Viewing 2D

SCC0250 - Computação Gráfica

Prof. Fernando V. Paulovich
http://www.icmc.usp.br/~paulovic
paulovic@icmc.usp.br

Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)

6 de maio de 2010

Sumário

Sumário

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

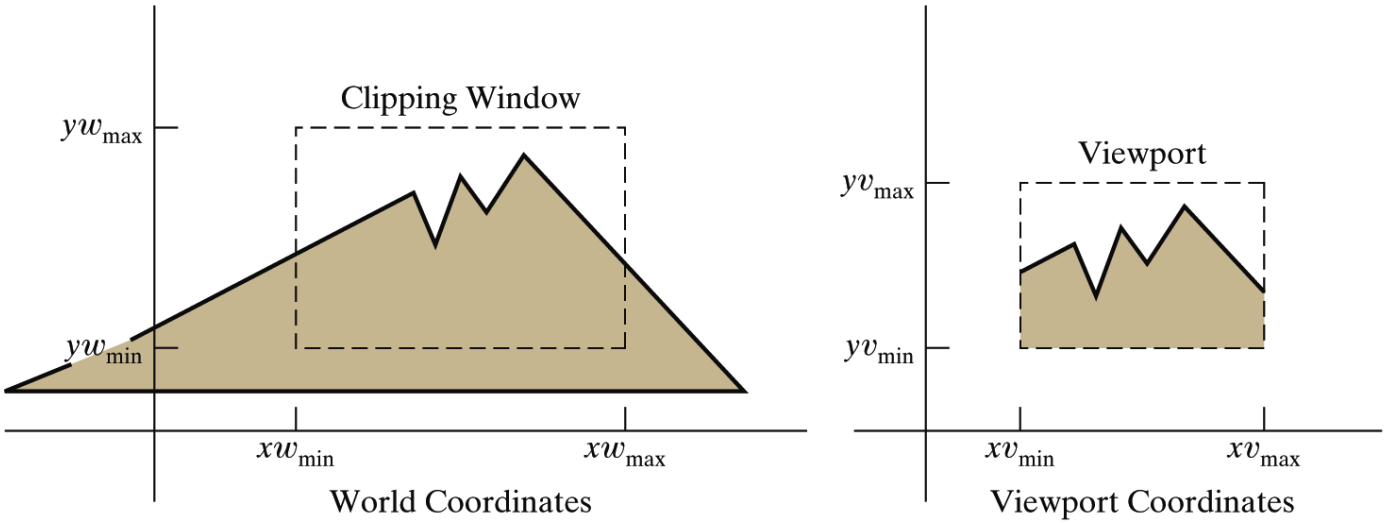
Janela de Recorte ou Clipping Window

- 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 mostra diferentes seções da imagem em diferentes posições

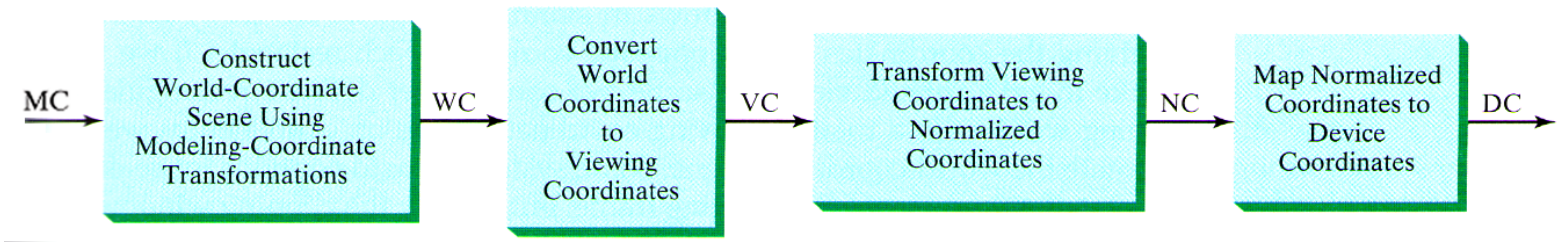
Introdução


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 que o processo de visão seja independente do dispositivo de saída, os sistemas gráficos convertem a descrição dos objetos para coordenadas normalizadas (entre 0 e 1 ou entre -1 e 1) e aplica as rotinas de *recorte*

Sumário

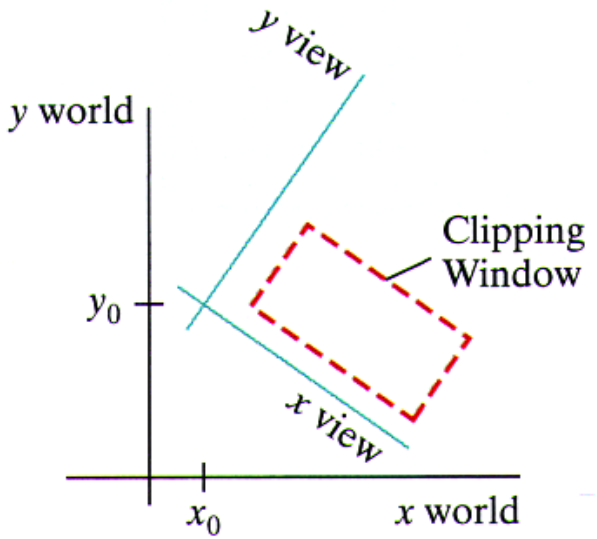
A Janela de Recorte

- Embora seja possível criar *Janelas de Recorte* de qualquer formato, a maioria as APIs gráficas somente suporta janelas retangulares alinhadas aos eixos x e y devido o custo computacional

- Normalmente a *Janela de Recorte* é especificadas no sistema de coordenadas do mundo

Sistema de Coordenadas de Visão da Janela de Recorte

- Normalmente, a transformação de visão é definida em um *sistema de coordenadas de visão* dentro do *sistema de coordenadas de 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



World Coordinates

Sistema de Coordenadas de Visão da Janela de Recorte

- Escolhe-se uma origem $\mathbf{P}_0 = (x_0, y_0)$ no sistema de coordenadas de visão e uma orientação usando um vetor $\mathbf{V}$ que dá a direção y_{view}
  - $\mathbf{V}$ é chamado de **view-up vector** 2D

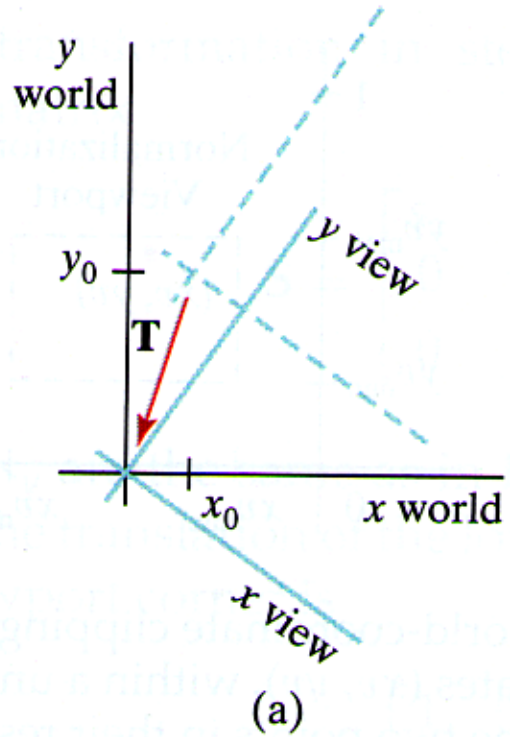
- Outra abordagem é definir um ângulo de rotação relativa a x ou y e a partir desse obter o **view-up vector**

