UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Viewing 2D

Disciplina: Computação Gráfica (BCC35F)
Curso: Ciência da Computação

Prof. Walter T. Nakamura
waltertakashi@utfpr.edu.br

Campo Mourão - PR

Baseados nos materiais elaborados pelas professoras Aretha Alencar (UTFPR) e Rosane Minghim (USP)

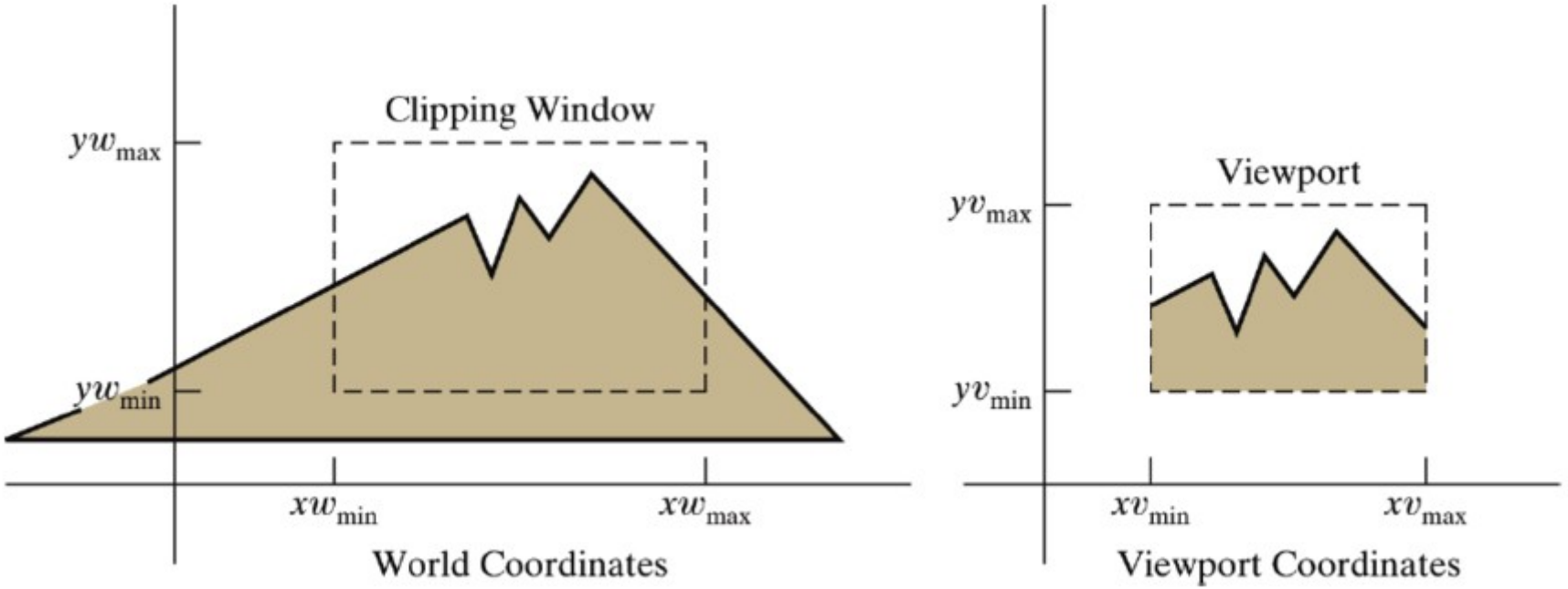
Introdução

- **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

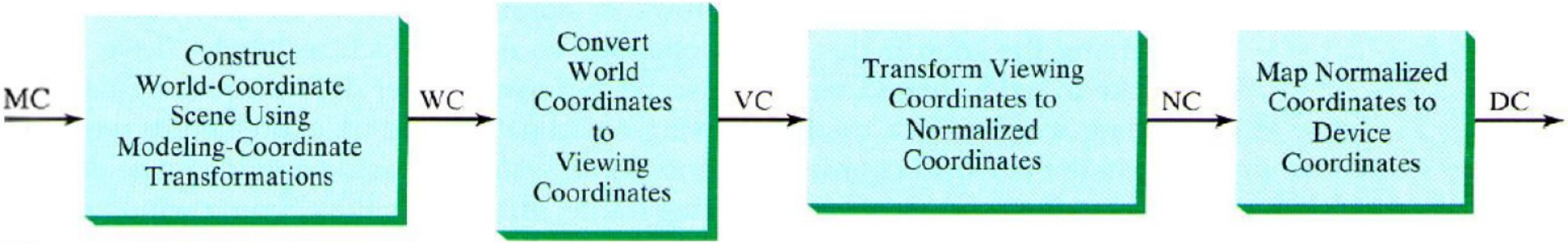
- **Janela de Recorte (Clipping Window)**
  - Uma **seção** de uma cena 2D que é selecionada para ser mostrada
    - Tudo o que estiver fora dessa seção será “cortado fora”
- **Viewport**
  - A Janela de Recorte pode ser posicionada dentro de uma janela do sistema usando outra janela chamada de **Viewport**
    - 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



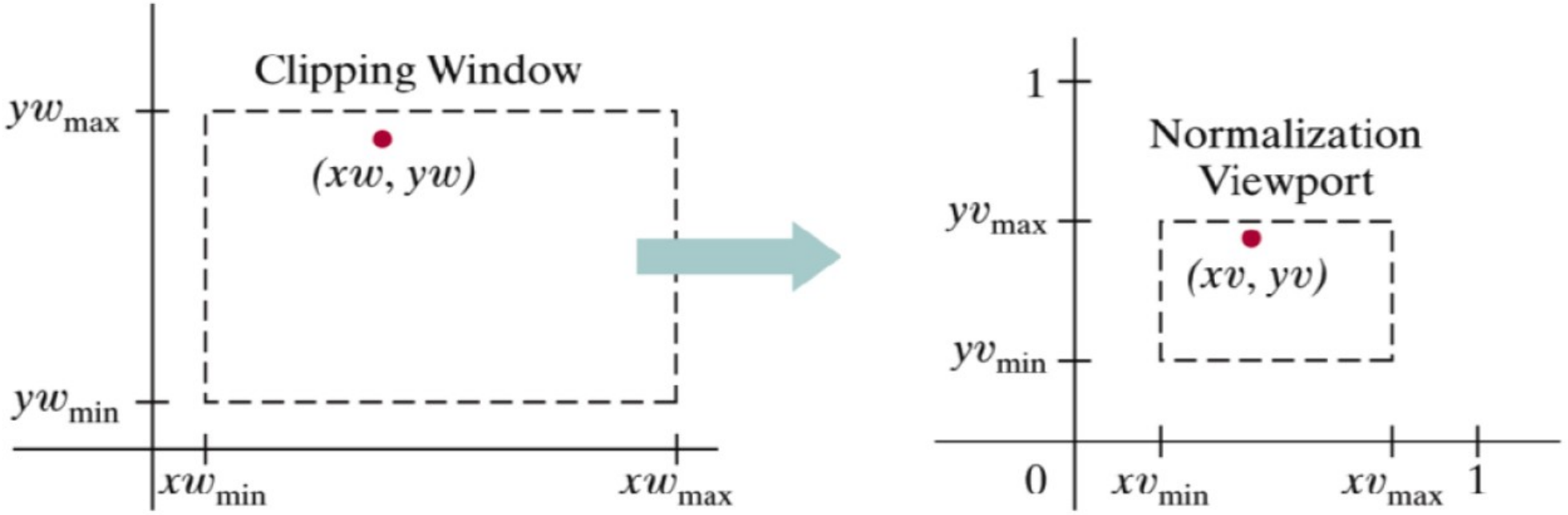
  - Para tornar o processo **independente** dos requisitos de diferentes dispositivos de saída, sistemas gráficos convertem a descrição dos objetos para coordenadas normalizadas (entre 0 e 1 ou entre -1 e 1).

