Introducción a la Computación Gráfica **(1316)**

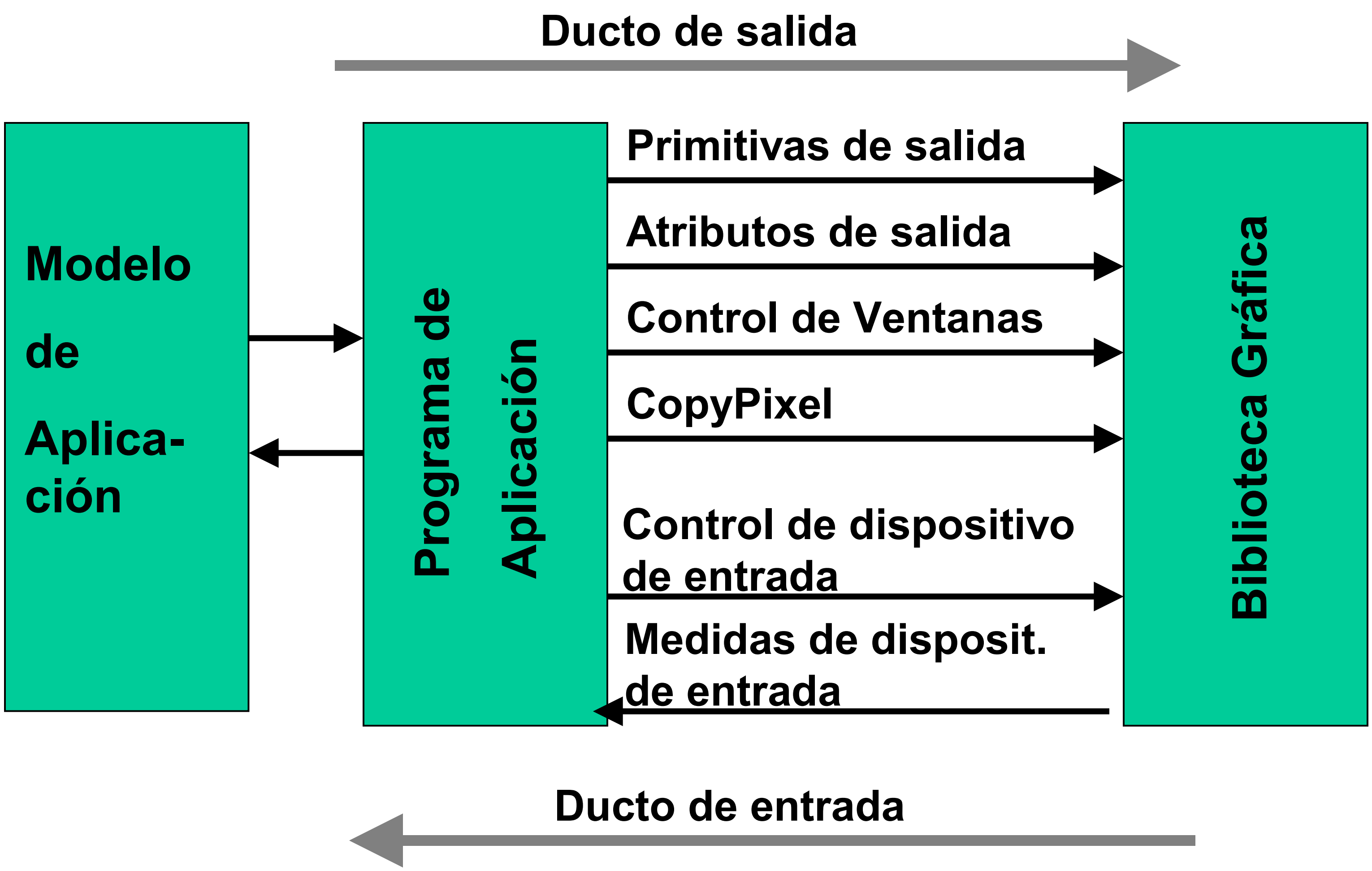
Algoritmos básicos de gráficos de barrido para dibujar primitivas bidimensionales

Capítulo 3 del libro:

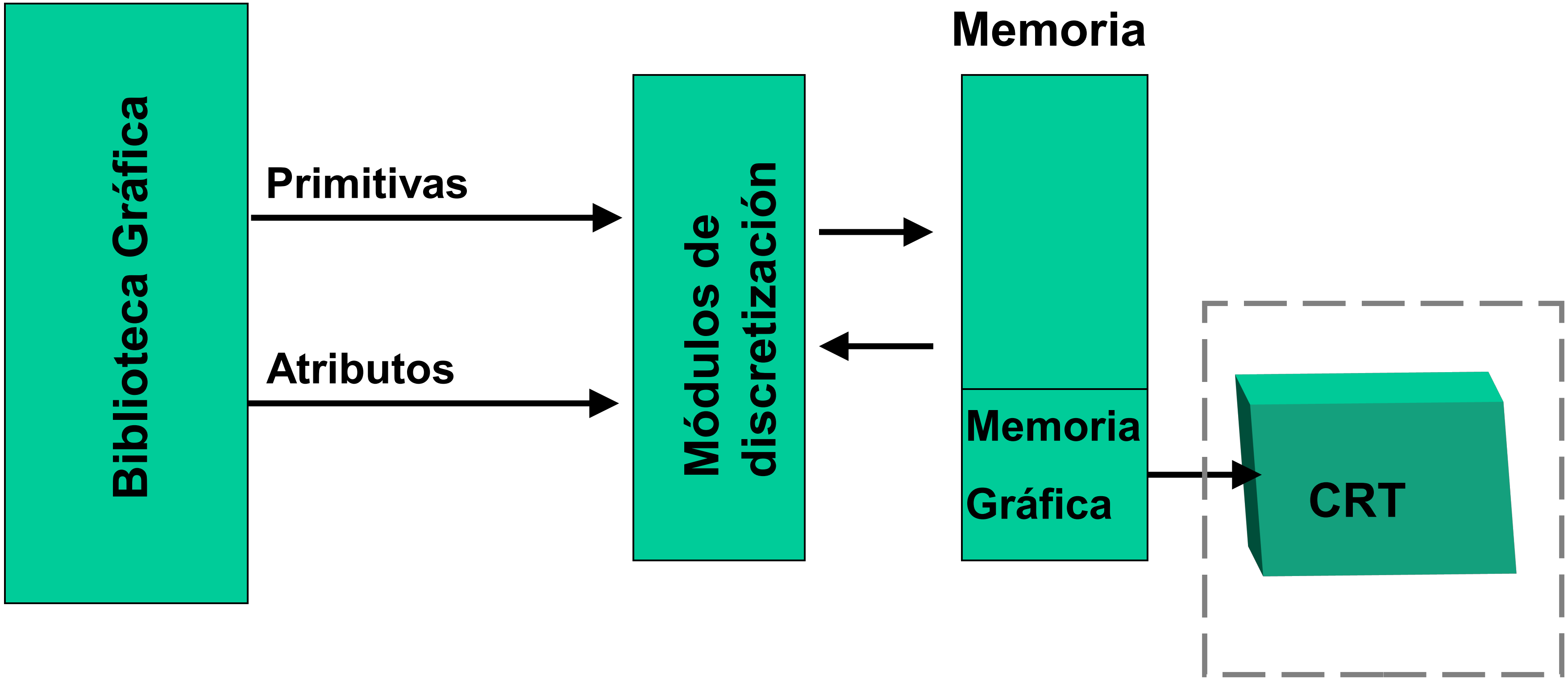
Introducción a la Graficación por Computador