Sistema de Coordenadas de Visão da Janela de Recorte

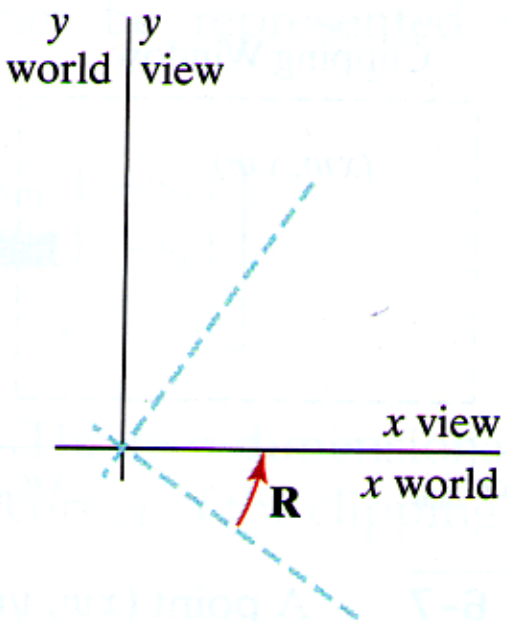
- Uma vez estabelecido o sistema de coordenadas de visão, é possível transformar a descrição dos objetos em uma cena usando translações e rotações para sobrepor os diferentes sistemas de coordenadas
  - Translado a origem $\mathbf{P}_0$ para a origem do sistema de coordenadas de mundo
  - Rotaciono o sistema de visão para alinhá-lo com o sistema de coordenadas de mundo
- Essa conversão, entre coordenadas do mundo em coordenadas de visão é dada por

$$\mathbf{M}_{WC,VC} = \mathbf{R} \cdot \mathbf{T}$$

Sistema de Coordenadas de Visão da Janela de Recorte



(a)



(b)

Sumário

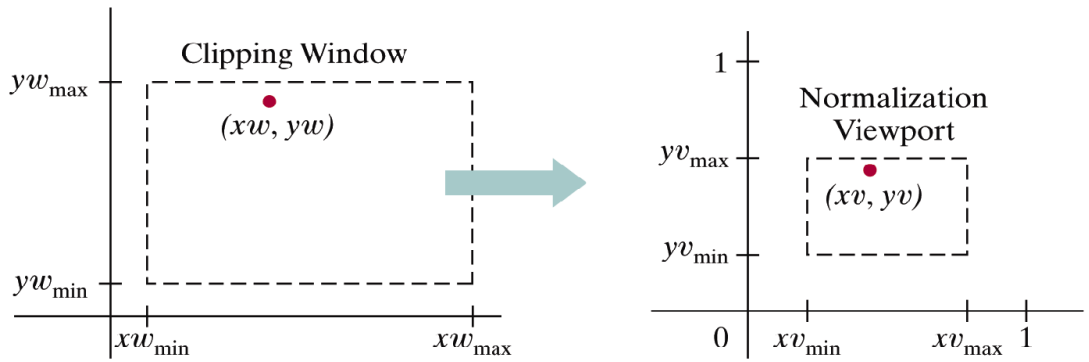
Normalização e Transformações de Viewport

- 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

Mapeando a Janela de Recorte em uma 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)

Mapeando a Janela de Recorte em uma 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 o posição (xv, yv) na *viewport* temos

$$xv = S_x xw + t_x$$

$$yv = S_y yw + t_y$$

Mapeando a Janela de Recorte em uma 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

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

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

Mapeando a Janela de Recorte em uma Viewport Normalizada

- Como simplesmente mapeamos o sistema de coordenadas de mundo para uma *viewport*, é possível obter o mesmo resultado usando uma sequencia 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})

Mapeando a Janela de Recorte em uma Viewport Normalizada

- Onde a matriz de escala é

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

- E a matriz de translação é

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

Mapeando a Janela de Recorte em uma Viewport Normalizada

- Sendo a matriz composta

$$\mathbf{M}_{window,normviewport} = \mathbf{T} \cdot \mathbf{S}$$

- Igual a

$$\mathbf{M}_{window,normviewport} = \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

- Com s_x, s_y, t_x e t_y dados anteriormente

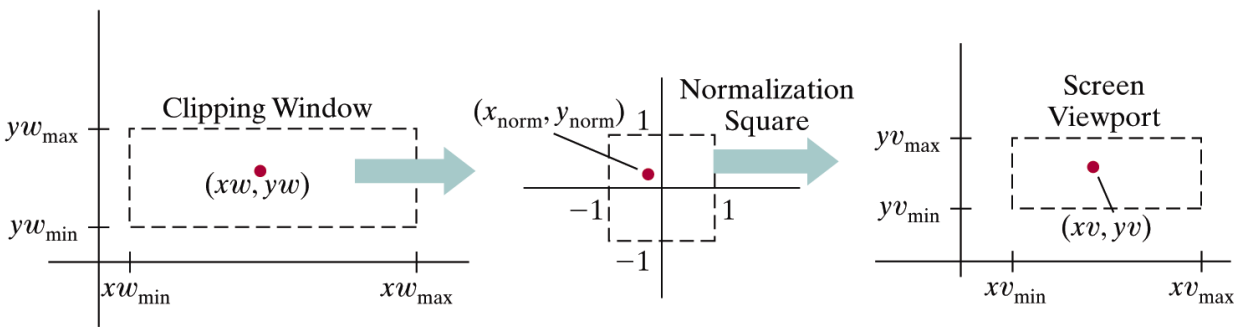
Mapeando a Janela de Recorte em uma Viewport Normalizada

- 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 a da *Janela de Recorte*
  - Em outras palavras s_x tem de ser igual a s_y

Mapeando a Janela de Recorte em um Quadrado Normalizado

- Uma 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

Mapeando a Janela de Recorte em um Quadrado Normalizado

- 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 $xv_{max} = yv_{max} = +1$

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

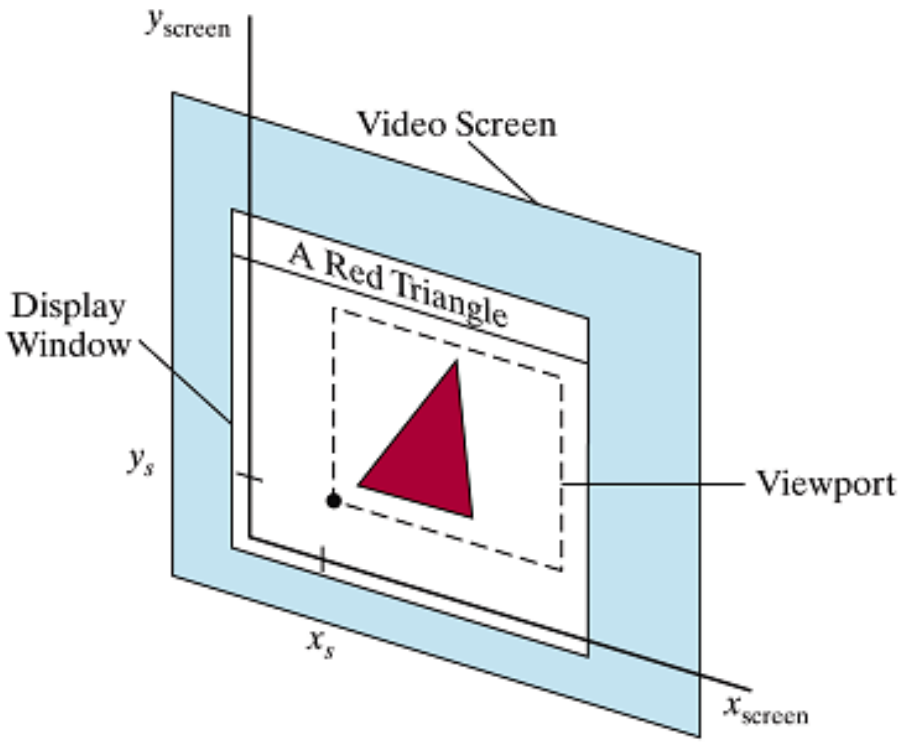
Mapeando a Janela de Recorte em um Quadrado Normalizado

