3 Recortes (Clipping)

Por lo general, cualquier procedimiento que identifica las partes de una imagen que se encuentran ya sea adentro o afuera de una región especifica del espacio se denomina **algoritmo de recorte** o solo **recorte**.
La región contra la cual se recorta un objeto se llama **ventana de recorte**.

Las aplicaciones de recorte incluyen:
- la extracción de parte de una escena definida para verla;
- el despliegue de ventanas múltiples; y
- operaciones de pintura y dibujo que permiten seleccionar partes de una imagen para copiarlas, moverlas, suprimirlas o duplicarlas.

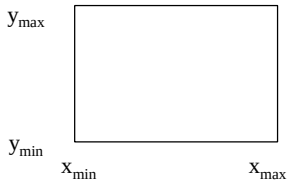
Dependiendo de la aplicación, la ventana de recorte puede ser un polígono general o incluso puede tener fronteras curvas.
Se consideraran principalmente métodos de recorte que emplean regiones rectangulares de recorte.

En el caso de una imagen, se desea desplegar aquellas partes de la imagen que se localizan dentro del área de la ventana. Se elimina cualquier parte que se encuentre afuera de la ventana.
Existen varias metodologías para el recorte de una imagen:
- Recorte analítico en coordenadas mundiales: Se pueden aplicar los algoritmos de recorte en coordenadas mundiales, de modo que solo el contenido de la ventana se mapea a coordenadas de dispositivo.
- Recorte durante la conversión de línea de rastreo en coordenadas de dispositivo: Es posible mapear toda la imagen en coordenadas mundiales primero a coordenadas de dispositivo, para recortarlas después contra las fronteras del puerto de vistas.

La meta es hacer los recortes de la forma mas eficiente posible, lo cual depende del tipo de imagen a desplegarse.
Las rutinas de recorte de líneas y polígonos son componentes estándares de los paquetes gráficos, y muchos paquetes manejan incluso objetos curvos, los cuales aveces se pueden manejar con aproximaciones a segmentos de línea recta.

3.1 Recorte de Puntos

Si se supone que la ventana de recortes es un rectángulo en una posición estándar, como se muestra a continuación



se guarda un punto P = (x,y) para desplegarlo si se satisfacen las desigualdades siguientes:

$$x_{min} \le x \le x_{max}$$
$$y_{min} \le y \le y_{max}$$

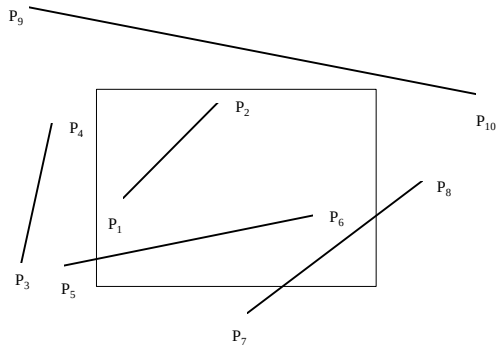
donde las aristas de la ventana de recorte (x_{min}, x_{max}, y_{min}, y_{max}) pueden ser ya sea las fronteras de la ventana en coordenadas mundiales o las fronteras del puerto de vista.
Si no satisface cualquiera de estas cuatro desigualdades, se recorta el punto (no se guarda para despliegue).

Aunque es recorte de puntos se utiliza con menor frecuencia que el recorte de líneas o de polígonos, algunas aplicaciones pueden requerir un procedimiento de recorte de puntos.

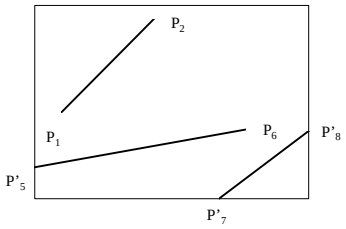
Por ejemplo, es posible aplicar el recorte de puntos en escenas que implican explosiones o espumas de mar que se modelaron con partículas (puntos) que se distribuyen en alguna región de la escena.

3.2 Recorte de Líneas

Las líneas intersectando una región rectangular de recorte (o cualquier polígono convexo) siempre se recortan a una solo segmento de línea.
Las líneas que yacen sobre el borde del rectángulo se consideran adentro y se despliegan.
La siguiente figura muestra ejemplos de líneas recortadas:



(a) antes del recorte



(b) recortadas

3.2.1 Recorte Analítico

Se puede probar un segmento de línea dado para determinar si cae por completo dentro del ventana de recorte.
Si no es así, se determina si cae por completo fuera de la ventana.
Por ultimo, si no se puede identificar una línea que se localice por completo adentro o afuera, se debe realizar cálculos de intersección con una o mas fronteras de recorte.
Se procesa las líneas mediante las pruebas “interna y externa” al verificar los extremos de la línea.
Se guarda una línea que tiene ambos extremos adentro de todas las fronteras de recorte, como la línea de P_1 a P_2.
Una línea cuyos extremos se localizan afuera de cualquiera de las regiones de recorte, como la línea de P_3 a P_4, esta afuera de la ventana.
Todas las otras líneas cruzan una o mas fronteras y pueden implicar el calculo de múltiples puntos de intersección.
Con el fin de reducir los cálculos, se trata de establecer algoritmos de recorte que identifiquen de manera eficiente las líneas exteriores y reduzcan los cálculos de intersección.

Para un segmento de línea con los extremos (x_1, y_1) y (x_2, y_2) y uno o ambos extremos afuera del rectángulo de recorte, la representación paramétrica, para $0 \le u \le 1$,

$$x = x_1 + u(x_2 - x_1)$$
$$y = y_1 + u(y_2 - y_1)$$

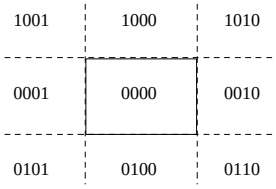
puede servir para determinar los valores del parámetro u para una intersección con las coordenadas de la frontera de recorte.

Si el valor de *u* para una intersección con una arista de frontera del rectángulo esta fuera del rango 0 a 1, la línea no se encuentra en el interior de la ventana en esa frontera.
Si el valor de *u* esta en el rango de 0 a 1, de hecho, el segmento de línea cruza el área de recorte.
Se puede aplicar este método en cada arista de frontera de recorte a su vez, para determinar si se debe desplegar cualquier parte del segmento de línea.
Los segmentos de línea para lelos a la ventana se pueden manejar como casos especiales.

El recorte de segmentos de línea con estas pruebas paramétricas requiere una gran cantidad de cálculos y se pueden aplicar procedimientos mas rápidos para el recorte.

3.2.2 Recorte de Líneas de Cohen-Sutherland

Este es uno de los procedimientos de recorte de líneas mas antiguo y común.
Por lo general, el método acelera el procesamiento de segmentos de línea al realizar pruebas iniciales que reducen el numero de intersecciones que se deben calcular.
A todos los extremos de línea de una imagen se asigna un código binario de cuatro dígitos, que se conoce como **código de región**, el cual identifica la localización del punto con respecto de las fronteras del rectángulo de recorte.
Como se muestra en la siguiente figura, las regiones se determinan en relación con las fronteras.