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
- No OpenGL, a **janela de recorte** é especificada no sistema de coordenadas do mundo

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:

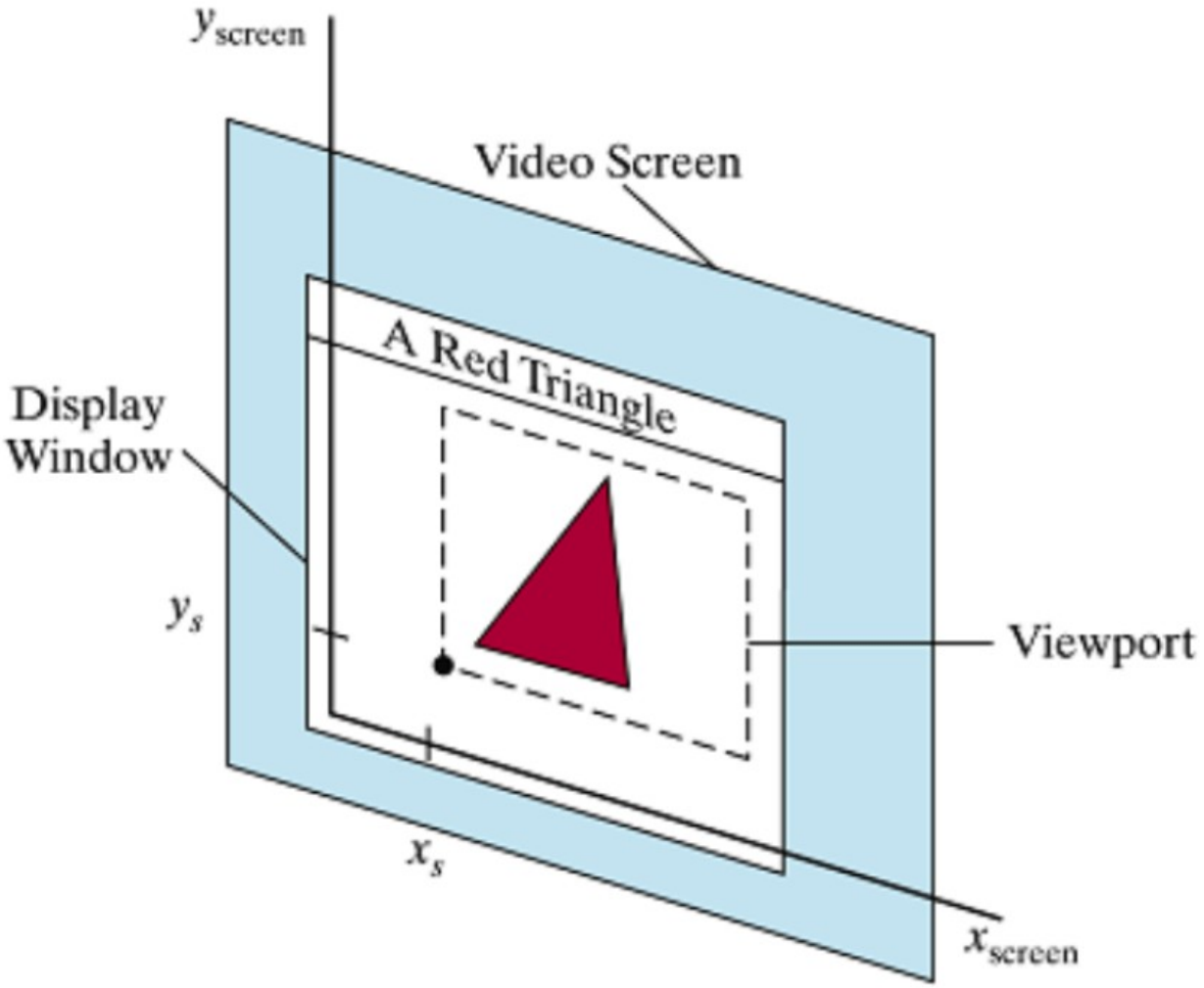
$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}}$$

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

- 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

Mapeando a Janela de Recorte em uma Viewport Normalizada

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



Programação OpenGL: 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:

```
glMatrixMode(GL_PROJECTION);
```

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

```
glLoadIdentity();
```

Programação OpenGL: Definindo a Janela de Recorte

- A Janela de Recorte é definida por:

```
gluOrtho2D(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.

Programação OpenGL: Definindo a Viewport

- A viewport é definida e posicionada por:

```
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 em pixels.
- Se não invocarmos a função `glViewport(...)`, a viewport irá ocupar todo o tamanho disponível na janela de exibição

Programação OpenGL: Múltiplas Viewports

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

```
#include <GL/glut.h>

class wcPt2D {
    public: GLfloat x, y;
};

void init (void) {
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0, 300.0, 0, 300.0);
    glMatrixMode(GL_MODELVIEW);
}

void triangle(wcPt2D *verts) {
    GLint k;
    glBegin(GL_TRIANGLES);
        for (k = 0; k < 3; k++)
            glVertex2f(verts[k].x, verts[k].y);
    glEnd();
}

void square(wcPt2D *verts){
    GLint k;

    glBegin(GL_LINE_LOOP);
        for (k = 0; k < 4; k++)
            glVertex2f(verts[k].x, verts[k].y);
    glEnd();
}
```

Programação OpenGL: Múltiplas Viewports

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

```
void display(void) {
    wcPt2D vertsTriangle[3] = { {75, 75}, {225, 75}, {150, 225}};
    wcPt2D vertsQuadr[4] = { {50, 50}, {250, 50}, {250, 250}, {50, 250}};

    glClear(GL_COLOR_BUFFER_BIT);   // Limpa a tela

    glLoadIdentity();               // Carrega a matriz identidade
    glColor3f(0.0, 0.0, 1.0);       // Define a cor de preenchimento como azul
    glViewport(0, 125, 250, 250);   // Define a viewport da esquerda
    triangle(vertsTriangle);        // Exibe o triângulo
    square(vertsQuadr);             // Exibe o quadrado

    /* Rotaciona o triângulo e mostra na viewport da direita */
    glColor3f(1.0, 0.0, 0.0);       // Define a cor de preenchimento como vermelho
    glViewport(250, 200, 100, 100); // Define o viewport da direita
    glTranslatef(150, 150, 0);      // Posiciona na posição original
    glRotatef(90.0, 0.0, 0.0, 1.0); // Rotaciona os objetos em 90 graus
    glTranslatef(-150, -150, 0);    // Posiciona os objetos na origem
    triangle(vertsTriangle);        // Exibe o triângulo vermelho rotacionado
    square(vertsQuadr);             // Exibe o quadrado rotacionado

    glFlush();
}
```

Programação OpenGL: Múltiplas Viewports

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

```
int main (int argc, char ** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(200, 0);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Exemplo de divisao de tela");

    init();
    glutDisplayFunc(display);

    glutMainLoop();
}
```

Algoritmos de Recorte

- 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

Recorte de Ponto 2D

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

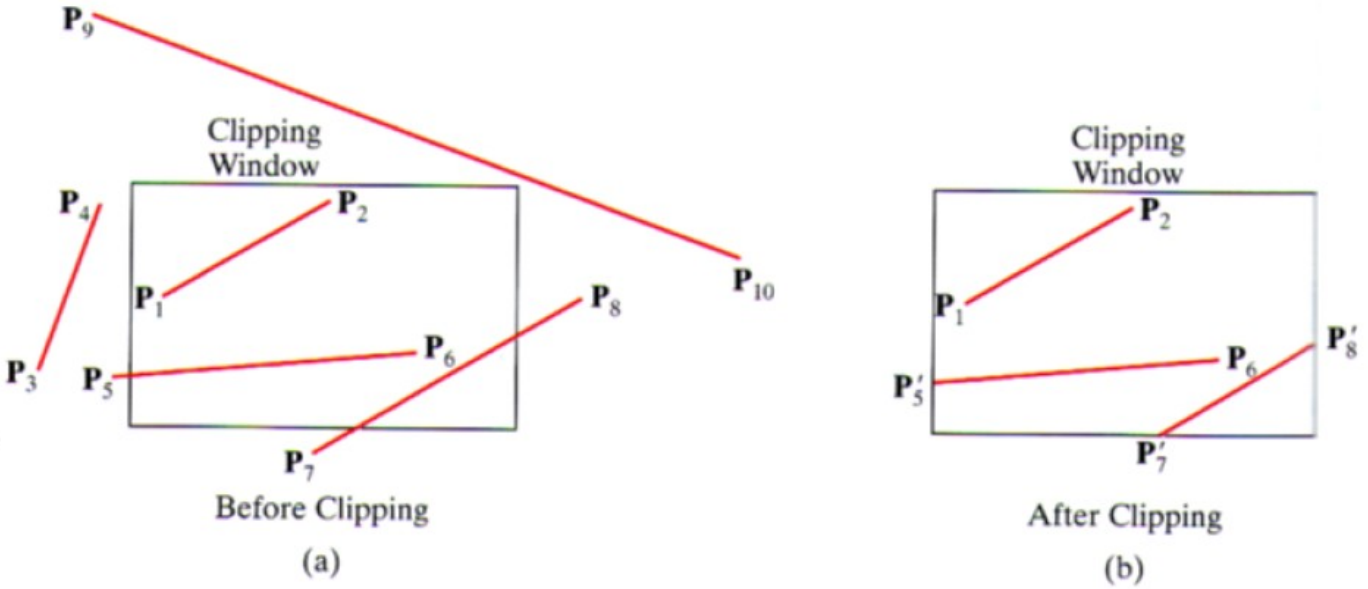
$$xw_{min} \leqslant x \leqslant xw_{max}$$