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

$$\mathbf{M}_{normsquare,viewport} = \begin{bmatrix} \frac{xv_{max} - xv_{min}}{2} & 0 & \frac{xv_{max} + xv_{min}}{2} \\ 0 & \frac{yv_{max} - yv_{min}}{2} & \frac{yv_{max} + yv_{min}}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

Mapeando a Janela de Recorte em um Quadrado Normalizado

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



Sumário

Modo de Projeção OpenGL

- Antes de definir a *Janela de Recorte* e a *viewport*, é necessário definir que a matriz em uso é a matriz de projeção

```
1 glMatrixMode(GL_PROJECTION);
```

- Não esqueça que as transformações são cumulativas, então quando necessário carregar a matriz identidade

```
1 glLoadIdentity();
```

Definindo a Janela de Recorte

- A *Janela de Recorte* é definida por

```
1 glOrtho2D(GLfloat xwmin, GLfloat xwmax, GLfloat ywmin, GLfloat ywmax);
```

- Se a Janela de Recorte não for especificada, as coordenadas padrão serão $xw_{min} = yw = min = -1$ e $xw_{max} = yw_{max} = +1$
  - O processo de recorte ocorre em um quadrado normalizado entre -1 e 1

Definindo a Viewport

- A *viewport* é definida e posicionada por

```
1 glViewport(GLint xvmin, Glint yvmin, GLsizei vpWidth, GLsizei vpHeight);
```

- Todos os parâmetros são dados no sistema de coordenadas da tela, relativas a janela de visão
  - $(xvmin, yvmin)$: canto inferior esquerdo
  - $vpWidth$ e $vpHeight$: largura e altura da *viewport*

Exemplo

```
1  #include <GL/glut.h>
2  #include <stdlib.h>
3
4  void init(void) {
5    glClearColor(1.0, 1.0, 1.0, 0.0); //define a cor de fundo
6
7    glMatrixMode(GL_PROJECTION); //matrix em uso: projeção
8    glLoadIdentity(); //carrega a identidade
9    gluOrtho2D(-100.0, 100.0, -100.0, 100.0); //define janela de corte
10 }
11
12 void desenha_objeto() {
13   glBegin(GL_TRIANGLES); //desenha um triangulo
14     glVertex2i(50, -50);
15     glVertex2i(0, 50);
16     glVertex2i(-50, -50);
17   glEnd();
18
19   glBegin(GL_LINE_LOOP); //desenha um quadrado
20     glVertex2i(-100, -100);
21     glVertex2i(100, -100);
22     glVertex2i(100, 100);
23     glVertex2i(-100, 100);
24   glEnd();
25 }
```

Codificando

```
1  void desenha(void) {
2    glClear(GL_COLOR_BUFFER_BIT); //desenha o fundo (limpa a janela)
3
4    glMatrixMode(GL_MODELVIEW); //matrix em uso: modelview
5
6    glViewport(10, 10, 200, 200); //define a viewport
7    glColor3f(1.0, 0.0, 0.0); //altera o atributo de cor
8    desenha_objeto(); //desenha o objeto
9
10   glViewport(310, 10, 100, 100); //define a viewport
11   glColor3f(0.0, 1.0, 0.0); //altera o atributo de cor
12   glRotatef(90,0,0,1);
13   desenha_objeto(); //desenha o objeto
14
15   glFlush(); //processa as rotinas OpenGL o mais rápido possível
16 }
```

Codificando

```
1  int main(int argc, char**argv) {
2    glutInit(&argc, argv);
3    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
4    glutInitWindowPosition(50, 100);
5    glutInitWindowSize(550, 250);
6    glutCreateWindow("Titulo");
7
8    init(); // inicialização (após a criação da janela)
9    glutDisplayFunc(desenha); // registra a função de desenho
10   glutMainLoop(); // desenha tudo e espera por eventos
11
12   return EXIT_SUCCESS;
13 }
```

Sumário

Mantendo Razão de Aspecto

```
1  #include <GL/glut.h>
2  #include <stdlib.h>
3
4  void init(void)
5  {
6      //define cor de fundo (limpeza do frame buffer)
7      glClearColor(1.0, 1.0, 1.0, 0.0);
8  }
9
10 void display(void)
11 {
12     glClear(GL_COLOR_BUFFER_BIT); //desenha o fundo (limpa a janela)
13
14     glColor3f(1.0, 0.0, 0.0); //altera o atributo de cor
15
16     glBegin(GL_POLYGON); //desenha um quadrado
17       glVertex2i(-50, -50);
18       glVertex2i(50, -50);
19       glVertex2i(50, 50);
20       glVertex2i(-50, 50);
21     glEnd();
22
23     glFlush(); //processa as rotinas OpenGL o mais rápido possível
24 }
```

Mantendo Razão de Aspecto

```
1  void reshape(GLint width, GLint height)
2  {
3      // Evita a divisao por zero
4      if (height == 0) height = 1;
5
6      // Especifica as dimensões da Viewport
7      glViewport(0, 0, width, height);
8
9      // Inicializa o sistema de coordenadas
10     glMatrixMode(GL_PROJECTION);
11     glLoadIdentity();
12
13     // Estabelece a janela de seleção (left, right, bottom, top)
14     if (width <= height)
15     {
16         gluOrtho2D (-100, 100, -100*height/width, 100*height/width);
17     }
18     else
19     {
20         gluOrtho2D (-100*width/height, 100*width/height, -100, 100);
21     }
22 }
```

Mantendo Razão de Aspecto

```
1  int main(int argc, char**argv)
2  {
3      glutInit(&argc, argv);
4      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
5      glutInitWindowPosition(50, 100);
6      glutInitWindowSize(400, 400);
7      glutCreateWindow("Titulo");
8
9      init(); // inicialização (após a criação da janela)
10     glutDisplayFunc(display); // registra função de desenho
11     glutReshapeFunc(reshape); //registra função de alteração de tamanho
12     glutMainLoop(); // desenha tudo e espera por eventos
13
14     return EXIT_SUCCESS;
15 }
```

Sumário

Algoritmos de Recorte

Algoritmo de Recorte

- No *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 retos)
  - Áreas-preenchidas (polígonos)
  - Curvas
  - Texto

- Os três primeiros 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

Sumário

Recorte de Ponto 2D

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

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

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

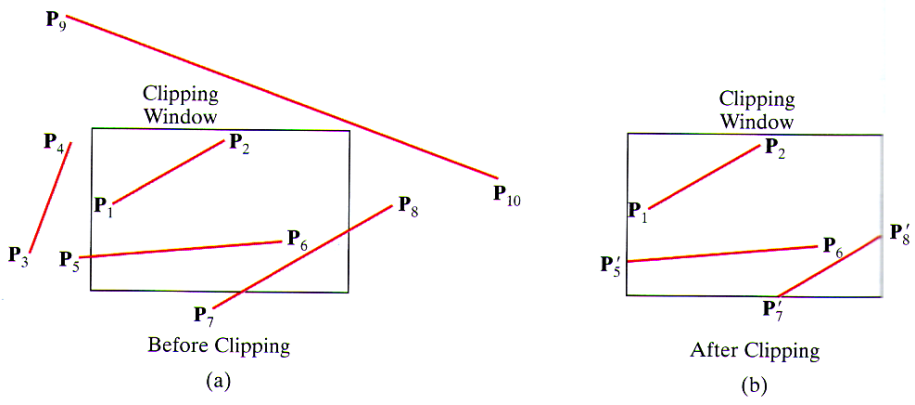
- Esse processo é especialmente útil para cortes em sistemas de partículas, como nuvens, fumaça, explosões, etc.