Foley – Van Dam – Feiner – Hughes - Phillips

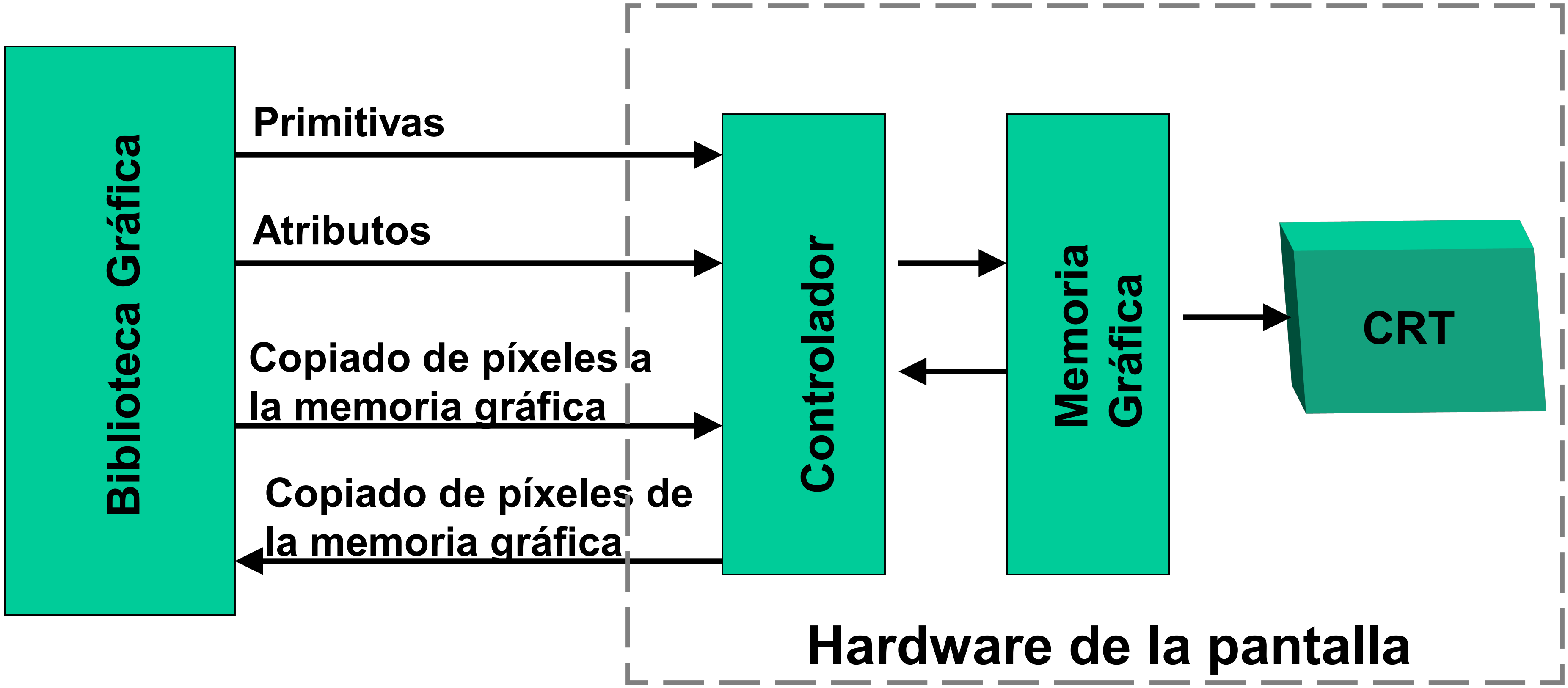
Esquema General (I)



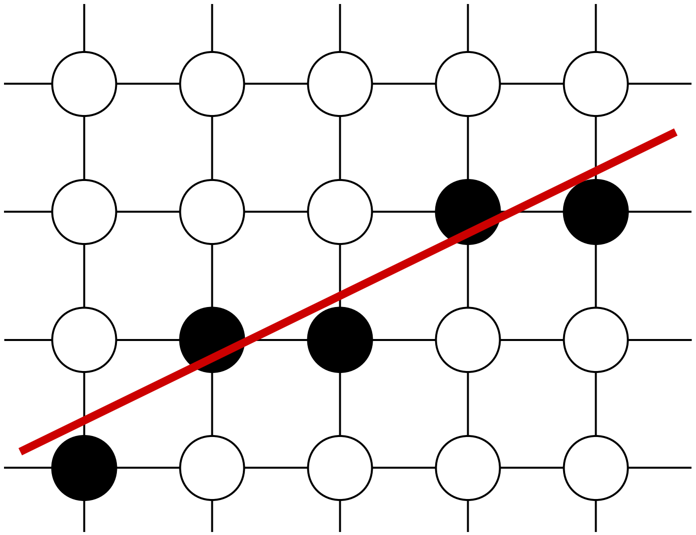
Esquema General (II)



Esquema General (III)