Se utiliza cada posición de bit en el código de región para indicar una de las cuatro posiciones de coordenadas relativas del punto con respecto de la ventana de recorte: a la izquierda, a la derecha, arriba o abajo.
Al numerar las posiciones de bit en el código de región como 1 a 4 de derecha a izquierda, se pueden correlacionar las regiones de coordenadas con las posiciones de bit como

bit 1: izquierda
bit 2: derecha
bit 3: abajo
bit 4: arriba

Un valor de 1 en cualquier posición de bit indica que el punto esta en posición relativa; de otro modo se establece como 0 la posición de bit.
Si un punto esta adentro del recorte del rectángulo, tiene un código de región 0000.
Un punto que se halla abajo y a la izquierda del rectángulo tiene un código de 0101.

Se determinan los valores de bit en el código de región al comparar los valores de las coordenadas de los extremos (x,y) con las fronteras de recorte.
Se especifica el bit 1 como 1 si $x < wx_{min}$, donde x es el extremo de la línea.
Se pueden determinar los otros tres valores de bit al utilizar comparaciones similares.
En el caso de los lenguajes que permiten el manejo de bits, se pueden determinar los valores de bit del código de región con los dos pasos siguientes:

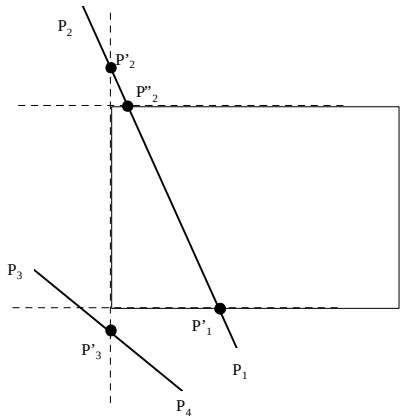
1. Se calculan las diferencias entre las coordenadas de los extremos de la línea y las fronteras de recorte.
2. Se utiliza el bit del signo resultante de cada calculo de diferencias para establecer el valor correspondiente en el código de región.
   El bit 1 es el bit de signo de $x - wx_{min}$.
   El bit 2 es el bit de signo de $wx_{max} - x$.
   El bit 3 es el bit de signo de $y - yw_{min}$.
   El bit 4 es el bit de signo de $yw_{max} - y$.

Cuando se establecen los códigos de región para todos los extremos de línea se puede determinar con rapidez que líneas se localizan por completo dentro de la ventana de recorte y que líneas se hallan fuera en su totalidad.

- Cualquier línea que se encuentre por completo dentro de las fronteras de la ventana tiene un código de región de 0000 para ambos extremos y se aceptan estas líneas en forma común.
- Cualquier línea que tenga un 1 en la misma posición de bit en los códigos de bit para cada extremo se halla por completo afuera del rectángulo de recorte y se rechazan estas líneas en forma común. Se eliminaría la línea que tiene un código de región 1001 para un extremo y un código de 0101 para el otro. Los dos extremos de esta línea están a la izquierda del rectángulo de recorte como lo indica el 1 en la primera posición de bit de cada código de región.

Un método que se puede emplear para probar estas líneas para el recorte total consiste en realizar la operación lógica *and* con ambos códigos de región.

- Las líneas que no se pueden identificar como completamente adentro o completamente afuera de la ventana de recorte mediante estas pruebas se verifican para saber si intersectan las fronteras de la ventana. Como se muestra en la siguiente figura, estas líneas pueden o no cruzar hacia el interior de la ventana.



Se inicia el proceso de recorte para una línea al comparar un extremo que se halla fuera con una frontera de recorte para determinar el porcentaje de la línea que se puede eliminar. Así se verifica la parte restante de la línea contra las demás fronteras y se continua ya sea hasta que se elimine por completo la línea o hasta que se localice una sección dentro de la ventana.
Se establece el algoritmo para verificar los extremos de la línea contra las fronteras de recorte en el orden izquierda, derecha, abajo, arriba.

Con objeto de ilustrar los pasos específicos en el recorte de líneas contra fronteras rectangulares al utilizar el algoritmo de Cohen-Sutherland, se demuestra como se podrían procesar las líneas de la figura anterior.

1. Al iniciar con el extremo de abajo de la línea de P_1 a P_2, se verifica P_1 contra las fronteras izquierda, derecha e inferior a su vez, y se encuentra el punto de intersección P'_1 con la frontera inferior y se elimina la sección de la línea de P_1 a P'_1. Ahora, se ha reducido la línea a la sección de P'_1 a P_2.
2. Puesto que P_2 esta fuera de la ventana de recorte, se verifica esta tabla contra las fronteras y se encuentra que se localiza a la izquierda de la ventana. Se calculo el punto de intersección P'_2, pero este punto esta arriba de la ventana.
3. De esta manera, el calculo final de la intersección da como resultado $P"_2$ y se guarda la línea de P'_1 a $P"_2$.
4. Con esta operación se concluyen el procesamiento de esta línea, entonces se guarda esta línea y se continua con la siguiente.
5. El punto P_3 en la siguiente línea esta a la izquierda del rectángulo de recorte, se determina la intersección P'_3 y se elimina la sección de la línea de P_3 a P'_3.
6. Al verificar los códigos de región para la sección de la línea de P'_3 a P_4, se encuentra que el resto de la línea se halla abajo de la ventana de recorte y también se puede eliminar.

Se pueden calcular los puntos de intersección con una frontera de recorte al utilizar la forma de intersección de la pendiente de la ecuación de líneas.
Para una línea con coordenadas de extremo (x_1, y_1) y (x_2, y_2), la coordenada de y del punto de intersección con una frontera vertical se puede calcular como

$$y = y_1 + m(x - x_1)$$

donde se asigna el valor de x ya sea x_{min} o x_{max}, y se calcula la pendiente de la línea como
$m = (y_2 - y_1) / (x_2 - x_1)$.
De modo similar, si se busca la intersección con una frontera horizontal, se puede calcular la coordenada de x como

$$x = x_1 + (y - y_1) / m$$

ya sea con y como y_{min} o y_{max}.

3.2.3 Recorte de Líneas de Liang-Barsky

El algoritmo de Cohen-Sutherland es probablemente el algoritmo de recorte de línea mas comúnmente utilizado ya que es el que mas tiempo ha estado y ha sido publicado extensivamente.

En 1978, Cyrus y Beck publicaron un algoritmo que toma un enfoque para recorte de línea fundamentalmente diferente y generalmente mas eficiente.
La técnica de Cyrus y Beck se puede utilizar para recortar una línea de 2D contra un rectángulo o un polígono convexo arbitrario en un plano, o una línea de 3D contra un poliedro convexo arbitrario en 3D.