Sumário

Recorte de Linha 2D

Algoritmo de Recorte

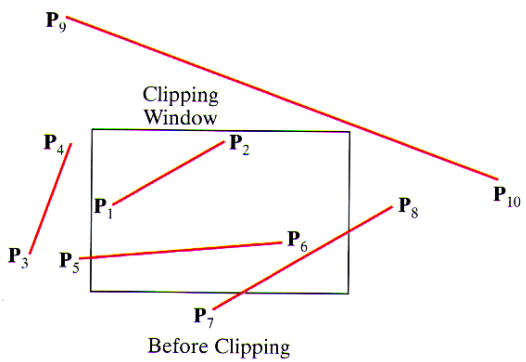
- Processa cada linha em uma cena por meio de uma série de testes e cálculos de intersecção para determinar se uma linha ou parte dela precisa ser desenhada

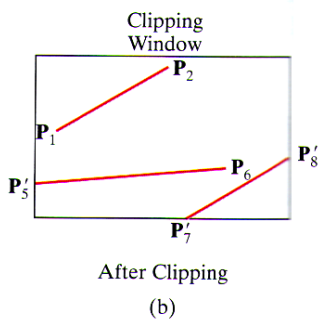


- A tarefa mais cara computacionalmente do recorte é calcular as intersecções das linhas com a *Janela de Recorte*
  - Portanto, o objetivo é minimizar o cálculo de intersecções

Recorte de Linha 2D

- É fácil determinar se uma linha está completamente dentro da janela, mas é mais difícil determinar se essa está completamente fora
  - Quando os dois pontos limitantes de uma linha estão dentro da janela (linha $\overline{\mathbf{P}_1\mathbf{P}_2}$), a linha está completamente dentro
  - Quando os dois pontos limitantes estão fora de qualquer uma das quatro fronteiras (linha $\overline{\mathbf{P}_3\mathbf{P}_4}$), 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





Recorte de Linha 2D

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

$$x = x_0 + u(x_{end} - x_0)$$
$$y = y_0 + u(y_{end} - y_0)$$
$$0 \leq u \leq 1$$

- 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 essa parte contra as outras fronteiras até determinar se a reta será eliminada ou encontrar a seção que está dentro da janela

Recorte de Linha 2D

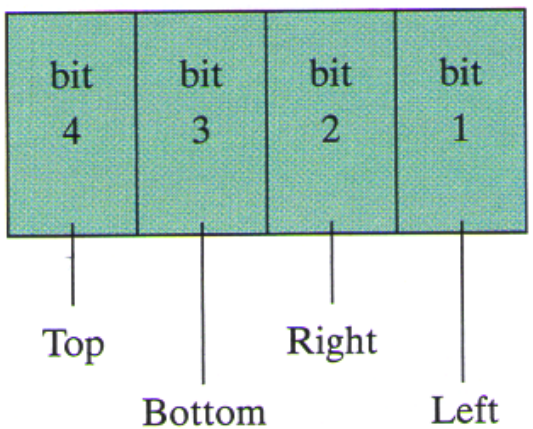
- Essa abordagem apesar de simples, não é muito eficiente
- É possível reformular o teste inicial e os cálculos de interseções para reduzir o tempo de processamento

Recorte de Linha de Cohen-Sutherland

Algoritmo de Recorte de Cohen-Sutherland

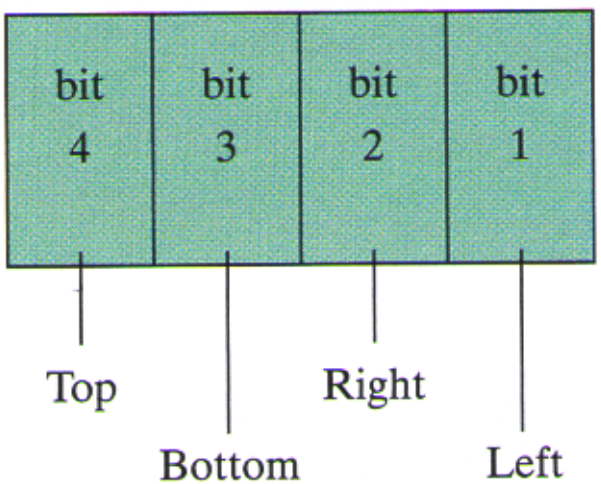
- Um dos primeiros algoritmos para acelerar o processo de recorte
- O tempo de recorte é reduzido executando mais testes antes dos cálculos das intersecções

- Inicialmente a cada ponto final das linhas é assinalado um valor binário de 4 dígitos, o **código da região**
  - Cada bit é usado para indicar se o ponto esta dentro ou fora de uma das fronteiras da janela de recorte



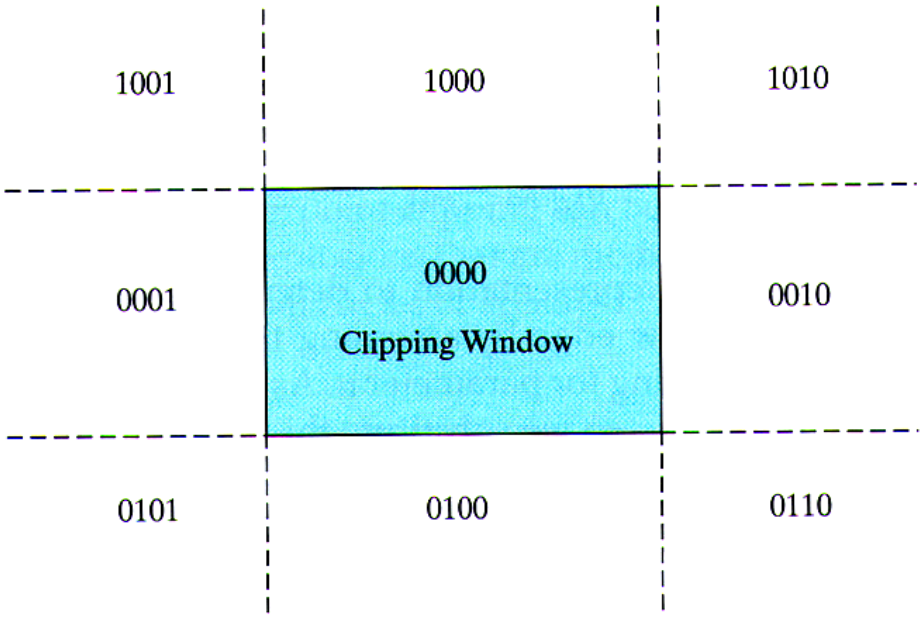
Recorte de Linha de Cohen-Sutherland

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



Recorte de Linha de Cohen-Sutherland

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



- Um ponto abaixo e a esquerda a janela de recorte recebe valor 0101, um ponto dentro 0000