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

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

Recorte de Linha 2D

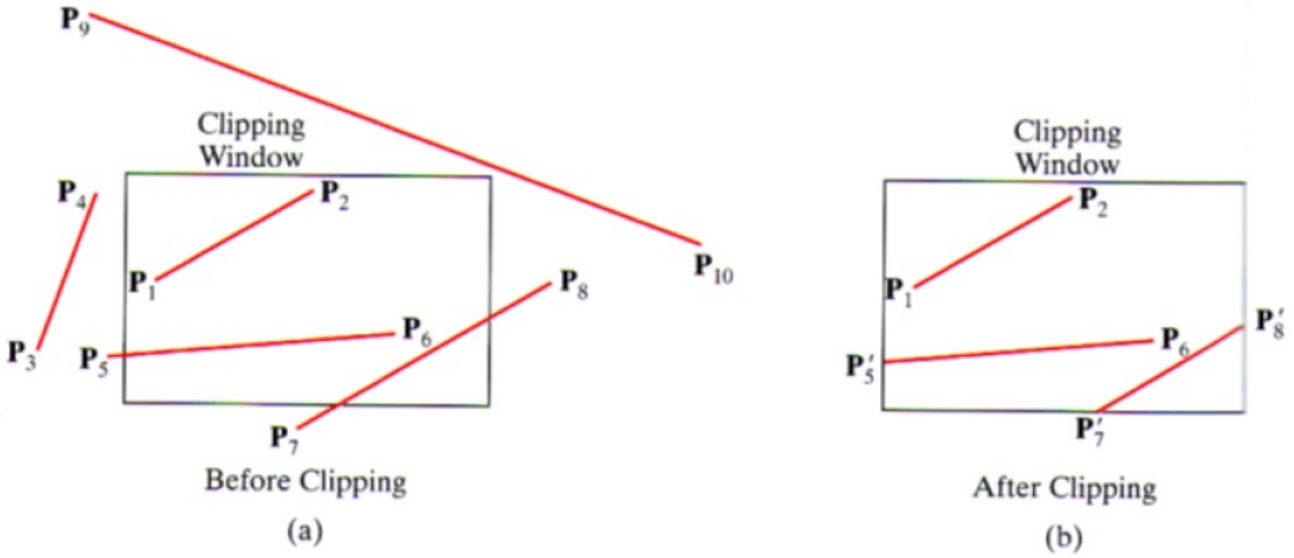
- Processa **cada linha** em uma cena por meio de uma série de testes e cálculos de intersecção para determinar se uma linha ou parte dela precisa ser desenhada



- 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

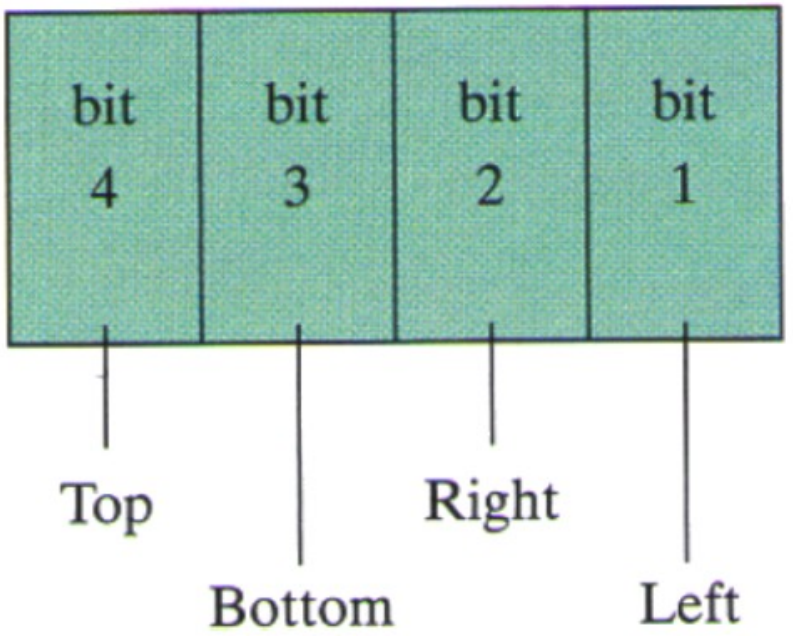
Recorte de Linha 2D

- 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 $\overline{P_1 P_2}$), 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