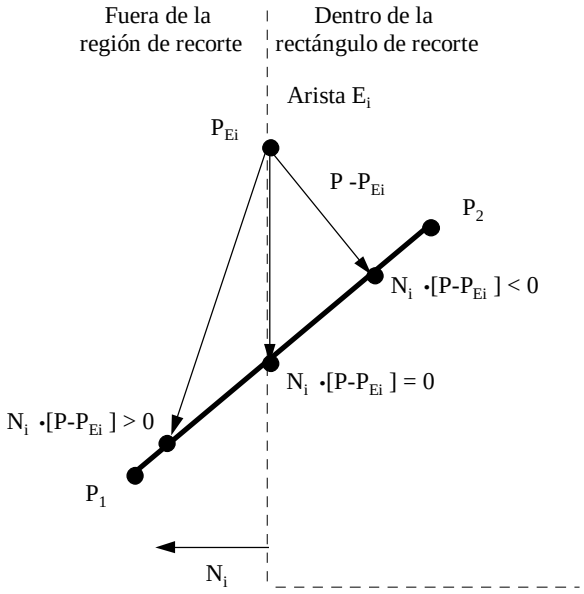
En 1984, Liang y Barsky desarrollaron de forma independiente un algoritmo para recorte de línea paramétrica mas eficiente que es especialmente rápido en el caso de regiones de recorte horizontales y verticales de 2D y 3D.
Además de aprovechar estos bordes de recorte simples, introdujeron pruebas triviales de rechazo mas eficientes que trabajan para regiones de recorte generales. Además de tomar provecho de estos bordes sencillos de recorte, introdujeron pruebas de rechazo triviales mas eficientes que trabajan para regiones de recorte generales.

Aquí se sigue el desarrollo original de Cyrus-Beck para introducir el concepto de recorte paramétrico.
Como estamos interesados solo con rectángulos de recorte horizontales y verticales, se reduce hacia el final la formulación de Cyrus-Beck a la mas eficiente formulación de Liang-Barsky.

En el algoritmo de Cohen-Sutherlan, para las líneas que no pueden aceptarse o rechazarse trivialmente, se calcula la intersección (x,y) de un segmento de línea con un arista de recorte sustituyendo el valor conocido de x o y del arista de recorte vertical u horizontal, respectivamente.
El algoritmo de línea paramétrica, sin embargo, encuentra el valor del parámetro u en la representación paramétrica del segmento de línea para el punto donde el segmento intersecta la línea infinita del arista de recorte.
Como todas las aristas de recorte son, en general, intersectadas por la línea, cuatro valores de u se calculan.
Una serie de comparaciones sencillas se usan para determinar cual (si alguno) de los cuatro valores de u corresponde a las intersecciones actuales.
Solo entonces se calculan los valores (x,y) para una o dos intersecciones reales.
En general, este enfoque ahorra tiempo sobre el algoritmo de calculo de intersección de Cohen-Sutherland ya que evita el ciclo repetitivo necesario para recortar sobre múltiples aristas.
También, cálculos en un espacio de parámetro de 1D son mas sencillos que aquellos en coordenadas de 3D.
Liang y Barsky mejoran sobre Cyrus-Beck al examinar cada valor de u según se genera, rechazando algunos segmentos de línea antes de haber calculado los cuatro valores de u.

El algoritmo de Cyrus-Beck se basa en la siguiente formulación de la intersección entre dos líneas.
La siguiente figura muestra un arista E_i de la región de recorte y la normal N_i hacia afuera de la arista, al igual que una línea de P_1 a P_2 que debe recortarse a la arista.

La arista o el segmento de línea podrían ser extendidos para encontrar el punto de intersección.



Los segmentos de línea expresados de forma paramétrica como, para $0 \le u \le 1$,

$$x = x_1 + u(x_2 - x_1) = x_1 + u\Delta x$$
$$y = y_1 + u(y_2 - y_1) = y_1 + u\Delta y$$

o

$$P = P_1 + u(P_2 - P_1) = P_1 + u\Delta P$$

donde $P = (x,y)$, $P_1 = (x_1,y_1)$ y $P_2 = (x_2,y_2)$.

Se escoge un punto arbitrario P_{Ei} sobre el arista E_i y se consideran los tres vectores $P - P_{Ei}$ de P_{Ei} a los tres puntos designados en la línea de P_1 a P_2:

- el punto de intersección a ser determinado,
- un extremo de la línea en el plano interno, y
- un extremo de la línea en el plano externo.

Se puede distinguir en que región esta un punto viendo el valor del producto de punto $N_i \bullet \left[P - P_{E_i}\right]$.

- Este valor es negativo para un punto está en el plano interior, ya que el ángulo entre los dos vectores está dentro del intervalo de 90 a 180 grados ($a \cdot b \cos q$), siendo $\cos q$ negativo en tal intervalo.
- cero para un punto en la línea conteniendo el arista ($a \cdot b \cos q$), ya que el ángulo entre los dos vectores seria 90 grados, y cos 90 = 0.
- positivo para un punto que esta en el medio plano exterior, ya que el ángulo entre los dos vectores está dentro del intervalo de 0 a 90 grados ($a \cdot b \cos q$), siendo $\cos q$ positivo en tal intervalo.

Ahora se puede resolver el valor de u en la intersección de P_1P_2 con el arista:

$$N_i \bullet \left[P - P_{E_i}\right] = 0$$

Primero se sustituye por P:

$$N_i \bullet \left[P_1 + u(P_2 - P_1) - P_{E_i}\right] = 0$$

Luego, se agrupan términos y se distribuye el producto punto:

$$N_i \bullet \left[P_1 - P_{E_i}\right] + N_i \bullet u\left[P_2 - P_1\right] = 0$$

Si $D = (P_2 - P_1)$ es el vector de P_1 a P_2, y se resuelve para u:

(100) $$u = \frac{N_i \bullet \left[P_1 - P_{E_i}\right]}{-N_i \bullet D}$$

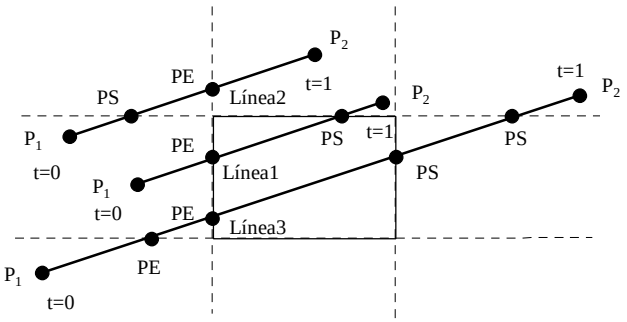
Esto da un valor valido para u solo si el denominador de la expresión no es cero.
Para que esto sea cierto, el algoritmo verifica que:

- $N_i \neq 0$ (o sea, la normal no debe ser 0; esto solo puede ocurrir por un error)
- $D \neq 0$ (o sea, $P_1 \neq P_2$)
- $N_i \bullet D \neq 0$ ($N_i \bullet D = N_i D \cos q$, donde $\cos q = 0$ para los ángulos de 90° y 270°, o sea, N_i y la línea de P_1 a P_2 no son perpendiculares, o, dicho de otro modo, el arista E_i y la línea de P_1 a P_2 no son paralelas. Si fueran paralelas, no habría una sola intersección para esta arista, por lo cual el algoritmo se mueve al siguiente caso)

(Para dos vectores partiendo del origen, V_1 y V_2, el producto punto de $V_1 \bullet V_2 = x_1x_2 + y_1y_2$)

La ecuación (100) puede utilizarse para encontrar las intersecciones de P_1P_2 y cada arista del rectángulo de recorte.
1. Se hace este calculo determinando la normal y un punto arbitrario P_{Ei} - por ejemplo, un extremo del arista - para cada arista de recorte, y aplicando estos cálculos y valores para todas los segmentos de línea.
2. Dado los cuatro valores de u para un segmento de línea (uno por cada arista), el siguiente paso es determinar cual (si hay) de los valores corresponde a intersecciones internas del segmento de línea con aristas del rectángulo de recorte.
3. Como un primer paso, cualquier valor de u fuera del intervalo [0 1] puede descartarse, ya que esta fuera de P_1P_2. Luego, se necesita determinar si la intersección yace en el borde de recorte.

Podríamos simplemente tratar de clasificar los valores restantes de u, dentro del intervalo [0 1], escogiendo los valores intermedios de u para puntos de intersección, como se muestra en la siguiente figura para el caso de la línea 1.



Pero como distinguimos este caso del caso de la línea 2, donde ninguna porción del segmento de línea yace en el rectángulo de recorte y los valores intermedios de u corresponden a puntos no sobre el borde de recorte ?
También, cual de las cuatro intersecciones de la línea 3 son las que están en el borde de recorte ?

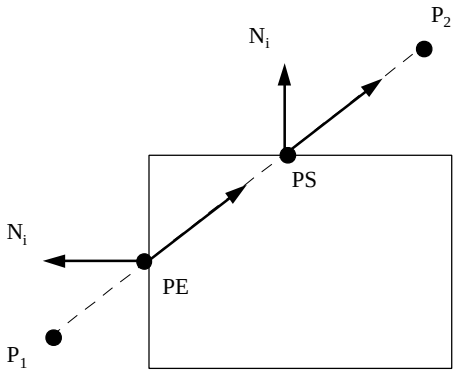
Las intersecciones en la figura anterior se caracterizan como *potencialmente entrando* (PE) o *potencialmente saliendo* (PS) el rectángulo de recorte, como sigue:

- Si el movimiento de P_1 a P_2 causa cruzar una arista particular para entrar al plano interno del arista, la intersección es PE;
- si causa salir del plano interno, es PS.

Con esta distinción, dos puntos de intersección interiores de una línea intersectando el rectángulo de recorte tienen etiquetas opuestas.

Formalmente, las intersecciones pueden clasificarse como PE o PS en la base del ángulo entre de P_1P_2 y N_i:

- Si el ángulo es menor que 90°, la intersección es PS;
- si es mayor que 90°, es PE.



La información se contiene en el signo del producto de punto de N_i y P_1P_2:

$N_i \bullet D < 0 \Rightarrow$ PE (ángulos mayor a 90°)

$N_i \bullet D > 0 \Rightarrow$ PS (ángulos menor a 90°)

o

$-N_i \bullet D > 0 \Rightarrow$ PE (ángulos mayor a 90°)

$-N_i \bullet D < 0 \Rightarrow$ PS (ángulos menor a 90°)

$N_i \bullet D$ es simplemente el denominador de la ecuación (100), lo que significa que, en el proceso de calcular u, la intersección puede categorizarse trivialmente.

Con esta categorización, la línea 3 en la figura sugiere el ultimo paso en el proceso.

1. Se debe escoger un par (PE,PS) que defina la línea recortada.
2. La porción de línea infinita a través de P_1P_2 que esta dentro de la región recortada se limita por la intersección de PE con el mayor valor de u (en el caso de que existan varias PEs), el cual se llama u_E, y la intersección de PS con el menor valor de u (en el caso de que existan varias PSs) el cual se llama u_S. El segmento de línea intersectante se define entonces por el rango (u_E, u_S).
3. Pero como estamos interesados en intersectar P_1P_2, no la línea infinita, la definición del rango debe ser modificada para que u=0 sea un limite inferior para u_E y u=1 sea un limite superior para u_S.

Que ocurre si $u_E > u_S$?

Este es exactamente el caso para la línea 2, y se rechaza.

Valores de u_E y u_S que corresponden a intersecciones actuales se usan para calcular las coordenadas correspondientes de x y y.

La siguiente tabla muestra para cada arista los valores de N_i, un punto canónico sobre la arista, P_{Ei}, el vector P_1 - P_{Ei} y el parámetro u dado. (PE o PS se dan de acuerdo a que $x_2 > x_1$ y $y_2 > y_1$)

Arista de Recorte$_i$	Normal N_i	P_{Ei}	$P_1 - P_{Ei}$	$u = \frac{N_i \bullet \left[P_1 - P_{E_i}\right]}{-N_i \bullet D}$	PE o PS
izquierda: $x = x_{min}$	(-1,0)	(x_{min}, y)	$(x_1 - x_{min}, y_1 - y)$	$-(x_1 - x_{min}) / (x_2 - x_1)$	PE
derecha: $x = x_{max}$	(1,0)	(x_{max}, y)	$(x_1 - x_{max}, y_1 - y)$	$(x_1 - x_{max}) / -(x_2 - x_1)$	PS
abajo: $y = y_{min}$	(0,-1)	(x, y_{min})	$(x_1 - x, y_1 - y_{min})$	$-(y_1 - y_{min}) / (y_2 - y_1)$	PE
arriba: $y = y_{max}$	(0,1)	(x, y_{max})	$(x_1 - x, y_1 - y_{max})$	$(y_1 - y_{max}) / -(y_2 - y_1)$	PS

Como una de las coordenadas de cada normal es 0, no es necesario obtener la coordenada correspondiente en P_{Ei} (denotada por una x o y intermedia).
Se ve de la tabla que el numerador, el producto $N_i \bullet \left[P_1 - P_{E_i}\right]$ determinando si el extremo P_1 esta dentro o fuera de un arista especificado, se reduce a la distancia horizontal o vertical del punto al arista.