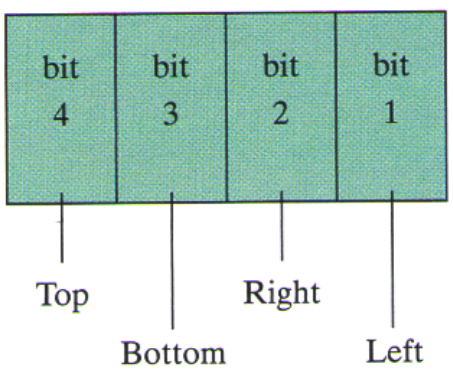
Recorte de Linha de Cohen-Sutherland

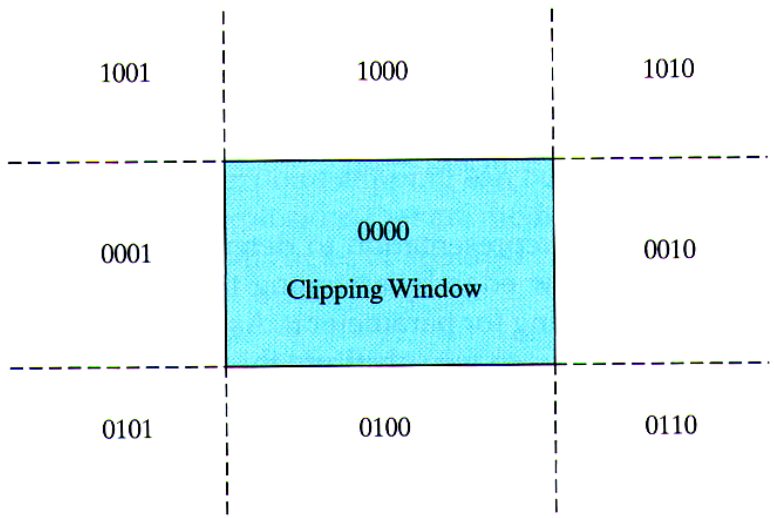
- Os valores dos bits são determinados comparando suas coordenadas (x, y) com as fronteiras de recorte
  - O bit 1 é definido como 1 se $x < xw_{min}$
  - Os outros são obtidos de forma similar

- 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
     - bit 1 é o sinal de $x - xw_{min}$
     - bit 2 é o sinal de $xw_{max} - x$
     - bit 3 é o sinal de $y - yw_{min}$
     - bit 4 é o sinal de $yw_{max} - y$

Recorte de Linha 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 nas mesmas posições dos pontos finais está completamente fora da janela de recorte
    - Uma linha com pontos finais identificados por 1001 e 0101 está completamente a esquerda da janela de recorte



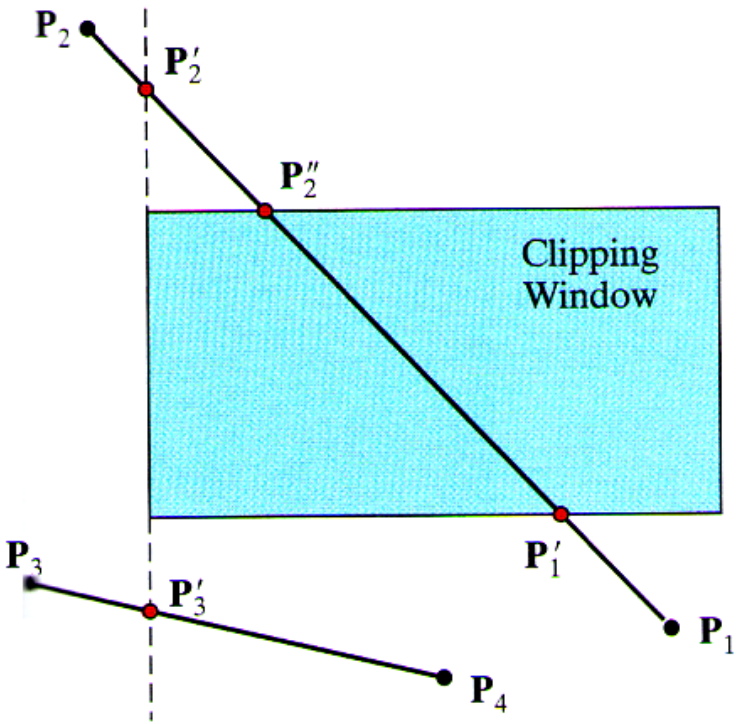


Recorte de Linha de Cohen-Sutherland

- 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

Recorte de Linha de Cohen-Sutherland

- As linhas que não podem ser identificadas como completamente fora ou dentro da janela de recorte são então processadas para verificar intersecções



Recorte de Linha de Cohen-Sutherland

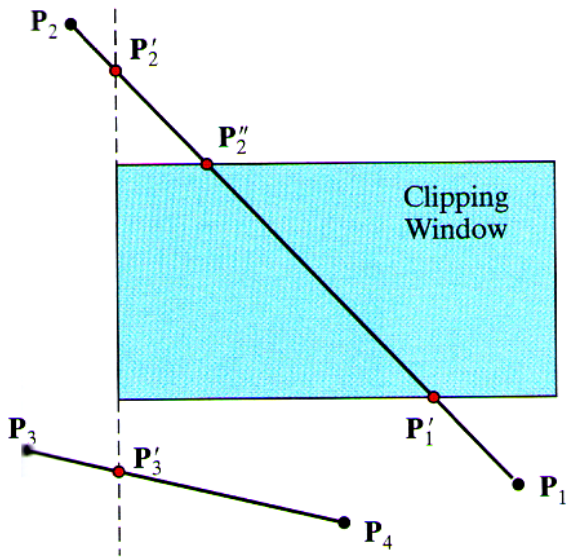
- Conforme cada intersecção com as fronteiras da janela de recorte são calculadas, a linha é recortada até restar apenas o que está dentro da janela, ou nenhuma parte esteja dentro da mesma

- 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

Recorte de Linha de Cohen-Sutherland

Processando a fronteira esquerda

- $\mathbf{P}_1 = 0100 \rightarrow$ está dentro da fronteira esquerda
- $\mathbf{P}_2 = 1001 \rightarrow$ está fora da fronteira esquerda
  - Calcula a intersecção $\mathbf{P}'_2$ e recorta a seção $\overline{\mathbf{P}_2\mathbf{P}'_2}$



- As outras fronteiras seguem o mesmo princípio

Recorte de Linha de Cohen-Sutherland

- Para se determinar as intersecções da reta definida pelos pontos (x_0, y_0) e (x_{end}, y_{end}) podemos usar a equação explicita

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

- O valor de x será xw_{min} ou xw_{max} e a inclinação será $m = (y_{end} - y_0)/(x_{end} - x_0)$