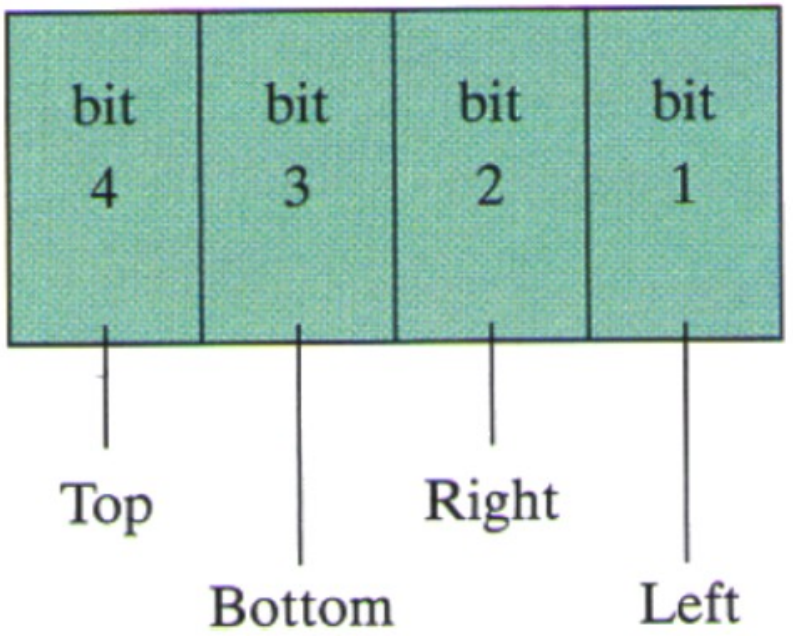
Recorte de Linha de Cohen-Sutherland

- **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



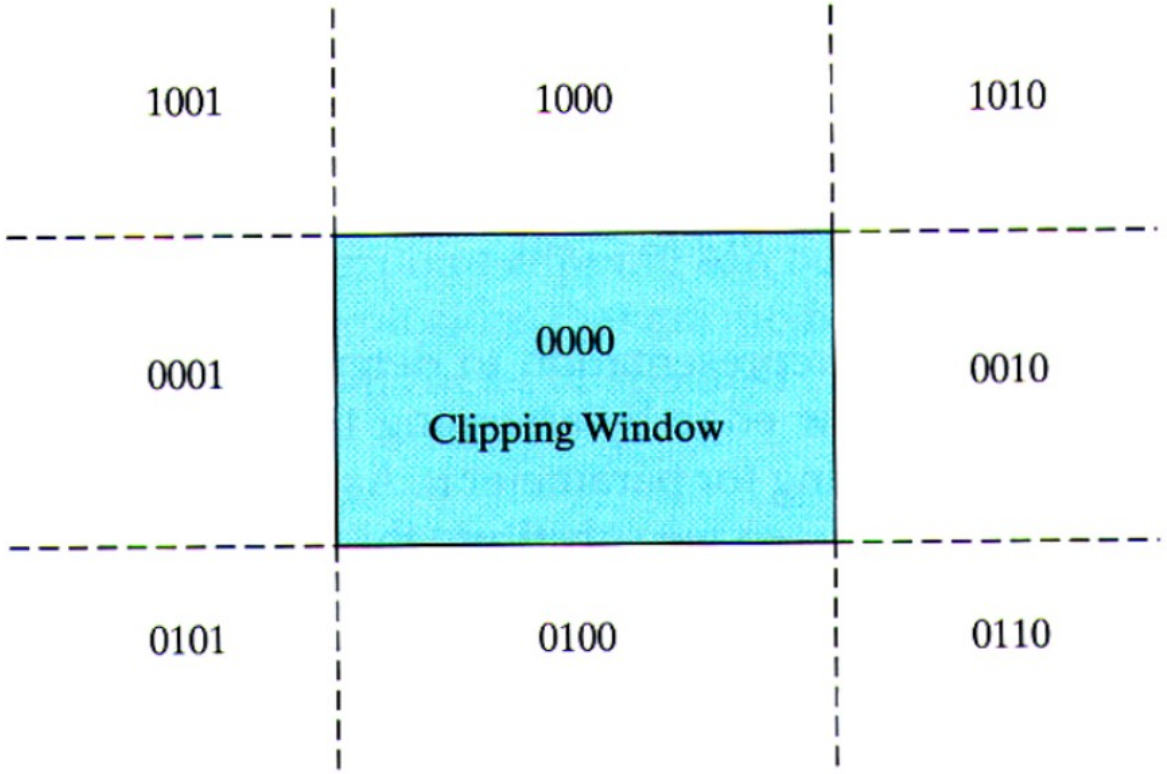
Recorte de Linha de Cohen-Sutherland

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



Recorte de Linha de Cohen-Sutherland

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



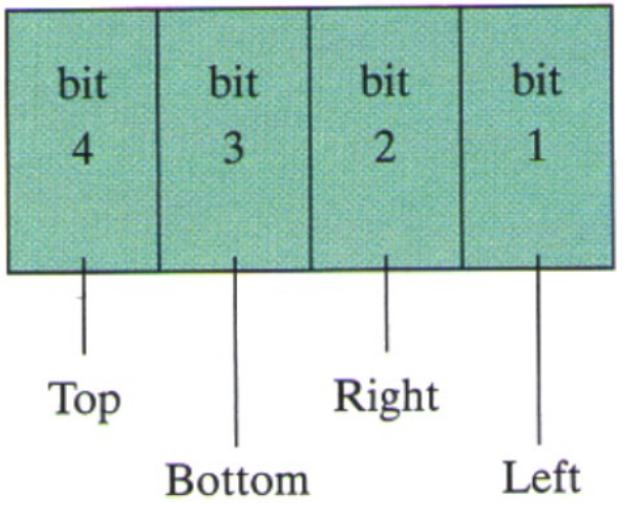
- Um ponto abaixo e a esquerda a janela de recorte recebe valor 0101, um ponto dentro 0000

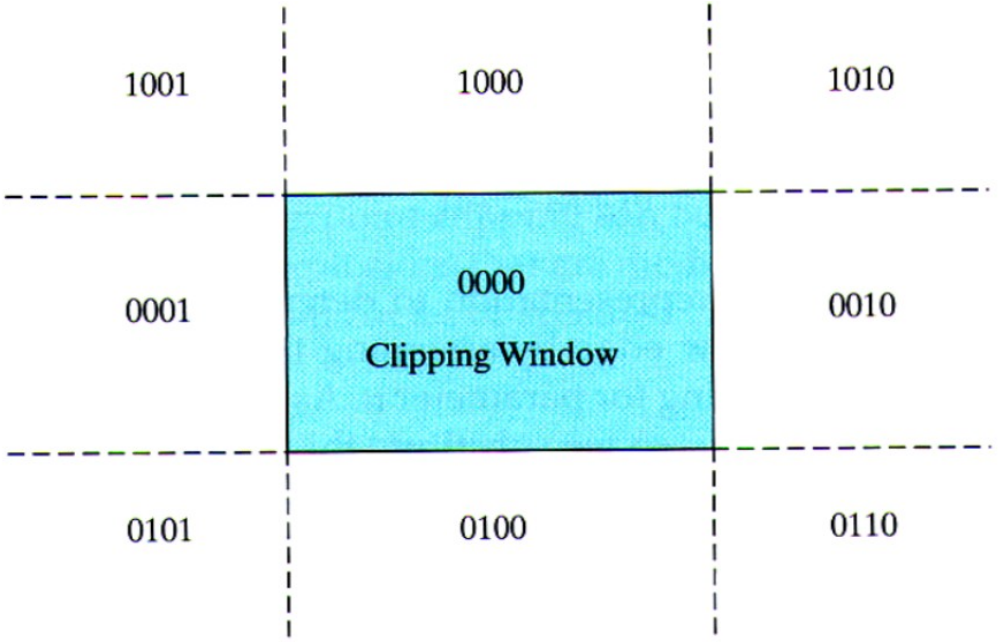
Recorte de Linha de Cohen-Sutherland

- Os valores dos bits são determinados comparando as coordenadas (x, y) do ponto com as fronteiras de recorte:
  - O bit 4 é definido como 1 se $y > yw_{max}$ (fronteira do topo (T))
  - O bit 3 é definido como 1 se $y < yw_{min}$ (fronteira inferior (B))
  - O bit 2 é definido como 1 se $x > xw_{max}$ (fronteira direita (R))
  - O bit 1 é definido como 1 se $x < xw_{min}$ (fronteira esquerda (L))

Recorte de Linha de Cohen-Sutherland

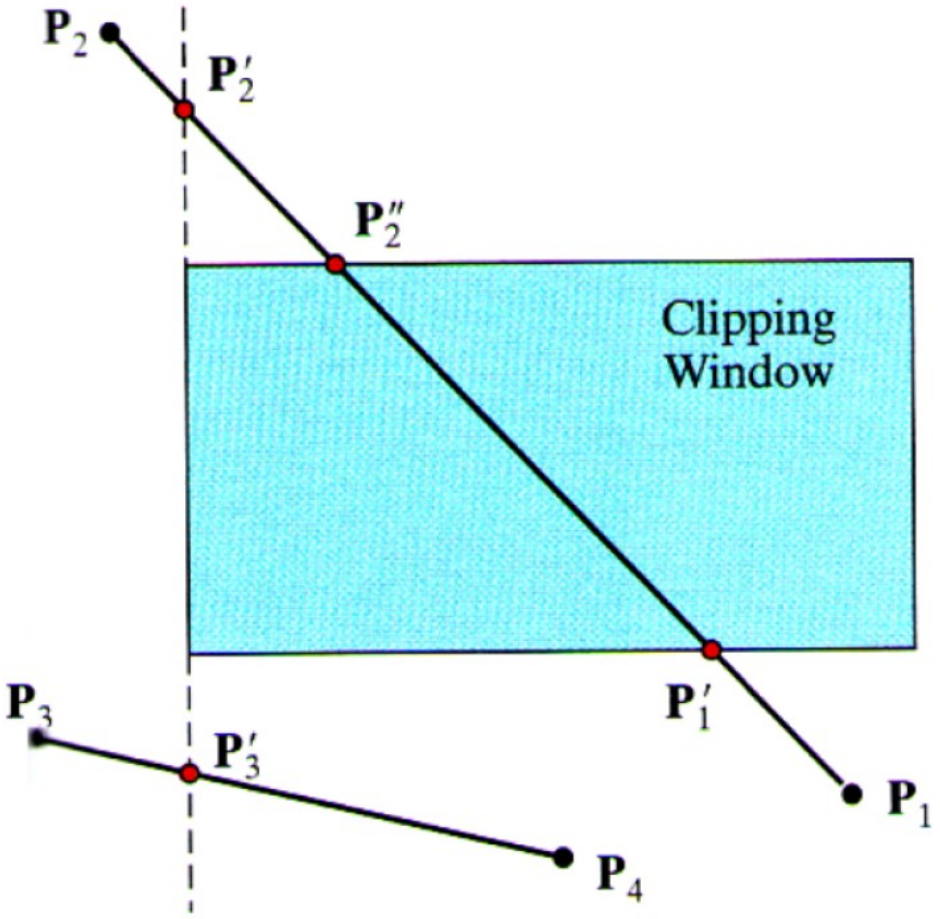
- Com base nesses códigos é possível determinar rapidamente se uma linha está **completamente fora** ou **dentro** da janela:
  - Linhas completamente dentro tem seus pontos definidos como 0000
  - Linhas que tenham **1 nas mesmas posições** dos pontos finais está completamente fora da janela de recorte
    - Uma linha com pontos finais identificados por 1001 e 0101 está completamente a esquerda da janela de recorte.





