_y = \frac{yv_{\max} - yv_{\min}}{yw_{\max} - yw_{\min}}$$

- E os fatores de translação são

$$t_x = \frac{xw_{\max}.xv_{\min} - xw_{\min}.xv_{\max}}{xw_{\max} - xw_{\min}}$$

$$t_y = \frac{yw_{\max}.yv_{\min} - yw_{\min}.yv_{\max}}{yw_{\max} - yw_{\min}}$$

Mapeamento

Janela para Viewport Normalizada

- Como simplesmente mapeamos o sistema de coordenadas do mundo para uma *viewport*, é possível obter o mesmo resultado usando uma sequência de transformações
  - Converter o retângulo da *Janela de Recorte* no retângulo da *viewport*
- Isso pode ser obtido fazendo
  1. Escala a *Janela de Recorte* para ter o tamanho da *viewport* usando o ponto fixo $(xw_{\min}, yw_{\min})$
  2. Translada $(xw_{\min}, yw_{\min})$ para $(xv_{\min}, yv_{\min})$

Mapeamento

Janela para Viewport Normalizada

- Onde a matriz de escala é

$$S = \begin{bmatrix} s_x & 0 & xw_{\min}(1 - s_x) \\ 0 & s_y & yw_{\min}(1 - s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

- E a matriz de translação é

$$T = \begin{bmatrix} 1 & 0 & xv_{\min} - xw_{\min} \\ 0 & 1 & yv_{\min} - yw_{\min} \\ 0 & 0 & 1 \end{bmatrix}$$

Mapeamento

Janela para Viewport Normalizada

- Sendo a matriz composta igual a

$$M_{\text{window,normviewport}} = T.S = \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

- Com s_x, s_y, t_x e t_y dados anteriormente

Mapeamento

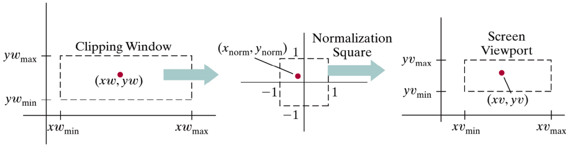
Janela para Viewport Normalizada

- Nesse mapeamento, as posições relativas dos objetos são mantidas
  - Um objeto dentro da *Janela de Recorte* estará dentro da *viewport*
- As proporções relativas dos objetos só serão mantidas se a razão de aspecto da *viewport* for igual à da *Janela de Recorte*
  - Em outras palavras s_x tem que ser igual a s_y

Mapeamento

Janela para Viewport Normalizada

- Um outra abordagem para a transformação de visão é transformar a *Janela de Recorte* em um quadrado normalizado, fazer o recorte em coordenadas normalizadas e então transferir a descrição da cena para a *viewport* especificada no sistema de coordenadas da tela



- Nessa representação, (parte dos) objetos fora dos limites $x = \pm 1$ e $y = \pm 1$ são facilmente detectados e removidos da cena

Mapeamento

Janela para Viewport Normalizada

- Para se mapear o conteúdo da *Janela de Recorte* para o quadrado normalizado procedemos similarmente a transformação *window-viewport* fazendo $xv_{\min} = yv_{\min} = -1$ e $xv_{\max} = yv_{\max} = +1$

$$M_{\text{window,normsquare}} = \begin{bmatrix} \frac{2}{xw_{\max}-xw_{\min}} & 0 & -\frac{xw_{\max}+xw_{\min}}{xw_{\max}-xw_{\min}} \\ 0 & \frac{2}{yw_{\max}-yw_{\min}} & -\frac{yw_{\max}+yw_{\min}}{yw_{\max}-yw_{\min}} \\ 0 & 0 & 1 \end{bmatrix}$$

Mapeamento

Janela para Viewport Normalizada

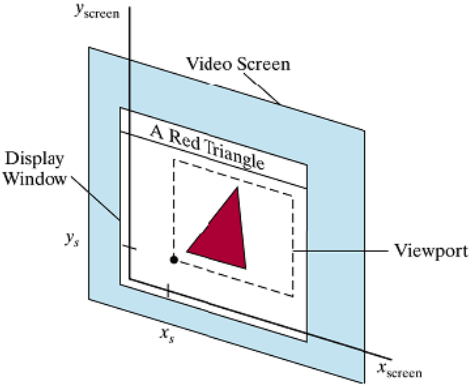
- Similarmente, após os algoritmos de recorte serem aplicados, o quadrado normalizado de tamanho 2 é transformado na *viewport* fazendo $xw_{\text{min}} = yw_{\text{min}} = -1$ e $xw_{\text{max}} = yw_{\text{max}} = +1$

$$M_{\text{normsquare,window}} = \begin{bmatrix} \frac{xv_{\text{max}} - xv_{\text{min}}}{2} & 0 & \frac{xv_{\text{max}} + xv_{\text{min}}}{2} \\ 0 & \frac{yv_{\text{max}} - yv_{\text{min}}}{2} & \frac{yv_{\text{max}} + yv_{\text{min}}}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