- Os valores de x da intersecção podem ser calculados usando

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

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

Recorte de Linha de Liang-Barsky

Algoritmo de Recorte de Liang-Barsky

- Um algoritmo mais rápido de recorte foi desenvolvido por Liang e Barsky (independentemente) envolvendo mais testes antes dos cálculos das intersecções
- Para uma reta com pontos finais (x_0, y_0) e (x_{end}, y_{end}), podemos descrever a reta na forma paramétrica

$$x = x_0 + u\Delta x$$

$$y = y_0 + u\Delta y$$

$$0 \leq u \leq 1$$

- Onde $\Delta x = x_{end} - x_0$ e $\Delta y = y_{end} - y_0$

Recorte de Linha de Liang-Barsky

- Combinando essas equações com as condições do recorte de pontos

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

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

- As seguintes desigualdades são obtidas

$$xw_{min} \leq x_0 + u\Delta x \leq xw_{max}$$

$$yw_{min} \leq y_0 + u\Delta y \leq yw_{max}$$

Recorte de Linha de Liang-Barsky

- Por sua vez essas desigualdades podem ser expressas por

$$u \cdot p_k \leq q_k, k = 1, 2, 3, 4$$

- Onde os parâmetros p e q são

$$p_1 = -\Delta x, q_1 = x_0 - xw_{min}$$

$$p_2 = \Delta x, q_2 = xw_{max} - x_0$$

$$p_3 = -\Delta y, q_3 = y_0 - yw_{min}$$

$$p_4 = \Delta y, q_2 = yw_{max} - y_0$$

Recorte de Linha de Liang-Barsky

Teste 1

- Qualquer linha paralela as arestas da janela tem $p_k = 0$, onde k corresponde a aresta $esquerda = 1$, $direita = 2$, $inferior = 3$ e $superior = 4$
  - Se além disso, $q_k < 0$, então a reta está completamente fora da janela
  - Se $q_k \geq 0$, a linha está dentro da borda paralela de recorte

Teste 2

- Se $p_k < 0$, a reta procede de fora para dentro desta fronteira particular da janela de recorte
- Se $p_k > 0$, a reta procede de dentro para fora

Recorte de Linha de Liang-Barsky

- Para valores $p_k \neq 0$, o valor de u que corresponde ao ponto da reta intersecta a fronteira k da janela pode ser calculado como

$$u = \frac{q_k}{p_k}$$

- Para cada linha os valores u_1 e u_2 que definem a parte da reta que está dentro da janela podem ser calculados
- u_1 é determinado levando em consideração as fronteiras das quais a reta procede de fora para dentro $(p < 0)$
  - Para essas calcula-se $r_k = q_k/p_k$, sendo u_1 o maior valor consistindo de 0 e vários valores de r
- u_2 é determinado examinando as fronteiras para qual a reta procede de dentro para fora $(p > 0)$
  - r_k é calculado para cada uma dessas e u_2 é considerado o menor valor consistindo de 1 e os valores r calculados

Recorte de Linha de Liang-Barsky

- Se $u_1 > u_2$, a linha está completamente fora da janela de recorte
- Caso contrário os valores das linhas recortadas são calculados para os dois valores do parâmetro u

Recorte de Linha de Liang-Barsky

```
1  GLint clip_test(GLfloat p, GLfloat q, GLfloat *u1, GLfloat *u2) {
2    GLfloat r;
3    GLint result = true;
4
5    if(p < 0) {
6      r = q / p;
7      if(r > *u2) result = false;
8      else if(r > *u1) *u1 = r;
9    }
10   else if(p > 0) {
11     r = q / p;
12     if(r < *u1) result = false;
13     else if(r < *u2) *u2 = r;
14   }
15   else { //p==0, reta paralela a fonteira de recorte
16     if(q < 0) { //reta fora da fronteira de recorte
17       result = false;
18     }
19   }
20
21   return result;
22 }
```

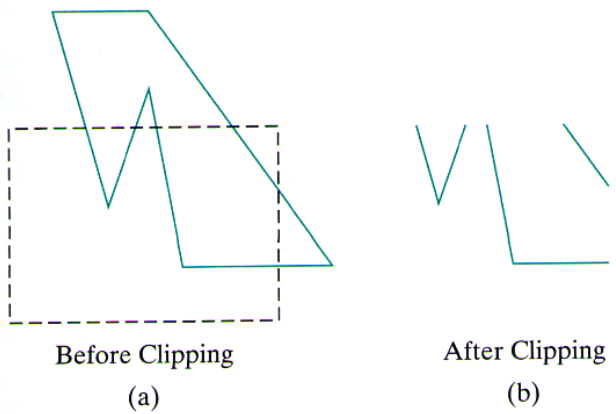
Recorte de Linha de Liang-Barsky

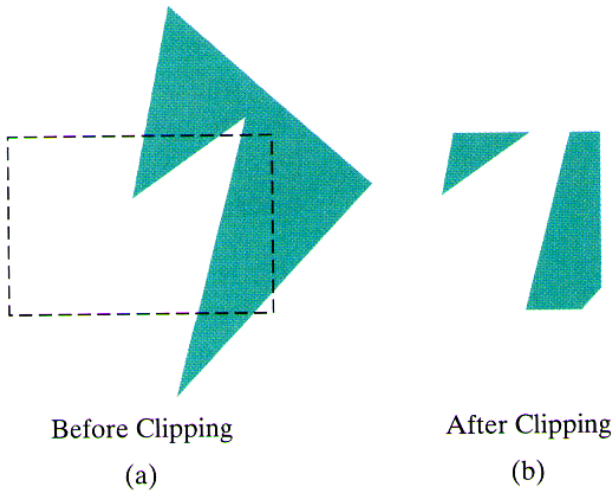
```
1  void clip(GLfloat xwmin, GLfloat xwmax, GLfloat ywmin, GLfloat ywmax,
2            GLfloat p1x, GLfloat p1y, GLfloat p2x, GLfloat p2y) {
3    GLfloat u1 = 0;
4    GLfloat u2 = 1;
5    GLfloat dx = p2x - p1x;
6    GLfloat dy = p2y - p1y;
7
8    if(clip_test(-dx, p1x - xwmin, &u1, &u2))
9      if(clip_test(dx, xwmax - p1x, &u1, &u2)) {
10       if(clip_test(-dy, p1y - ywmin, &u1, &u2))
11         if(clip_test(dy, ywmax - p1y, &u1, &u2)) {
12           if(u2 < 1) {
13             p2x = p1x + u2*dx;
14             p2y = p1y + u2*dy
15           }
16
17           if(u1 > 0) {
18             p1x = p1x + u1*dx;
19             p1y = p1y + u1*dy
20           }
21
22           //desenha a linha entre (p1x, p1y) e (p2x, p2y)
23           //..
24         }
25     }
26 }
```

Sumário

Recorte de Polígonos 2D

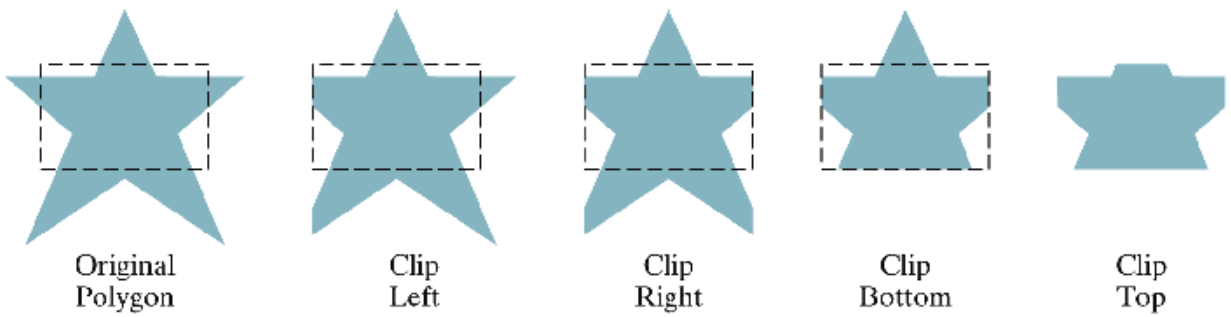
- Para fazer o corte de polígonos, os algoritmo de recorte de linhas não podem ser aplicados porque em geral esses não produziriam polígonos fechados
  - Produziriam linhas desconexas sem informação de como uni-las para formar o polígono recortado