El producto punto del denominador $N_i \bullet D$, que determina si la intersección es potencialmente entrante o saliente, se reduce a $\pm \Delta x$ o Δy ($\Delta x = x_2 - x_1$, $\Delta y = y_2 - y_1$); si Δx es positiva, la línea se mueve de izquierda a derecha y PE se ubica en el arista izquierdo, y PS en el arista derecho. (Los casos alternos son análogos.)
Finalmente, el parámetro u, la razón entre el numerador y el denominador, se reduce a la distancia a una arista dividida por Δx o Δy, exactamente la constante de proporcionalidad que se puede calcular directamente de la formulación de la línea paramétrica.
Es importante preservar los signos del numerador y denominador en lugar de cancelar signos de menos, ya que el denominador y numerador mantienen información sobre la dirección de las distancias, siendo importante para el algoritmo.

Por ejemplo, para una ventana de recorte dada por

$x_{min} = 10$, $x_{max} = 20$, $y_{min} = 15$, $y_{max} = 20$

Consideramos el segmento de línea:

$P_1 = (0,12)$, $P_2 = (30,22)$

Los cuatro parámetros u se muestran en la siguiente tabla:

Arista de Recorte$_i$	Normal N_i	$u = \frac{N_i \bullet \left[P_1 - P_{E_i}\right]}{-N_i \bullet D}$
izquierda: $x_{min} = 10$	(-1,0)	$-(x_1 - x_{min}) / (x_2 - x_1) = -(0-10)/(30-0) = 1/3$ (PE)
derecha: $x_{max} = 20$	(1,0)	$(x_1 - x_{max}) / -(x_2 - x_1) = (0-20)/-(30-0) = 2/3$ (PS)
abajo: $y_{min} = 15$	(0,-1)	$-(y_1 - y_{min}) / (y_2 - y_1) = -(12-15)/(22-12) = 3/10$ (PE)
arriba: $y_{max} = 20$	(0,1)	$(y_1 - y_{max}) / -(y_2 - y_1) = (12-20)/-(22-12) = 8/10$ (PS)

Utilizando la ecuación paramétrica para la línea dada por

$x = x_1 + u(x_2 - x_1)$
$y = y_1 + u(y_2 - y_1)$

Se obtiene las intersecciones para el mayor PE dado por $u = 1/3$

$x_E = 0 + (1/3)(30) = 10$
$y_E = 12 + (1/3)(10) = 15\ 1/3$

y para el menor PS dado por $u = 2/3$

$$x_S = 0 + (2/3)(30) = 20$$
$$y_S = 12 + (2/3)(10) = 18\ 2/3$$

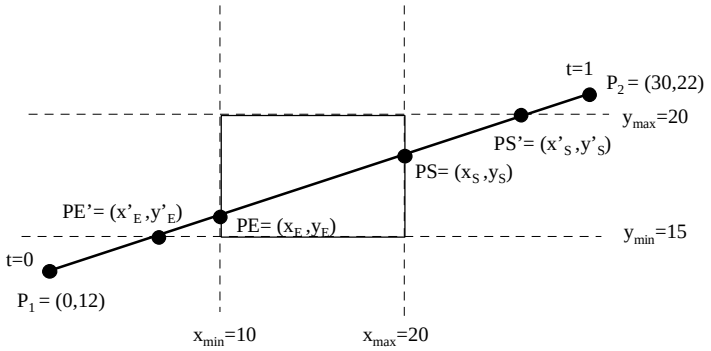
Las otras dos intersecciones dan, para $u = 3/10$

$$x'_E = 0 + (3/10)(30) = 9$$
$$y'_E = 12 + (3/10)(10) = 15$$

y para $u = 8/10$

$$x'_S = 0 + (8/10)(30) = 24$$
$$y'_S = 12 + (8/10)(10) = 20$$

La línea se muestra en la siguiente figura:



El algoritmo se puede resumir simplificando las ecuaciones como se muestra a continuación.

Las condiciones de recorte de puntos se describe de forma paramétrica:

$$x_{\min} \le x_1 + u\Delta x \le x_{\max}$$
$$y_{\min} \le y_1 + u\Delta y \le y_{\max}$$

Cada una de estas cuatro desigualdades se puede expresar como

$$up_k \le q_k, k = 1, 2, 3, 4$$

donde los parámetros p y q se definen como

$$p_1 = -\Delta x \quad q_1 = x_1 - wx_{\min}$$
$$p_2 = \Delta x \quad q_2 = wx_{\max} - x_1$$
$$p_3 = -\Delta y \quad q_3 = y_1 - wy_{\min}$$
$$p_4 = \Delta y \quad q_4 = wy_{\max} - y_1$$

La correspondencia con el análisis anterior es: $q_k = P_1 - P_{Ek}$ y $p_k = P_2 - P_1$, y la relación

$$u = q_k / p_k$$

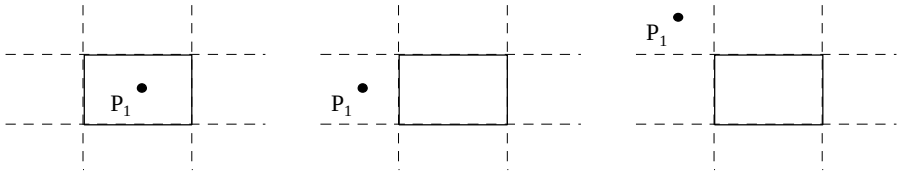
corresponde al resultado obtenido anteriormente para

$$u = \frac{N_i \bullet \left[P_1 - P_{E_i}\right]}{-N_i \bullet D}$$

3.2.4 Recorte de Líneas de Nicholl-Lee-Nicholl

Al crear mas regiones alrededor de la ventana de recorte, el algoritmo de Nicholl-Lee-Nicholl (NLN) evita los recortes múltiples de un segmento de línea individual.
Por ejemplo, en el método de Cohen-Sutherland las intersecciones múltiples se pueden calcular a lo largo de la trayectoria de una sola línea antes de calcular una intersección con el rectángulo de recorte o rechazar por completo la línea.
Estos cálculos extras de intersección se eliminan en el algoritmo de NLN al realizar mas pruebas de región antes de calcular las posiciones de las intersecciones.
Comparado con los algoritmos de Cohen-Sutherland y de Liang-Barsky, el algoritmo Nicholl-Lee-Nicholl lleva a cabo menos comparaciones y divisiones.
La deficiencia del algoritmo NLN es que solo puede aplicarse al recorte bidimensional, en tanto que los métodos de Cohen-Sutherland y de Liang-Barsky se extienden fácilmente a escenas tridimensionales.