Mapeamento

Janela para Viewport Normalizada

- O último passo consiste em posicionar a área da *viewport* na janela da tela



Algoritmos de Recorte

- O *Viewing Pipeline* serve para extrair uma porção designada de uma cena para ser apresentada em um dispositivo de saída
- Identifica as partes de uma imagem que estão fora da *Janela de Recorte*, eliminando essas da descrição da cena que é passada para o dispositivo de saída
- Por eficiência, o recorte é aplicado sobre *Janelas de Recorte* normalizadas
    - Isso reduz cálculos porque todas as matrizes de transformação de geometria e visão podem ser concatenadas para serem aplicadas a uma cena antes do recorte acontecer

Algoritmos de Recorte

- Existem diversos algoritmos para o recorte de
  - Pontos
  - Linhas (segmentos de linhas retas)
  - Áreas-preenchidas (polígonos)
  - Curvas
  - Texto
- Os três primeiro são componentes padrão dos pacotes gráficos
  - Maior rapidez de processamento se as fronteiras dos objetos forem segmentos de reta

Algoritmos de Recorte

- Na discussão que se segue a região de recorte será uma janela retangular na posição padrão, com arestas de fronteira em $xw_{\min}$, $xw_{\max}$, $yw_{\min}$ e $yw_{\max}$
  - Tipicamente correspondendo ao quadrado normalizado entre 0 e 1 ou -1 e 1

Algoritmos de Recorte

Recorte de Ponto 2D

- Dado um ponto $P(x, y)$, esse será apresentado no dispositivo de saída se e somente se

$$xw_{\text{min}} \leq x \leq xw_{\text{max}}$$

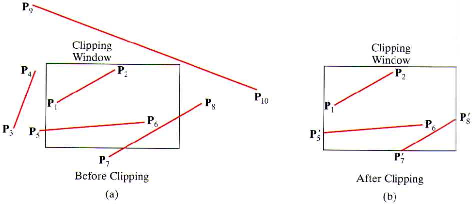
$$yw_{\text{min}} \leq y \leq yw_{\text{max}}$$

- Esse processo é especialmente útil para cortes em sistemas de partículas, como nuvens, fumaça, explosões, etc.

Algoritmos de Recorte

Recorte de Linha 2D

- Processa cada linha em uma cena por meio de uma série de testes e cálculos de interseção para determinar se uma linha ou parte dela precisa ser desenhada



- A tarefa mais cara computacionalmente é calcular as interseções das linhas com a *Janela de Recorte*
  - Portanto, o objetivo é minimizar o cálculo de interseções

Algoritmos de Recorte

Recorte de Linha 2D

- É fácil determinar se uma linha está completamente dentro da janela, mas é mais difícil determinar se essa linha está completamente fora
  - Quando os dois pontos limitantes de uma linha estão dentro da janela (linha $\overline{P_1 P_2}$), a linha está completamente dentro
  - Quando os dois pontos limitantes estão fora de qualquer uma das quatro fronteiras (linha $\overline{P_3 P_4}$), a linha está completamente fora
  - Se ambos testes falham, o segmento de linha intersecta ao menos uma das fronteiras da janela, e pode ou não cruzar o interior da mesma

Algoritmos de Recorte

Recorte de Linha 2D

- Partindo da definição paramétrica de um segmento de reta, com (x_0, y_0) e $(x_{\text{end}}, y_{\text{end}})$ temos que

$$x = x_0 + u(x_{\text{end}} - x_0)$$

$$y = y_0 + u(y_{\text{end}} - y_0)$$

$$0 \leq u \leq 1$$

- Podemos determinar a posição de interseção da reta com cada fronteira da janela substituindo o valor da coordenada da fronteira para x ou y e resolvendo para u
  - Se $0 > u > 1$, então não há cruzamento
  - Caso contrário, parte da reta está dentro da fronteira, e podemos processar esse parte contra as outras fronteiras até determinar se a reta será eliminada ou encontrar a seção que está dentro da janela

Algoritmos de Recorte

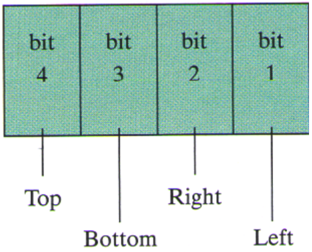
Recorte de Linha 2D

- Essa abordagem apesar de simples, não é muito eficiente
- É possível reformular o teste inicial e os cálculos de interseções para reduzir o tempo de processamento