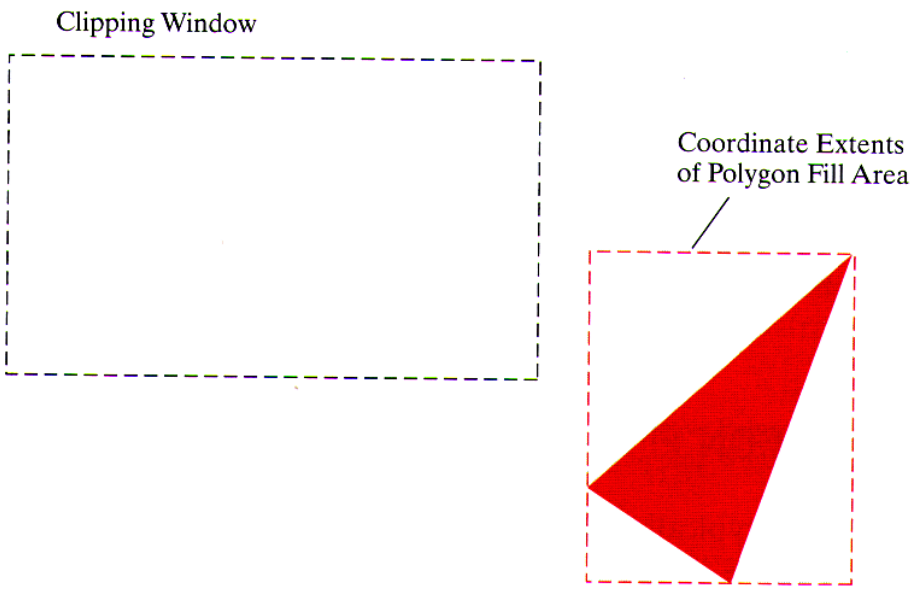
Recorte de Polígonos 2D

- 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



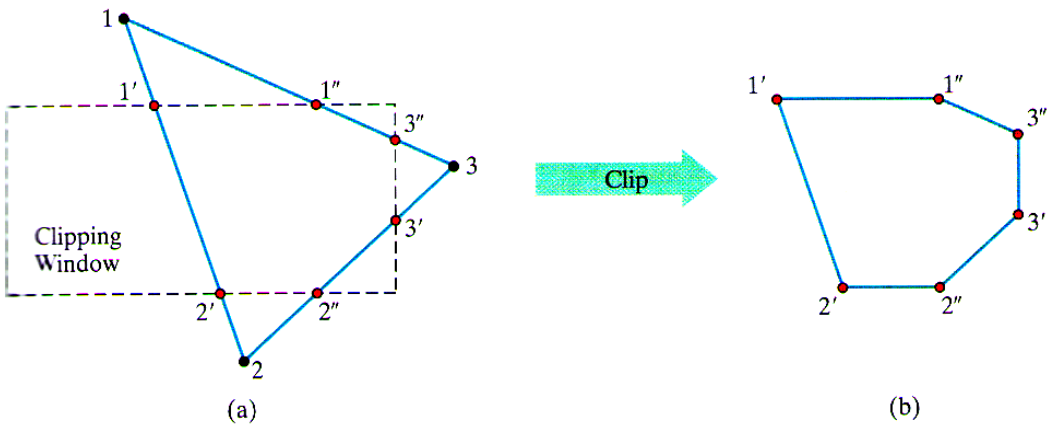
Recorte de Polígonos 2D

- 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 intersecções são calculadas



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



Recorte de Polígonos de Sutherland-Hodgman

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 possa ser passado imediatamente para o próximo estágio
  - Elimina a necessidade de uma lista de novos vértices para cada estágio de recorte
  - Permite implementação paralela do recorte

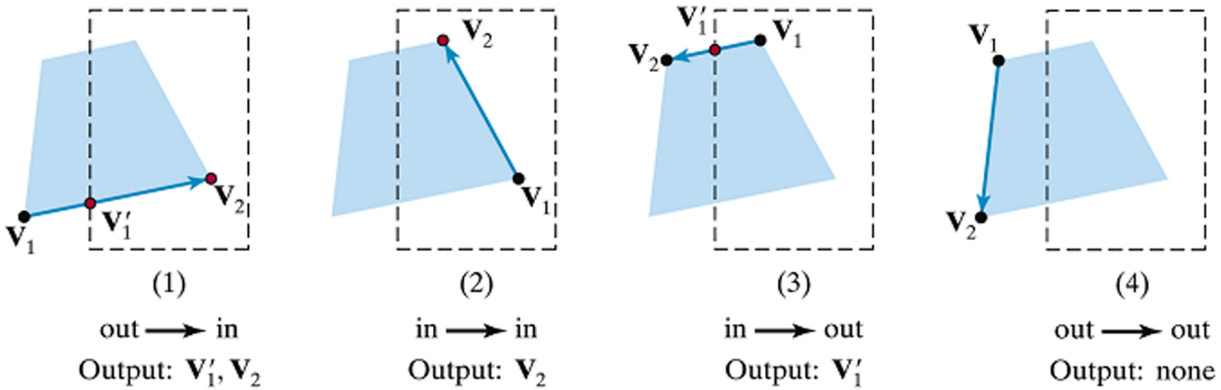
Recorte de Polígonos de Sutherland-Hodgman

- 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
  - Então o próximo par de vértices é mandado para o primeiro recortador

- 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 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 pontos finais estão fora da janela de recorte

Recorte de Polígonos de Sutherland-Hodgman

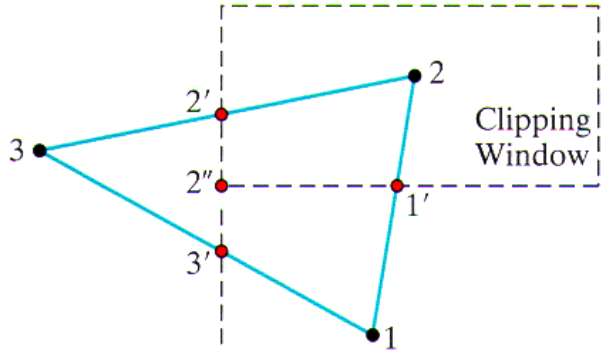
- Para facilitar a passagem dos vértices de um recortador para outro, a saída de cada recortador pode ser da seguinte forma