Recorte de Linha de Cohen-Sutherland

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



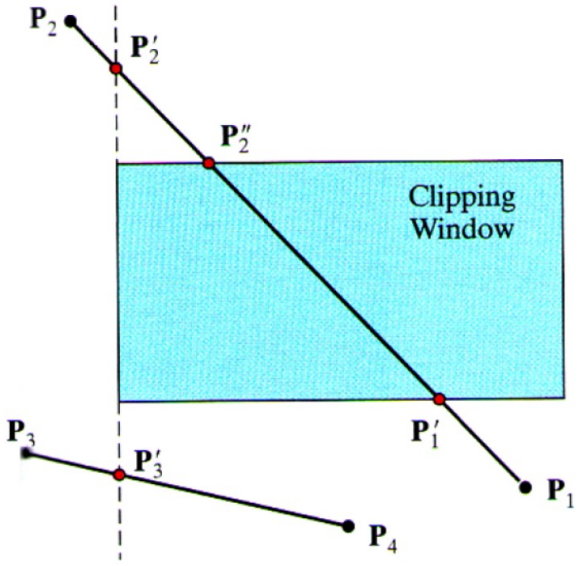
Recorte de Linha de Cohen-Sutherland

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

- 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 primeiro:
  - $P_1 = 0100 \rightarrow$ está dentro da fronteira da esquerda
  - $P_2 = 1001 \rightarrow$ está fora da fronteira da esquerda
    - Calcula a intersecção P_2' e recorta a seção P_2P_2'



  - As demais fronteiras (direita, embaixo e topo) seguem o mesmo princípio.

Recorte de Linha de Cohen-Sutherland

- Para se determinar os valores de y para as intersecções da reta definida pelos pontos (x_0, y_0) e (x_{end}, y_{end}) nas fronteiras verticais (esquerda e direita) podemos usar a equação explícita:

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

- O valor de x será xw_{min} (fronteira esquerda) ou xw_{max} (fronteira direita) e a inclinação será:

$$m = (y_{end} - y_0)/(x_{end} - x_0)$$

Recorte de Linha de Cohen-Sutherland

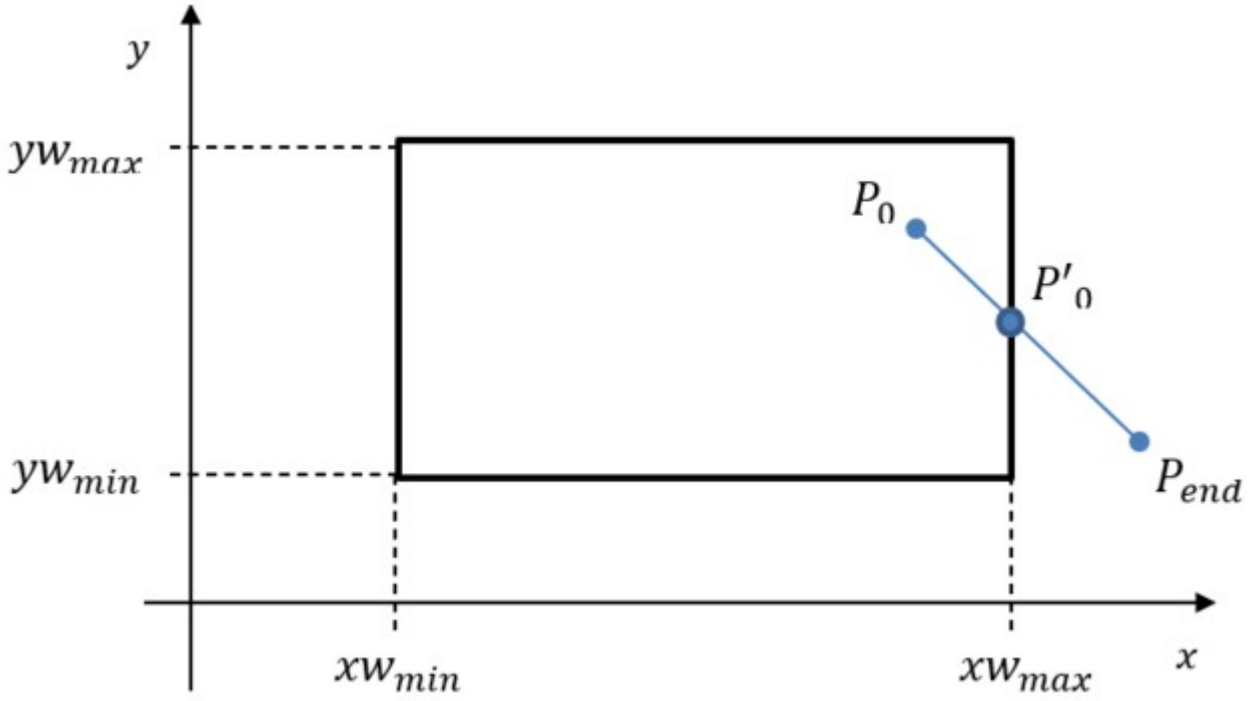
- Já para se determinar os valores de x para as intersecções da reta definida pelos pontos (x_0, y_0) e (x_{end}, y_{end}) nas fronteiras horizontais (topo e inferior) podemos usar a equação explícita:

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

- O valor de y será yw_{min} (fronteira inferior) ou yw_{max} (fronteira do topo)

Recorte de Linha de Cohen-Sutherland

- Dada uma janela de recorte com as seguintes dimensões: $xw_{min} = 100$, $xw_{max} = 300$, $yw_{min} = 150$ e $yw_{max} = 250$; e uma linha formada pelos pontos: $P_0 = (x_0, y_0) = (280, 225)$ e $P_{end} = (x_{end}, y_{end}) = (350, 160)$
- Aplique o algoritmo de Cohen-Sutherland para recorte de linhas 2D.



Recorte de Linha de Cohen-Sutherland

- Códigos Binários: 0 ⇒ Dentro; 1 ⇒ Fora
- $P_0 \Rightarrow (TBRL) = (0000)$ e $P_{end} \Rightarrow (TBRL) = (0010)$
- Observando os códigos binários:
  - A linha não está completamente dentro, pois para isso teríamos que ter ambos os códigos iguais a 0000
  - Não temos 1 nas mesmas posições dos pontos finais, então a linha não está completamente fora
  - A linha cruza a fronteira da direita (right), pois o bit da direita em P_0 é igual a 0 e o bit da direita em P_{end} é igual a 1.

Recorte de Linha de Cohen-Sutherland

- Cálculo das coordenas de P_0':
  - Sabemos a coordenada x de P_0', pois ela é igual a $xw_{max} = 300$

$$P_0' = (x, y) = (300, y)$$

  - Para a coordenada y de P_0', temos que:

$$m = \frac{y_{end} - y_0}{x_{end} - x_0} = \frac{160 - 225}{350 - 280} = \frac{-65}{70} \approx -0{,}9285$$

$$y = y_0 + m(x - x_0) = 225 + (-0{,}9285) \times (300 - 280) = 206{,}43$$

  - Assim temos que:

$$P_0' = (x, y) = (300; 206{,}43)$$

Algoritmo de Cohen-Sutherland em OpenGL

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

```
#include <GL/glut.h>

// Define os valores do quadrado e da linha
GLfloat xMin = 100, xMax = 300, yMin = 100, yMax = 300;
GLfloat p1x = 320, p1y = 350, p2x = 50, p2y = 150;
void square();
void line();
void processCodes(int c1, int c2);
int getCode(GLfloat x, GLfloat y);
void clip();

// Valores em binários referentes a 1000, 0100, 0010 e 0001
int top = 8, bottom = 4, right = 2, left = 1;
int c1, c2;

void init (void) {
    /* Define a cor da janela de exibição como branco */
    glClearColor(1.0, 1.0, 1.0, 0.0);

    /* Define os parâmetros para a janela de recorte da coordenada do mundo */
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0, 400.0, 0, 400);

    /* Define o moto para construir matrizes de transformação geométrica */
    glMatrixMode(GL_MODELVIEW);
}

void display(void) {
    glClear(GL_COLOR_BUFFER_BIT); // Limpa a tela
    glColor3f(0.0, 0.0, 1.0);     // Define a cor de preenchimento como azul
    square();   // Desenha o quadrado
    line();     // Desenha a linha
    c1 = getCode(p1x, p1y); // Obtém o código do ponto 1
    c2 = getCode(p2x, p2y); // Obtém o código do ponto 2
    processCodes(c1, c2);   // Processa os códigos dos 2 pontos
    glutPostRedisplay();
    glFlush();
}
```