Algoritmos de Recorte

Recorte de Linha 2D - Algoritmo de Cohen-Sutherland

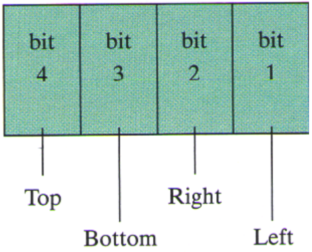
- Algoritmo de Recorte de Cohen-Sutherland:
  - Um dos primeiros algoritmos para acelerar o processo de recorte
  - O tempo de recorte é reduzido executando mais testes antes dos cálculos das interseções
  - Inicialmente a cada ponto final das linhas é assinalado um valor binário de 4 dígitos, o **código da região**



Algoritmos de Recorte

Recorte de Linha 2D - Algoritmo de Cohen-Sutherland

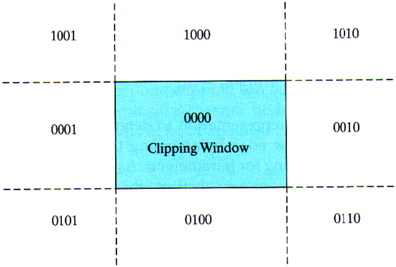
- Os valores binários indicam se o ponto está fora de uma fronteira
  - 0 (*false*): dentro ou sobre a fronteira
  - 1 (*true*): fora da fronteira



Algoritmos de Recorte

Recorte de Linha 2D - Algoritmo de Cohen-Sutherland

- As 4 fronteiras juntas criam nove regiões de separação do espaço



- Um ponto abaixo e à esquerda da *Janela de Recorte* recebe valor 0101, um ponto dentro 0000

Algoritmos de Recorte

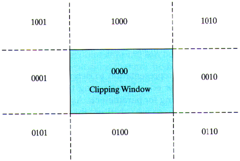
Recorte de Linha 2D - Algoritmo de Cohen-Sutherland

- Os valores dos bits são determinados comparando suas coordenadas (x, y) com as fronteiras de recorte
  - O bit 1 é definido como 1 se $x < xw_{\text{min}}$
  - Os outros são obtidos de forma similar
- É possível executar esse teste de forma mais eficiente usando operações binárias seguindo dois passos
  1. Calcular a diferença entre as coordenadas dos pontos e as fronteiras da janela
  2. Usar o sinal resultante para definir o valor do código (- vira 1, + vira 0)
     - bit 1 é o sinal de $x - xw_{\text{min}}$
     - bit 2 é o sinal de $xw_{\text{max}} - x$
     - bit 3 é o sinal de $y - yw_{\text{min}}$
     - bit 4 é o sinal de $yw_{\text{max}} - y$

Algoritmos de Recorte

Recorte de Linha 2D - Algoritmo de Cohen-Sutherland

- Com base nesses códigos é possível determinar rapidamente se uma linha está completamente fora ou dentro da janela
  - Linhas completamente dentro tem seus pontos definidos como 0000
  - Linhas que tenham 1 na mesma posição dos pontos finais estão completamente fora da janela de recorte
    - Uma linha com pontos finais identificados por 1001 e 0101 está completamente à esquerda da *Janela de Recorte*



Algoritmos de Recorte

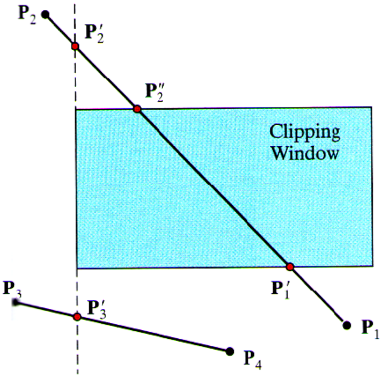
Recorte de Linha 2D - Algoritmo de Cohen-Sutherland

- Esses testes podem ser executados eficientemente usando operações lógicas
  1. Quando a operação **ou** entre dois pontos for **falsa** (0000) a linha está dentro
  2. Quando a operação **e** entre dois pontos for **verdadeira** (não 0000) a linha está completamente fora

Algoritmos de Recorte

Recorte de Linha 2D - Algoritmo de Cohen-Sutherland

- As linhas que não podem ser identificadas como completamente fora ou dentro da *Janela de Recorte* são então processadas para verificar interseções