Discretización de líneas



Se considerarán solamente líneas de pendiente m <= 1

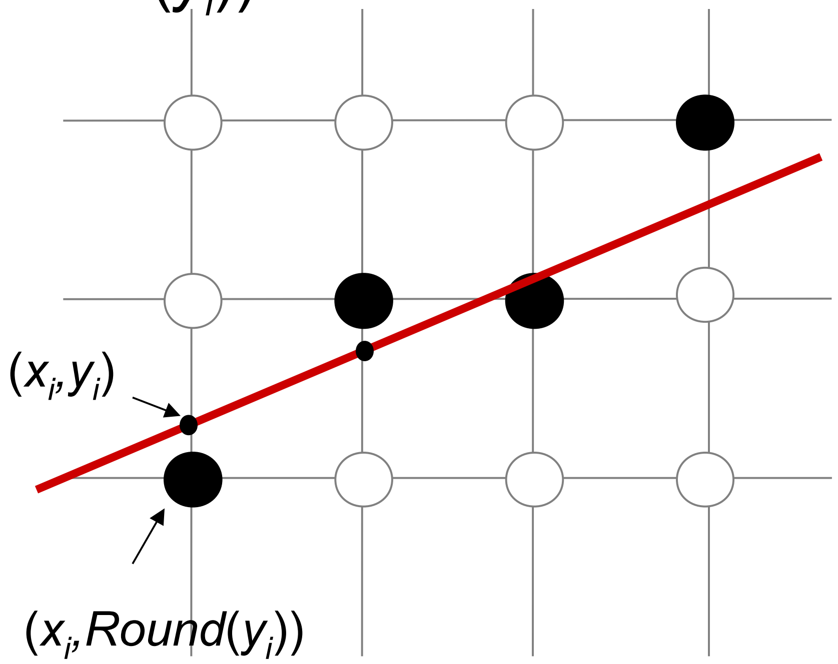
y líneas de un píxel de grosor.

Algoritmo incremental básico

$y_i = mx_i + B \quad \forall i$

y luego pintar el píxel $(x_i, round(y_i))$

¿Problemas?