Algoritmo de Cohen-Sutherland em OpenGL

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

```
int main (int argc, char ** argv) {
glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(50, 50);
    glutInitWindowSize(600, 600);
    glutCreateWindow("Exemplo de divisao de tela");
    init();
    glutDisplayFunc(display);
    glutMainLoop();
}

void square() {
    glBegin(GL_LINE_LOOP);
        glVertex2f(xMin, yMax);
        glVertex2f(xMax, yMax);
        glVertex2f(xMax, yMin);
        glVertex2f(xMin, yMin);
    glEnd();
}

void line(){
    glColor3f(1.0, 0, 0);
    glBegin(GL_LINES);
        glVertex2f(p1x, p1y);
        glVertex2f(p2x, p2y);
    glEnd();
}

int getCode(GLfloat x, GLfloat y){
    int code = 0;
    if (x < xMin)
        code = code | left;
    if (x > xMax)
        code = code | right;
    if (y < yMin)
        code = code | bottom;
    if (y > yMax)
        code = code | top;
    return code;
}
```

Algoritmo de Cohen-Sutherland em OpenGL

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

```
void processCodes(int c1, int c2){
    // Verifica se os códigos se encaixam nas duas condições:

    // Verifica se tem algum ponto que está fora da fronteira
    // Se o resultado for zero, os pontos estão completamente dentro
    // c1 | c2 == 0 --> Pontos completamente dentro

    // Verifica se tem algum ponto com 1 no mesmo bit
    // Se tiver, os pontos então estão completamente fora
    // c1 & c2 != 0 --> Pontos completamente fora
    if ( ((c1|c2) == 0) || ((c1&c2) != 0) ){

    }else{
        clip();
    }
}
```

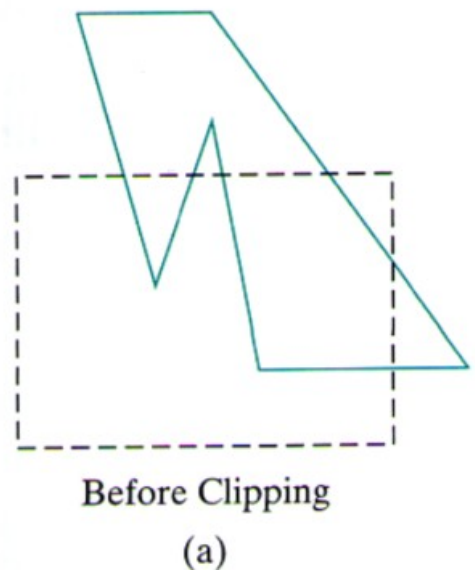
Algoritmo de Cohen-Sutherland em OpenGL

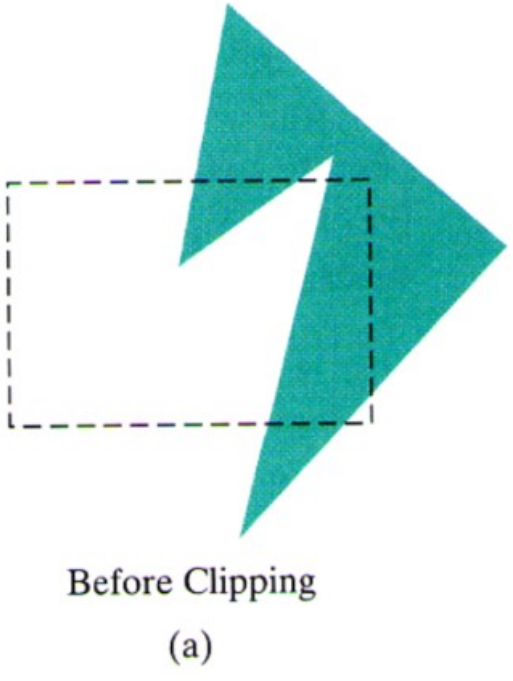
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

```
void clip(){
    int p_linha;
    GLfloat x, y, m;

    // Identifica o ponto que está do lado de fora da área
    if (c1 != 0)
        p_linha = c1;
    else
        p_linha = c2;

    // Calcula as intersecções
    m = (p2y - p1y)/(p2x - p1x);
    if (p_linha & left){
        x = xMin;
        y = p1y + m * (x - p1x);
    }
    if (p_linha & right){
        x = xMax;
        y = p1y + m * (x - p1x);
    }
    if (p_linha & bottom){
        y = yMin;
        x = p1x + ((y - p1y)/m);
    }
    if (p_linha & top){
        y = yMax;
        x = p1x + ((y - p1y)/m);
    }

    // Atualiza os valores dos pontos p1 e p2
    if (p_linha == c1){   //
        p1x = x;
        p1y = y;
    } else {
        p2x = x;
        p2y = y;
    }
}
```

Recorte de Polígonos 2D

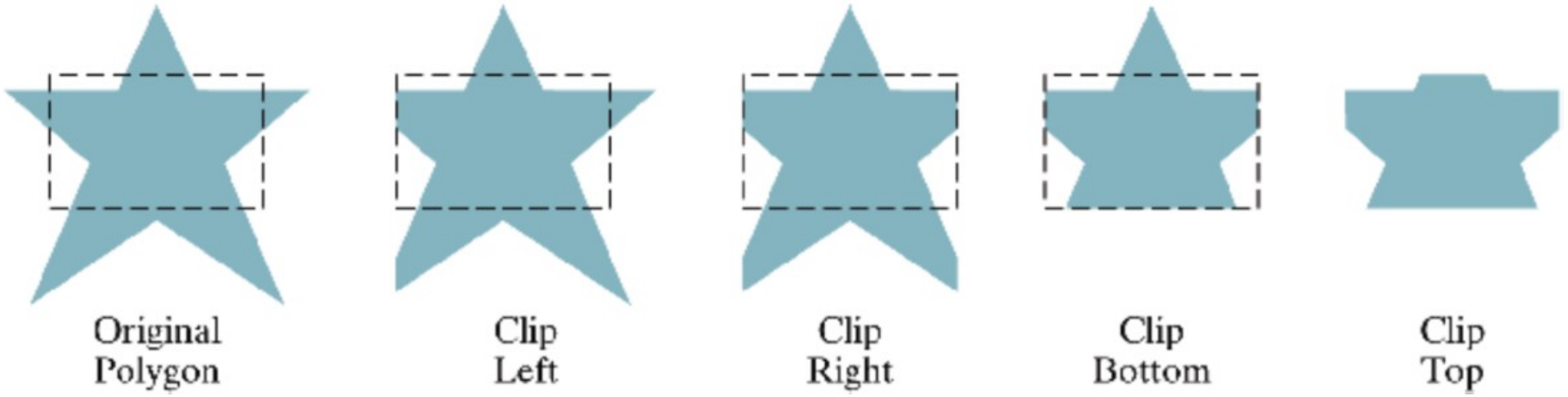
- Para fazer o corte de polígonos, os **algoritmos de recorte de linhas** não podem ser aplicados porque em geral esses não produziriam polígonos fechados:
  - Produziriam linhas desconexas sem informação de como uni-las para formar o polígono recortado