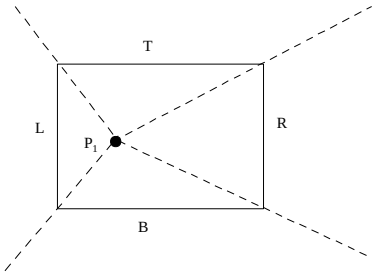
Para una línea con extremos P_1 y P_2, primero se determina la posición del punto P_1 para las nueve regiones posibles relativas al rectángulo de recorte.
Solo es necesario considerar las regiones que se muestran en la siguiente figura:



P_1 en la ventana P_1 en región de arista P_1 en región de esquina

Si P_1 se encuentra en cualquier otra de las seis regiones, se le puede mover a una de las tres regiones de la figura anterior utilizando una transformación de simetría.
Por ejemplo, la región que se halla directamente arriba de la ventana de recorte se puede transformar en la región a la izquierda de la ventana de recorte utilizando una reflexión con respecto de la línea $y = -x$, o se puede emplear una rotación de 90° en sentido del reloj.

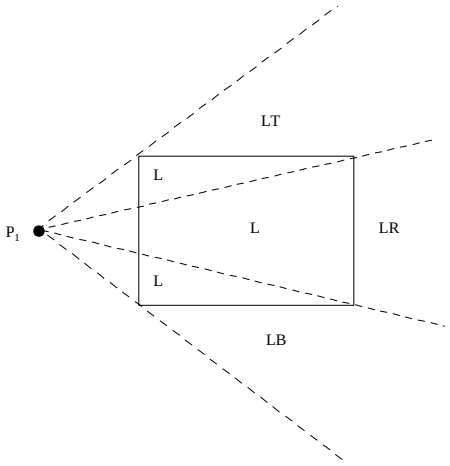
A continuación, determinamos la posición de P_2 en relación con P_1.
Para hacer esto, creamos algunas regiones nuevas en el plano, dependiendo de la ubicación de P_1.
Las fronteras de las nuevas regiones son segmentos de línea medio infinitos que empiezan en la posición de P_1 y pasan a través de las esquinas de la ventana.
Si P_1 se halla adentro de la ventana de recorte y P_2 se encuentra afuera, establecemos las cuatro regiones que se muestran en la siguiente figura:



Así, se realiza la intersección con la frontera de ventana apropiada, dependiendo de cual de las cuatro regiones (*L*, *T*, *R* o *B*) contiene P_2.

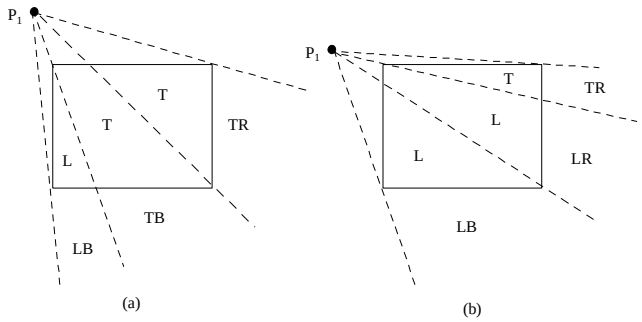
Desde luego, si tanto P_1 como P_2 se encuentran adentro del rectángulo de recorte, simplemente guardamos la línea completa.

Si P_1 se halla en la región a la izquierda de la ventana, establecemos las cuatro regiones, *L*, *LT*, *LR* y *LB*, que se muestran en la siguiente figura:



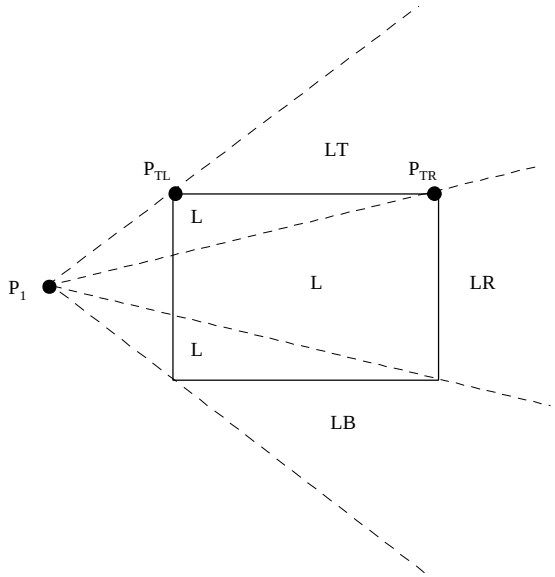
Estas cuatro regiones determinan una frontera única para el segmento de línea.
Por ejemplo, si P_2 se encuentra en la región *L*, recortamos la línea en la frontera izquierda y guardamos el segmento de línea desde este punto de intersección hasta P_2.
Pero si P_2 se halla en la región *LT*, guardamos el segmento de línea desde la frontera izquierda de la ventana hasta la frontera superior.
Si P_2 no esta en ninguna de estas cuatro regiones, *L*, *LT*, *LR* y *LB*, se recorta la línea completa.

Para el tercer caso, cuando P_1 se halla a la izquierda y arriba de la ventana de recorte, utilizamos las regiones de recorte que se muestran en la siguiente figura:



En este caso, tenemos las dos posibilidades que se muestran, dependiendo de la posición de P_1 con respecto a la esquina superior izquierda de la ventana.
Si P_2 se encuentra en una de las regiones *T*, *L*, *TR*, *TB*, *LR* o *LB*, determina una arista única de recorte de ventana para los cálculos de las intersecciones.
De otra manera, se rechaza la línea completa.

Para determinar la región en la cual se localiza P_2, comparamos la inclinación de la línea con las inclinaciones de las fronteras de las regiones de recorte.
Por ejemplo, si P_1 se encuentra a la izquierda del rectángulo de recorte, como se muestra en la siguiente figura:



entonces P_2 esta en la región *LT* si

inclinación P_1P_{TR} < inclinación P_1P_2 < inclinación P_1P_{TL}

o

$$(y_T - y_1) / (x_R - x_1) < (y_2 - y_1) / (x_2 - x_1) < (y_T - y_1) / (x_L - x_1)$$

Y se recorta toda la línea si

$$(y_2 - y_1) / (x_2 - x_1) > (y_T - y_1) / (x_L - x_1)$$

o

$$(y_T - y_1) (x_2 - x_1) < (x_L - x_1) (y_2 - y_1)$$

La diferencia de coordenadas y los cálculos de producto que se utilizan en las pruebas de inclinación se guardan y se utilizan también en los cálculos de las intersecciones.
A partir de las ecuaciones paramétricas

$$x = x_1 + u(x_2 - x_1)$$
$$y = y_1 + u(y_2 - y_1)$$

una posición de intersección de x en la frontera izquierda de la ventana es $x = x_L$, con
$u = (x_L - x_1) / (x_2 - x_1)$, de modo que la posición de la intersección de y es

$$y = y_1 + (y_2 - y_1) (x_L - x_1) / (x_2 - x_1)$$

Y una posición de intersección en la frontera superior tiene $y = y_T$, con
$u = (y_T - y_1) / (y_2 - y_1)$, de modo que la posición de la intersección de x es

$$x = x_1 + (x_2 - x_1) (y_T - y_1) / (y_2 - y_1)$$

3.2.5 Recorte de Líneas para ventanas de recorte no rectangulares

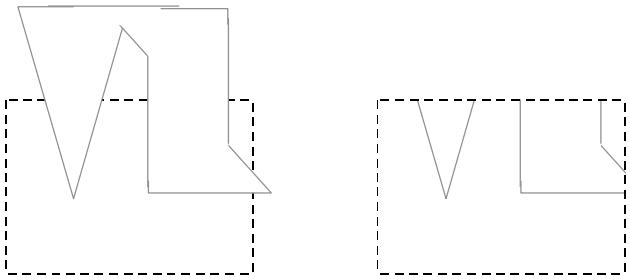
Los algoritmos que se basan en ecuaciones paramétricas de línea, como el método de Liang-Barsky y el planteamiento de Cyrus-Beck, se pueden extender con facilidad a ventanas de polígonos convexos.
Esto se logra mediante la modificación del algoritmo para incluir las ecuaciones paramétricas para las fronteras de la región de recorte.