Algoritmos de Recorte

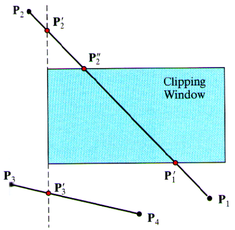
Recorte de Linha 2D - Algoritmo de Cohen-Sutherland

- Conforme cada interseção com as fronteiras da janela de recorte são calculadas, a linha é recortada até restar apenas o que está dentro da janela, ou nenhuma parte esteja dentro da mesma
- Para determinar se uma linha cruza alguma fronteira, é somente necessário verificar os bits correspondentes da fronteira dos pontos finais
  - Se um dos bits for 1 e outro 0, a linha cruza a fronteira

Algoritmos de Recorte

Recorte de Linha 2D - Algoritmo de Cohen-Sutherland

- Processando a fronteira esquerda
  - $P_1 = 0100 \rightarrow$ está dentro da fronteira esquerda
  - $P_2 = 1001 \rightarrow$ está fora da fronteira esquerda
    - Calcula a interseção P_2' e recorta a seção $\overline{P_2 P_2'}$



- As outras fronteiras seguem o mesmo princípio

Algoritmos de Recorte

Recorte de Linha 2D - Algoritmo de Cohen-Sutherland

- Para se determinar as interseções da reta definida pelos pontos (x_0, y_0) e $(x_{\mathrm{end}}, y_{\mathrm{end}})$ podemos usar a equação explícita

$$y = y_0 + m(x - x_0)$$

- O valor de x será xw_{min} ou xw_{max} e a inclinação será $m = (y_{\mathrm{end}} - y_0)/(x_{\mathrm{end}} - x_0)$
- Os valores de x da interseção podem ser calculados usando

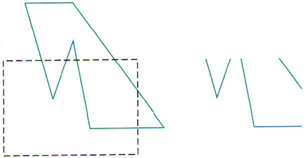
$$x = x_0 + \frac{y - y_0}{m}$$

- O valor de y será yw_{min} ou yw_{max}

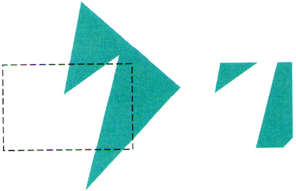
Algoritmos de Recorte

Recorte de Polígonos 2D

- Para fazer o recorte de polígonos, os algoritmos de recorte de linhas não podem ser aplicados porque em geral esses não produziriam polígonos fechados
  - Produziriam linhas desconexas sem informação de como uni-las para formar o polígono recortado



Before Clipping (a) After Clipping (b)

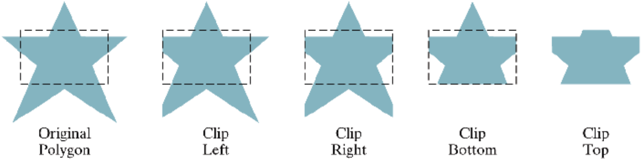


Before Clipping (a) After Clipping (b)

Algoritmos de Recorte

Recorte de Polígonos 2D

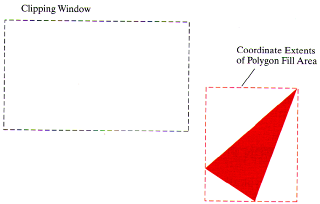
- É possível processar o polígono contra as fronteiras da *Janela de Recorte* de forma semelhante ao algoritmo de recorte de linhas
  - Isso é feito determinando o novo formato do polígono cada vez que uma fronteira de recorte é processada



Algoritmos de Recorte

Recorte de Polígonos 2D

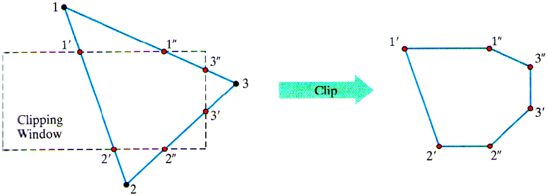
- É possível verificar se um polígono está completamente dentro ou fora da janela de recorte verificando suas coordenadas máximas e mínimas
- Quando uma área não puder ser identificada como completamente dentro ou fora, as interseções são calculadas



Algoritmos de Recorte

Recorte de Polígonos 2D

- Uma forma simples de realizar o recorte de **polígonos convexos** é criar uma nova lista de vértices a cada recorte realizado contra uma fronteira, e então passar essa lista para o próximo recorte, contra outra fronteira
- Para **polígonos côncavos** o processo é mais complexo podendo resultar em múltiplas listas de vértices



(a) (b)