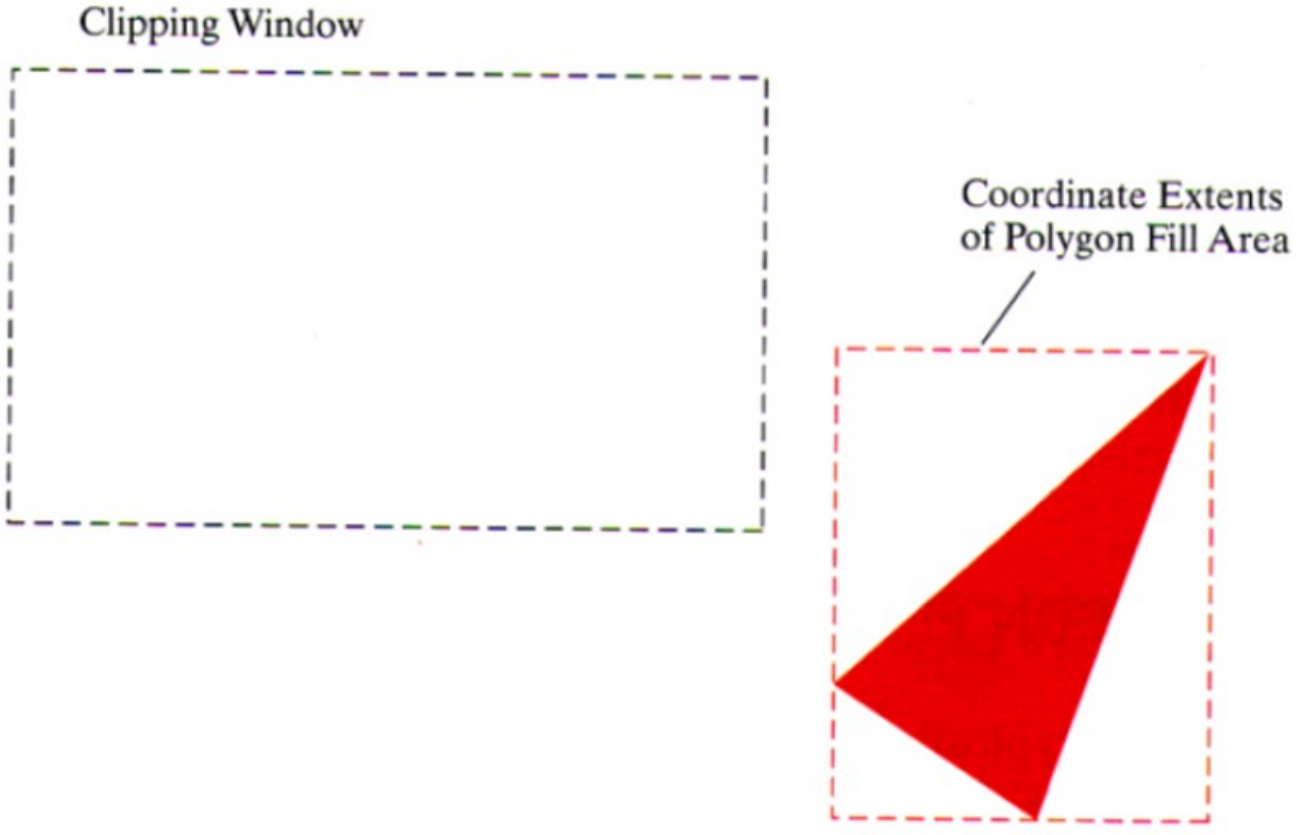
Recorte de Polígonos 2D

- 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.



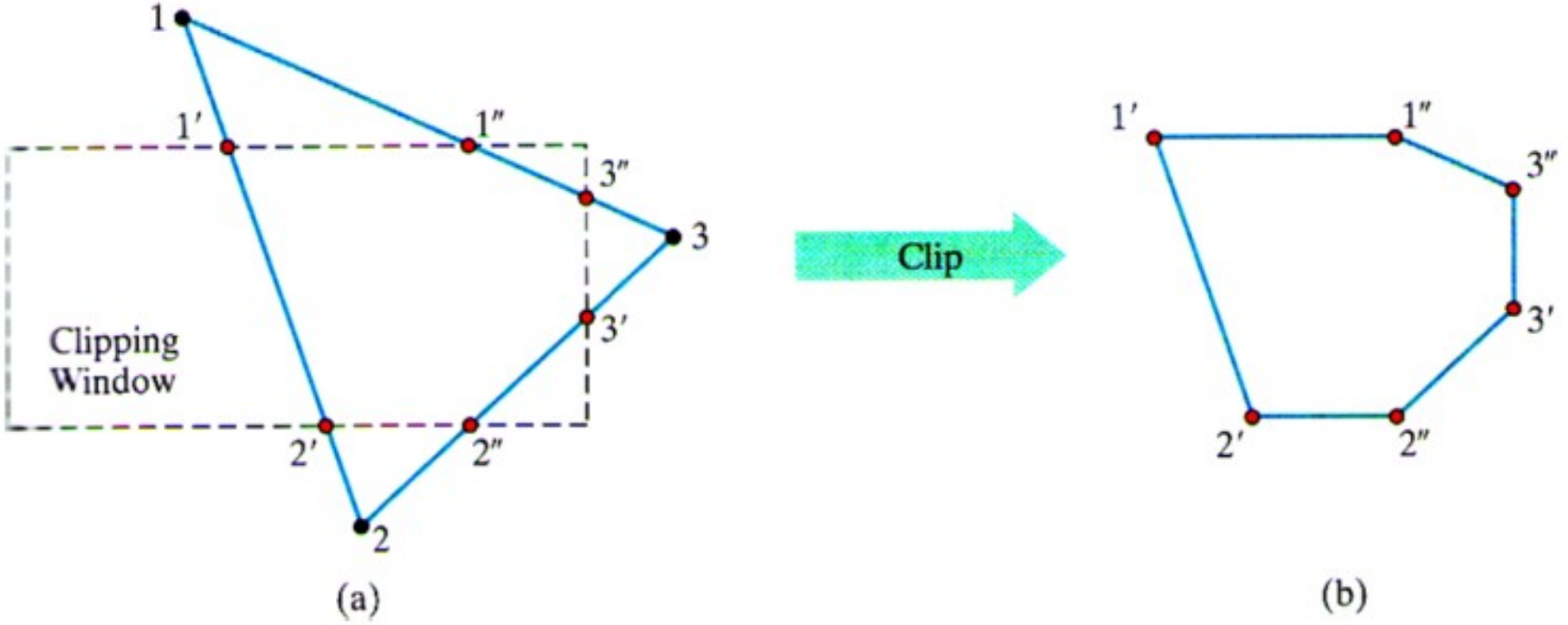
Recorte de Polígonos 2D

- 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

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

- **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
  - 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

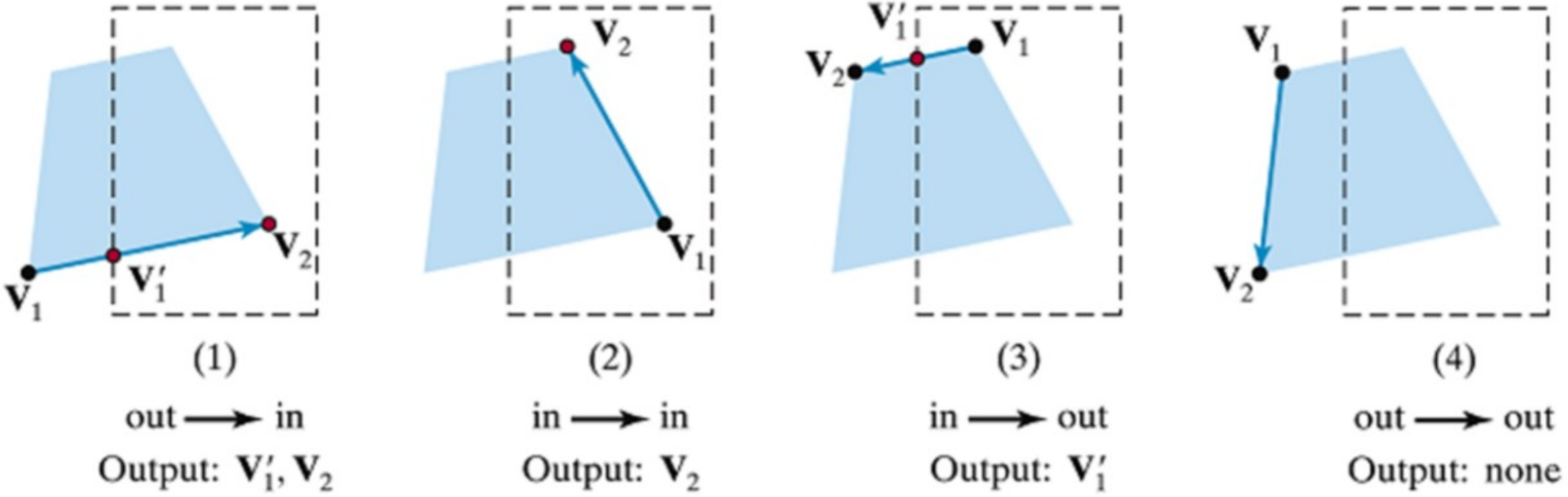
Recorte de Polígonos de Sutherland-Hodgman

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

- As arestas do polígonos precisam ser processadas uma a uma em **ordem anti-horária** para cada uma das fronteiras
- 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