El despliegue preliminar de los segmentos de línea se puede lograr mediante el procesamiento de líneas contra las extensiones de las coordenadas del polígono de recorte.
Para las regiones cóncavas de recorte de polígono, aun se puede aplicar estos procedimientos paramétricos de recorte si primero se divide el polígono cóncavo en un conjunto de polígonos convexos.

Las circunferencias u otras regiones de recorte con fronteras curvas también son posibles.
Los algoritmos de recorte para estas áreas son mas lentos porque los cálculos de las intersecciones comprenden ecuaciones curvas no lineales.
En el primer paso, las líneas se pueden recortar contra el rectángulo de la frontera (extensiones de coordenadas) de la región curva de recorte.
Las líneas que se pueden identificar como totalmente afuera del rectángulo de frontera se eliminan.
Para identificar las líneas internas, se puede calcular al distancia de los extremos de la línea desde el centro de la circunferencia.
Si el cuadrado de esta distancia para ambos extremos de una línea es menor o igual al radio al cuadrado, podemos guardar la línea completa.
Después se procesan las líneas restantes mediante los cálculos de las intersecciones, que deben resolver ecuaciones lineales de circunferencia simultáneas.

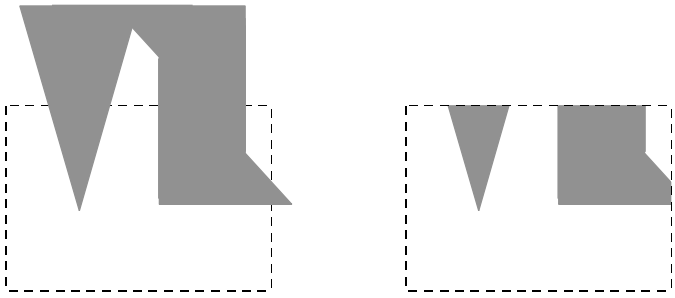
3.3 Recorte de Polígonos

Para recortar polígonos se necesita modificar los procedimientos de recorte de líneas analizados anteriormente.
Una frontera de polígono procesada con un algoritmo de recorte de líneas se puede desplegar como una serie de segmentos de línea sin conectar dependiendo de la orientación del polígono con respecto a la ventana de recorte, como se muestra en la siguiente figura:



Antes del recorte Después del recorte

Lo que en realidad deseamos desplegar es una área limitada después del recorte, como se ve en la siguiente figura:

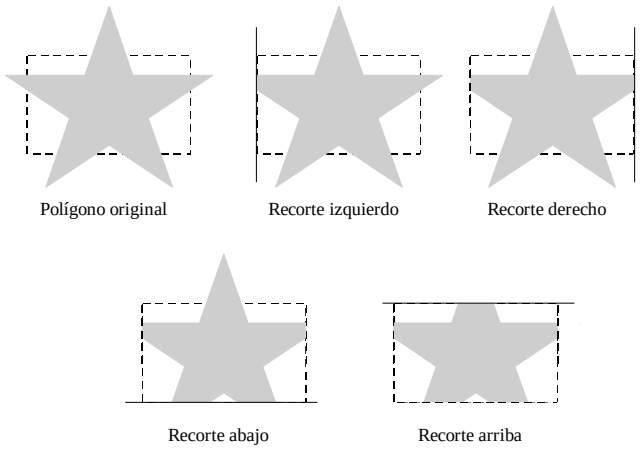


Antes del recorte Después del recorte

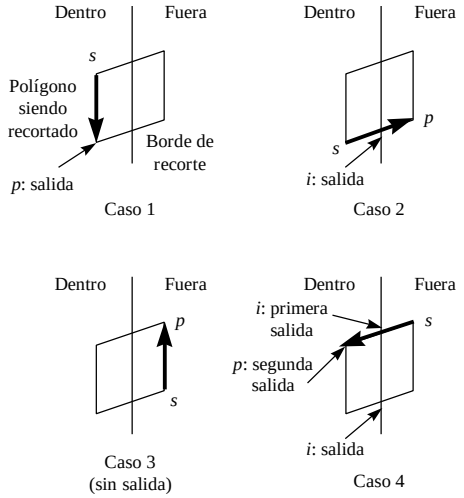
Para el recorte de polígonos, requerimos de un algoritmo que genere una o mas áreas cerradas que después se convierten por rastreo en el área de llenado apropiada.
La salida de un algoritmo de recorte de polígonos debe ser una secuencia de vértices que define las fronteras de los polígonos recortados.

3.3.1 Recorte de Polígonos de Sutherland-Hodgeman

El algoritmo de recorte de polígonos de Sutherland y Hodgeman (1974) utiliza una estrategia de *divide y conquista*: Resuelve una serie de problemas sencillos idénticos que, cuando se combinan, resuelven el problema total.
El problema sencillo es recortar un polígono contra un solo arista de recorte infinito.
Cuatro aristas de recorte, cada uno definiendo un borde del rectángulo de recorte, recortan sucesivamente un polígono contra un rectángulo de recorte.