Algoritmos de Recorte

Recorte de Polígonos 2D - Algoritmo de Sutherland-Hodgman

- Uma forma eficiente de realizar esse recorte é mandar os vértices dos polígonos para cada estágio de recorte de forma que os vértices recortados possam ser passados imediatamente para o próximo estágio
  - Elimina a necessidade de uma lista de novos vértices para cada estágio de recorte
  - Permite implementação paralela do recorte

Algoritmos de Recorte

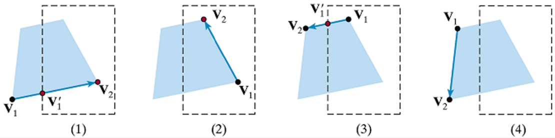
Recorte de Polígonos 2D - Algoritmo de Sutherland-Hodgman

- A estratégia deste algoritmo é mandar os pares de pontos finais de cada linha sucessiva do polígono para uma série de recortadores (esquerda, direita, inferior e superior)
  - Conforme o recorte é executado para um par de vértices, as coordenadas recortadas são enviadas para o próximo recortador

Algoritmos de Recorte

Recorte de Polígonos 2D - Algoritmo de Sutherland-Hodgman

- Existem 4 diferentes casos que precisam ser considerados quando uma aresta do polígono é processada
  1. O primeiro ponto final da aresta está fora da janela de recorte e o segundo dentro
  2. Ambos os pontos finais estão dentro da janela de recorte
  3. O primeiro ponto final da aresta está dentro da janela de recorte e o segundo fora
  4. Ambos os pontos finais estão fora da janela de recorte
- Para facilitar a passagem dos vértices de um recortador para outro, a saída de cada recortador pode ser da seguinte forma



Algoritmos de Recorte

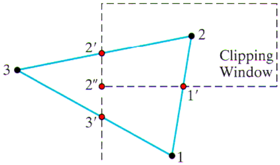
Recorte de Polígonos 2D - Algoritmo de Sutherland-Hodgman

- Conforme cada par de vértices sucessivos é passado para um dos recortadores, a saída é gerada para o próximo recortador de acordo com os seguintes testes
  1. Se o primeiro vértice está fora da janela e o segundo dentro, é mandado para o próximo recortador a interseção obtida e o segundo vértice
  2. Se ambos os vértices estão dentro, somente o segundo vértice é enviado
  3. Se o primeiro vértice está dentro da janela e o segundo fora, é mandado para o próximo recortador somente a interseção
  4. Se ambos os vértices estão fora, nada é enviado

UNIFESP
UNIVERSIDADE FEDERAL DE SÃO PAULO
1933

Algoritmos de Recorte

Recorte de Polígonos 2D - Algoritmo de Sutherland-Hodgman

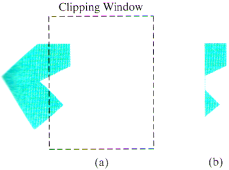


Input Edge:	Left Clipper	Right Clipper	Bottom Clipper	Top Clipper
$\{1, 2\}$:	(in – in) $\to \{2\}$			
$\{2, 3\}$:	(in – out) $\to \{2'\}$	$\{2, 2'\}$: (in – in) $\to \{2'\}$		
$\{3, 1\}$:	(out – in) $\to \{3', 1\}$	$\{2', 3'\}$: (in – in) $\to \{3'\}$	$\{2', 3'\}$: (in – out) $\to \{2''\}$	
		$\{3', 1\}$: (in – in) $\to \{1\}$	$\{3', 1\}$: (out – out) $\to \{\,\}$	
		$\{1, 2\}$: (in – in) $\to \{2\}$	$\{1, 2\}$: (out – in) $\to \{1', 2\}$	$\{2'', 1'\}$: (in – in) $\to \{1'\}$
			$\{2, 2'\}$: (in – in) $\to \{2'\}$	$\{1', 2\}$: (in – in) $\to \{2\}$
				$\{2, 2'\}$: (in – in) $\to \{2'\}$
				$\{2', 2''\}$: (in – in) $\to \{2''\}$

Algoritmos de Recorte

Recorte de Polígonos 2D - Algoritmo de Sutherland-Hodgman

- Limitação
  - Para polígonos côncavos, problemas podem ocorrer já que esse algoritmo apenas define como saída uma única lista de vértices

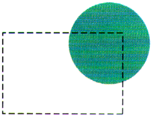


- Uma solução seria dividir o polígono côncavo em partes convexas

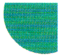
Algoritmos de Recorte

Recorte de Outras Primitivas 2D

- Áreas curvas podem ser recortadas usando abordagens parecidas com as apresentadas
  - Se as áreas curvas forem aproximações poligonais, o recorte é o mesmo apresentado anteriormente



Before Clipping



After Clipping