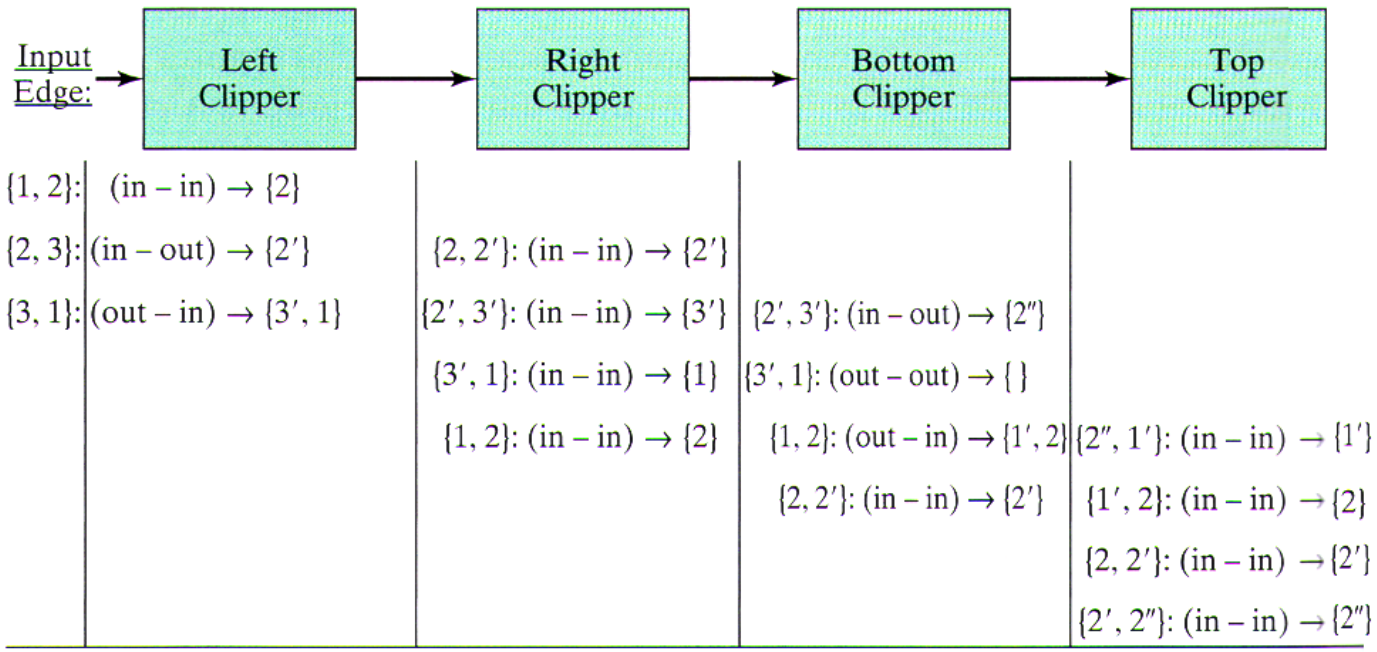


Recorte de Polígonos de Sutherland-Hodgman

- 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 intersecção obtida e o segundo vértice
  2. Se ambos 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 intersecção
  4. Se ambos vértices estão fora, nada é enviado

Recorte de Polígonos de Sutherland-Hodgman

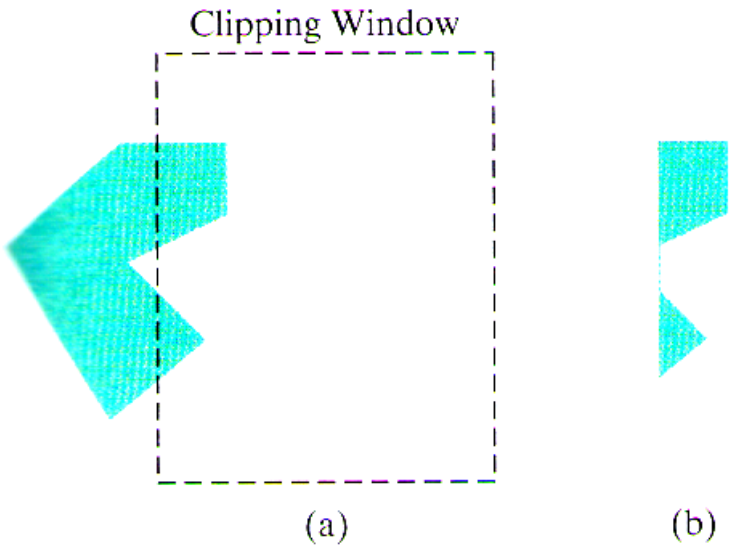




Recorte de Polígonos de Sutherland-Hodgman

Limitação

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

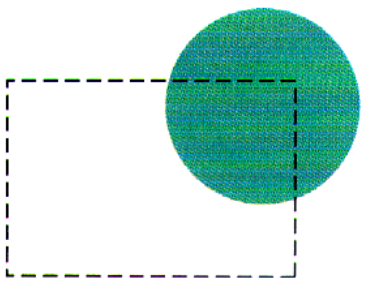


- Uma solução seria dividir o polígono côncavo em partes convexas

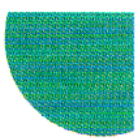
Sumário

Recorte de Curvas

- Áreas curvas podem ser recortadas usando abordagens parecidas com as apresentadas
  - Se as curvas forem aproximações poligonais, então o recorte é o mesmo apresentado anteriormente
  - Caso contrário o procedimento de recorte irá envolver equações não-lineares



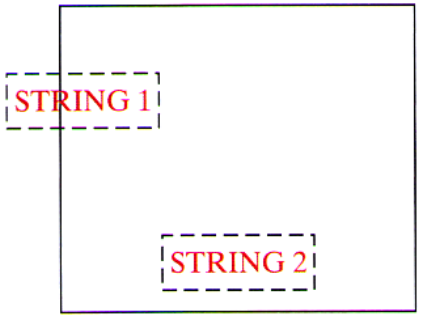
Before Clipping



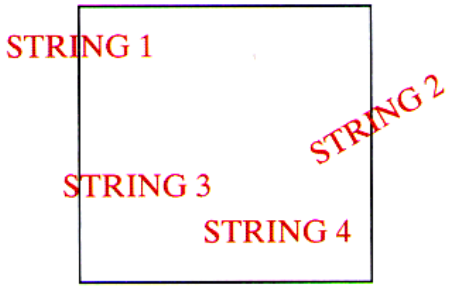
After Clipping

Recorte de Texto

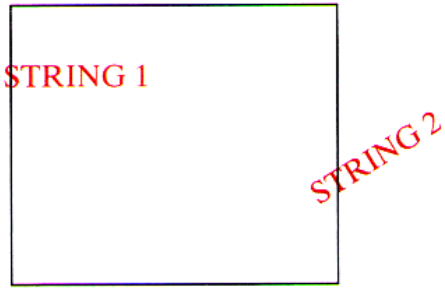
- Existem diversas formar para se fazer o recorte de texto, a escolha dependendo do pacote gráfico utilizado e como as letras são geradas



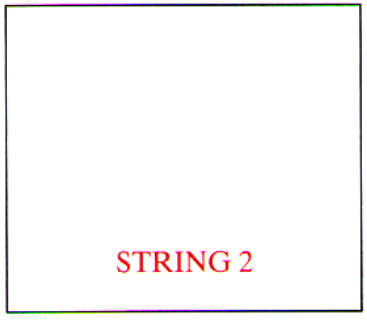
Before Clipping



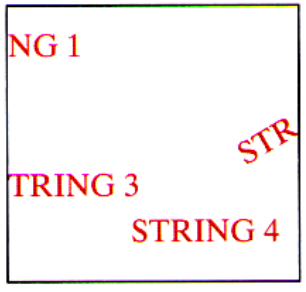
Before Clipping



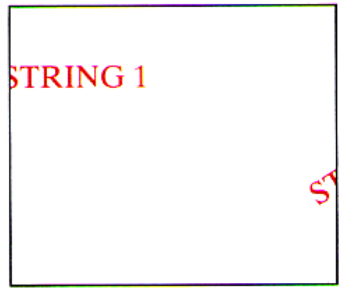
Before Clipping



After Clipping



After Clipping



After Clipping