El algoritmo acepta una serie de vértices de polígono v_1, v_2,..., v_n.
En 2D (el algoritmo también se aplica a 3D), los vértices definen los aristas del polígono de v_i a v_{i+1}, y de v_n a v_1.
Se procesan todos los vértices del polígono contra cada una de las fronteras infinitas del rectángulo de recorte.
Empezando por el conjunto inicial de vértices del polígono, se recortaría el polígono contra la frontera izquierda, derecha, abajo, y arriba, del rectángulo para producir una nueva secuencia de vértices.
En cada paso, cero, uno, o dos vértices se agregan a la lista de salida de vértices que define el polígono recortado.
Hay cuatro casos posibles al procesar los vértices en secuencia, como se muestra a continuación:

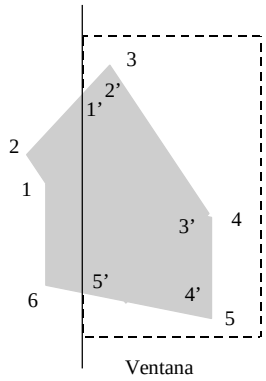


Consideremos el arista de polígono del vértice *s* al vértice *p*, en la figura anterior.
Asumamos que ya se ha manejado el punto inicial *s* en la iteración previa.
Conforme cada par de vértices del polígono se pasa a un algoritmo de recorte de la frontera de la ventana, se realizan las siguientes pruebas:

- En el caso 1, cuando el arista del polígono esta completamente dentro del borde de recorte, el vértice *p* se agrega a la lista de vértices.
- En el caso 2, el punto de intersección *i* se agrega como vértice ya que el arista intersecta el borde.
- En el caso 3, ambos vértices están fuera del borde, por lo cual no se agregan vértices a la lista de salida.
- En el caso 4, el punto de intersección *i* y el vértice *p* se agregan ambos a la lista de salida.

Una vez que se procesan todos los vértices para una frontera de ventana de recorte, la lista de salida de vértices se recorta contra la frontera de ventana siguiente.

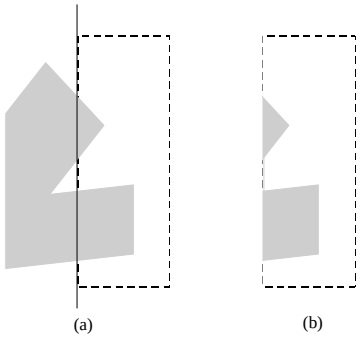
La siguiente figura ilustra el algoritmo:



Se encuentra que los vértices 1 y 2 se hallan afuera de la frontera.
Al desplazarnos a lo largo del vértice 3, que se encuentra adentro, calculamos la intersección y guardamos tanto el punto de intersección como el vértice 3.
Se determina que los vértices 4 y 5 se encuentran adentro y también se guardan.
El vértice 6 se halla afuera, de modo que encontramos y guardamos el punto de intersección.
Al utilizar los cinco punto que guardamos, repetiremos el proceso para la siguiente frontera de la ventana.

Implementar el algoritmo según se describió requiere establecer el almacenamiento para una lista de salida de vértices conforme un polígono se recorta contra cada frontera de ventana.
Podemos eliminar las listas intermedias de salida de vértices simplemente con recortar vértices individuales en cada paso y pasar los vértices recortados al algoritmo de recorte de la frontera siguiente.

Los polígonos convexos se recortan de manera correcta mediante el algoritmo de Sutherland-Hodgeman, pero los polígonos cóncavos se pueden desplegar con líneas ajenas, como se muestra en la siguiente figura:



Esto ocurre cuando el polígono recortado debe tener dos o mas secciones separadas.
Pero debido a que existe solo una lista de salidas de vértices, el ultimo vértice en la lista se une siempre al primer vértice.
Hay varias cosas que podríamos hacer para desplegar en forma correcta los polígonos cóncavos.
Para uno, podríamos dividir el polígono cóncavo en dos o mas polígonos convexos y procesar cada uno de estos por separado.
Otra posibilidad es modificar el planteamiento de Sutherland-Hodgeman para verificar la lista final de vértices para puntos de vértices múltiples a lo largo de cualquier frontera de ventana de recorte y unir de manera correcta los pares de vértices.
Por ultimo, podríamos utilizar un algoritmo de recorte de polígonos mas general, como el algoritmo de Weiler-Atherton, el cual se describe a continuación.

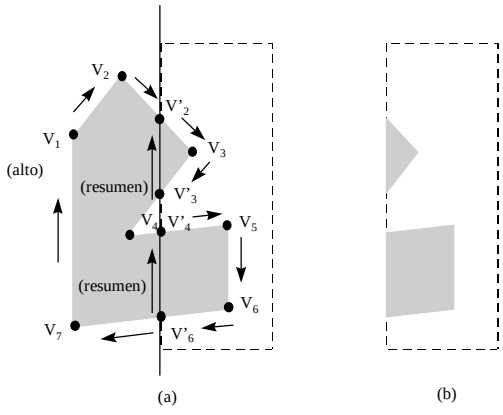
3.3.2 Recorte de Polígonos de Weiler-Atherton

Aquí, los procedimientos de procesamiento de vértices para las fronteras de ventana se modifican de modo que los polígonos cóncavos se despliegan en forma correcta.
Este procedimiento de recorte se desarrollo como un método para identificar las superficies visibles y, por tanto, se puede aplicar con regiones arbitrarias de recorte de polígonos.

La idea básica en este algoritmo es que en lugar de procesar siempre alrededor de las aristas del polígono como se procesan los vértices, en ocasiones deseamos seguir las fronteras de la ventana.
La trayectoria que seguimos depende de la dirección del procesamiento del polígono y de si el par de vértices del polígono que se presenta en ese momento representa un par del exterior al interior o un par del interior al exterior.
Para el procesamiento de vértices de polígonos en sentido del reloj, utilizamos las siguientes reglas:

- Para un par de vértices del exterior al interior, siga la frontera del polígono.
- Para un par de vértices del interior al exterior, siga la frontera de la ventana en la dirección del reloj.

En la siguiente figura se muestra la dirección del procesamiento en el algoritmo de Weiler-Atherton y el polígono recortado que resulta para un ventana de recorte rectangular:



3.4 Recorte de Curvas

Las áreas con fronteras curvas se pueden recortar con métodos similares a aquellos anteriores.
Sin embargo, los procedimientos de recorte de curvas comprenden ecuaciones no lineales y esto requiere mas procesamiento que para los objetos que tienen fronteras lineales.

El rectángulo de entrelazado para una circunferencia u otro objeto curvo se puede utilizar primero para probar la superposición con una ventana de recorte rectangular.
Si el rectángulo de entrelazado para el objeto esta por completo adentro de la ventana, guardamos el objeto.

Si se determina que el rectángulo esta por completo fuera de la ventana, eliminamos el objeto.
En cualquier caso, no es necesario ningún otro calculo.
Pero si la prueba del rectángulo de entrelazado fracasa, podemos buscar otros planteamientos que reducen los cálculos.
Para un circunferencia, podemos utilizar las extensiones de las coordenadas de los cuadrantes individuales y después octantes para la prueba preliminar antes de calcular las intersecciones de la ventana con la curva.
Para una elipse, podemos probar las extensiones de las coordenadas de los cuadrantes individuales.

Se pueden aplicar procedimientos similares cuando se recorta un objeto curvo contra una región de recorte de polígono general.
En el primer paso, podemos recortar el rectángulo de entrelazado del objeto contra el rectángulo de entrelazado de la región de recorte.
Si las dos regiones se superponen, será necesario resolver las ecuaciones simultáneas de líneas y curvas para obtener los puntos de intersección del recorte.