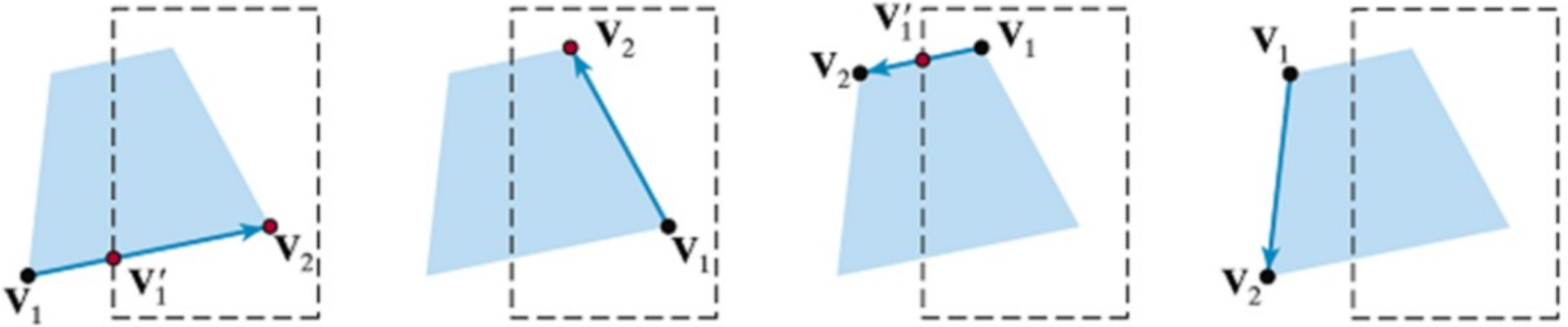
Recorte de Polígonos de Sutherland-Hodgman

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

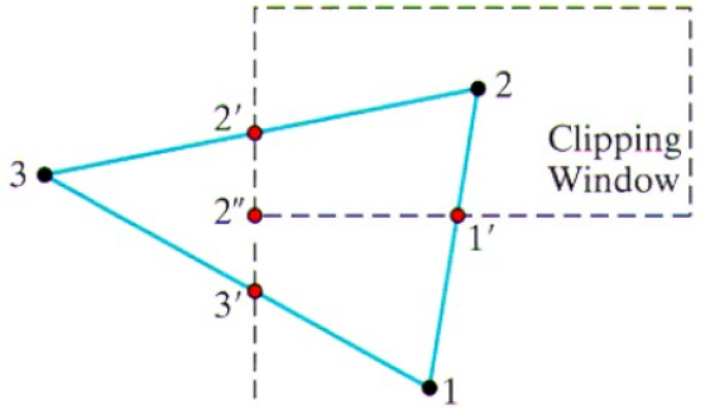


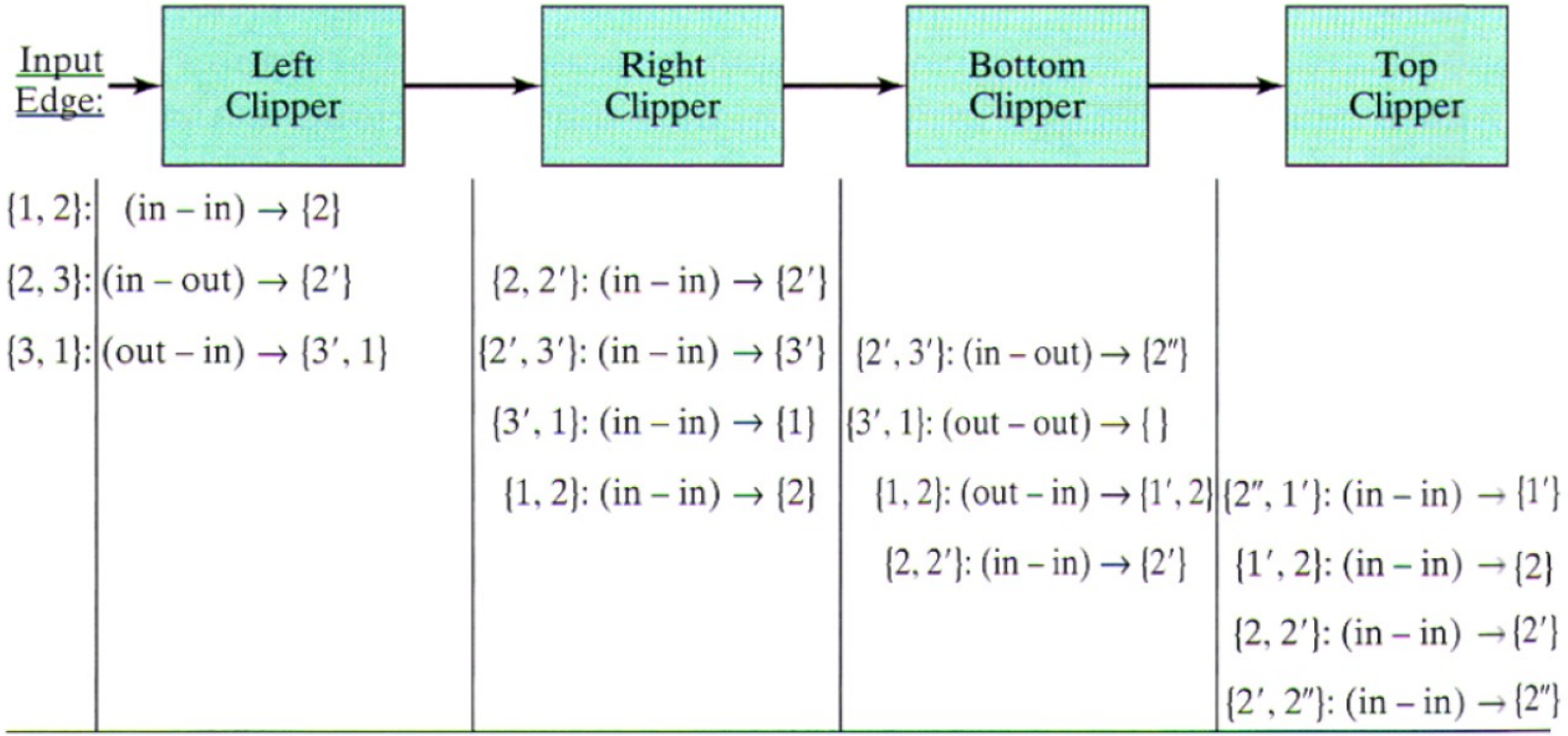
Recorte de Polígonos de Sutherland-Hodgman

- 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.



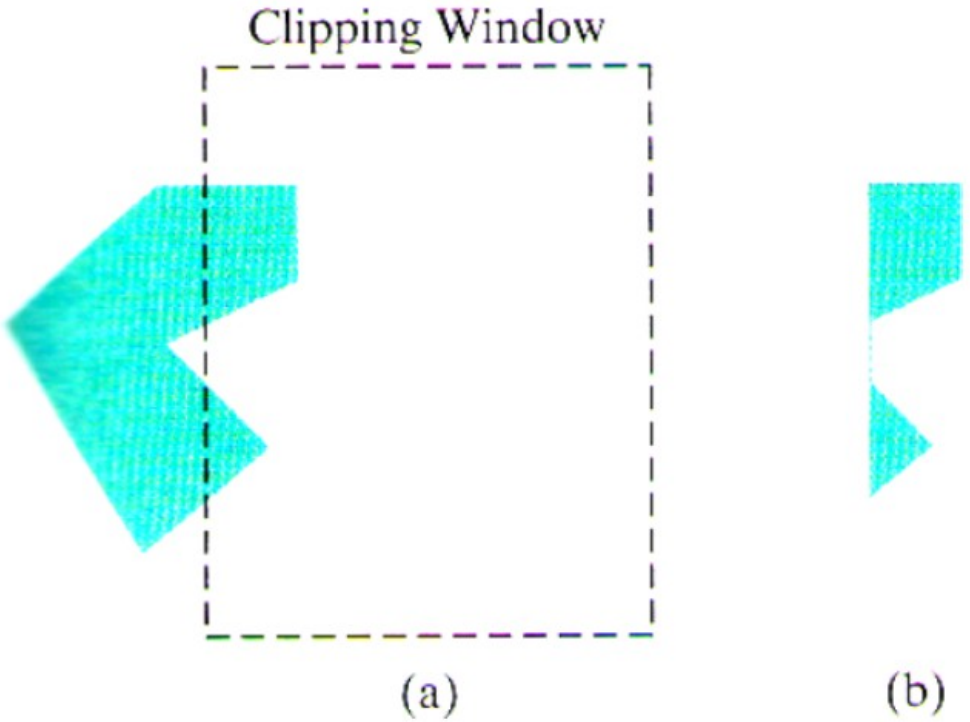
Recorte de Polígonos de Sutherland-Hodgman





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