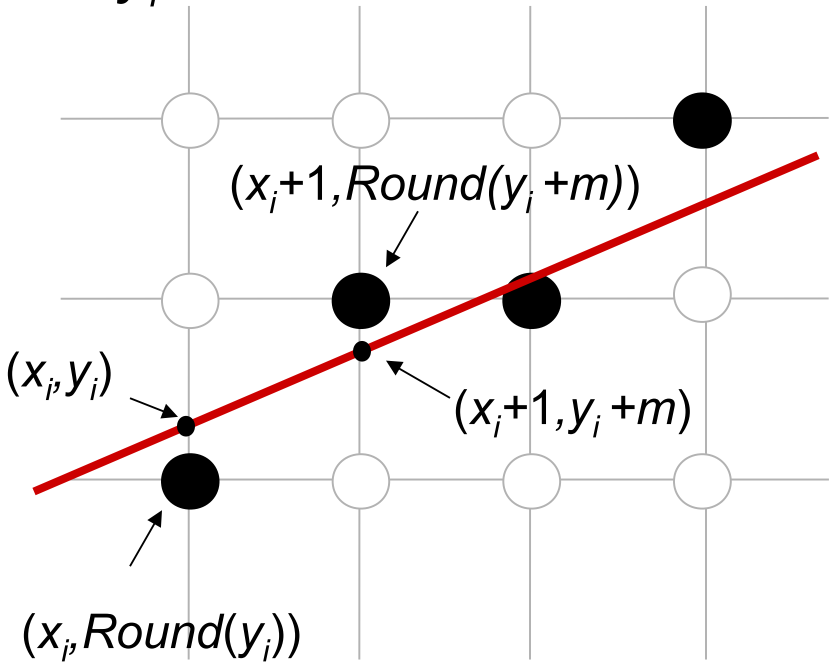


Algoritmo incremental básico

$y_i = mx_i + B \quad \forall i$

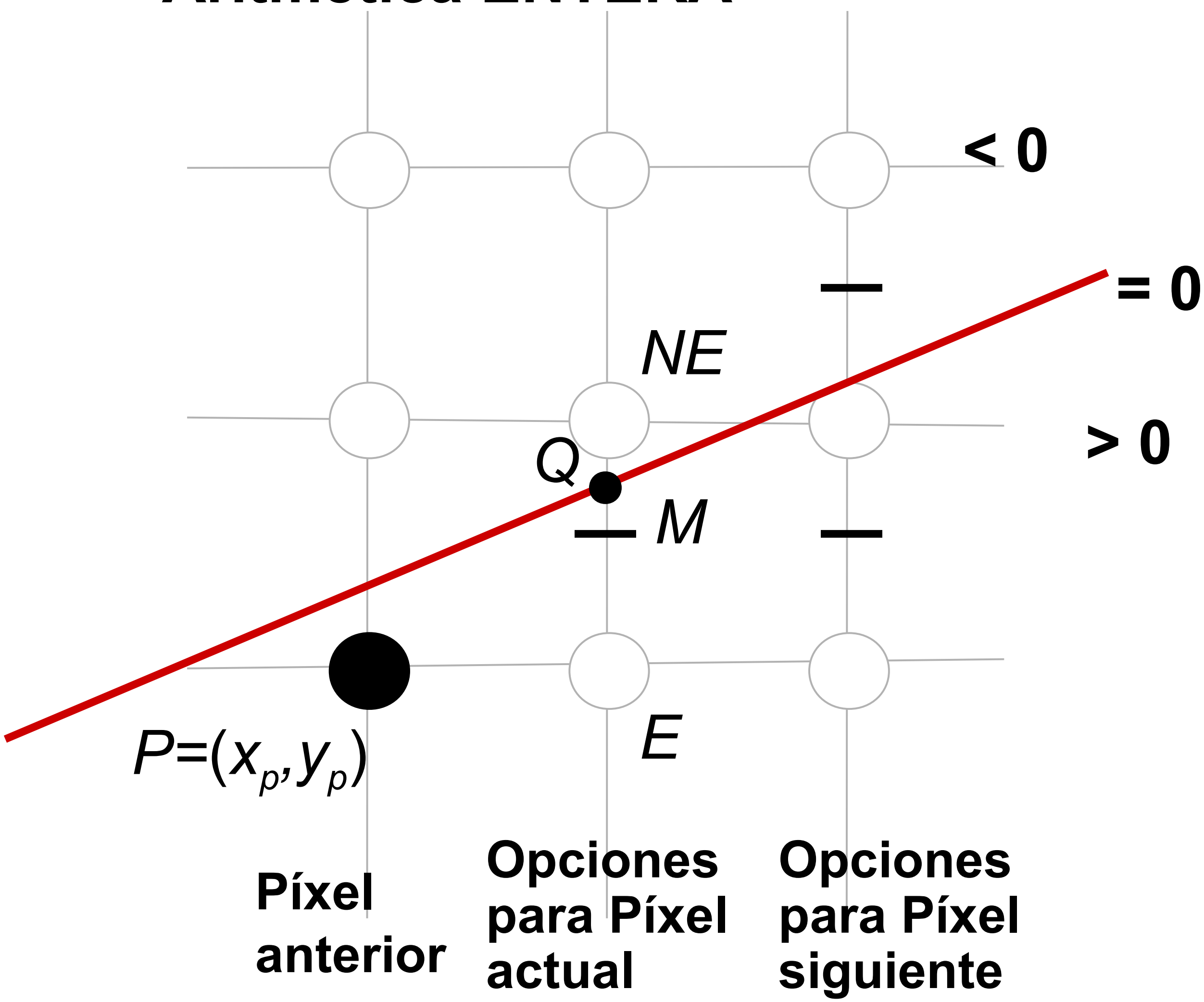
$y_{i+1} = mx_{i+1} + B = m(x_i + \Delta x) + B = y_i + m\Delta x$

Si $\Delta x = 1 \Rightarrow y_{i+1} = y_i + m$



Algoritmo de línea de punto medio

Aritmética ENTERA



Algoritmo de línea de punto medio

La representación explícita de la recta:

$$y=(dy/dx)\cdot x + B$$

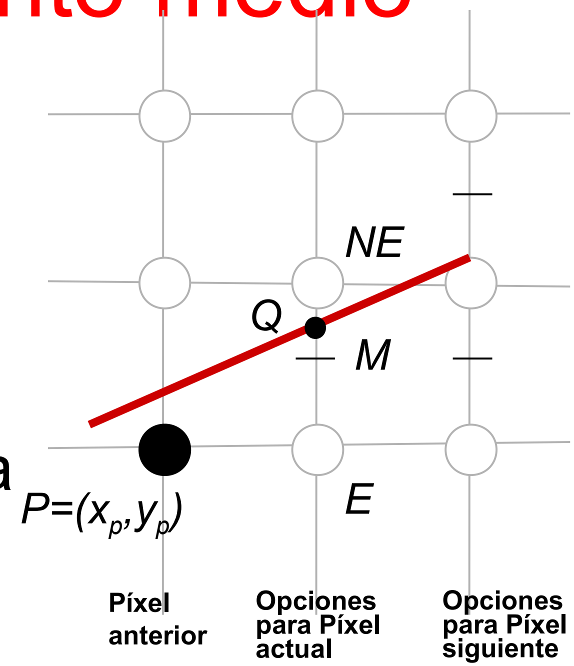
Se transforma en otra implícita:

$$F(x,y)= a\cdot x + b\cdot y + c = dy\cdot x - dx\cdot y + B\cdot dx = 0$$

Donde:

$$a=dy \; ; \; b=-dx \; ; \; c= B\cdot dx$$

Algoritmo de línea de punto medio



Propiedades de F(x,y)

- F(x,y)=0 en la línea
- F(x,y)>0 en los puntos debajo de la línea
- F(x,y)<0 en los puntos sobre la línea

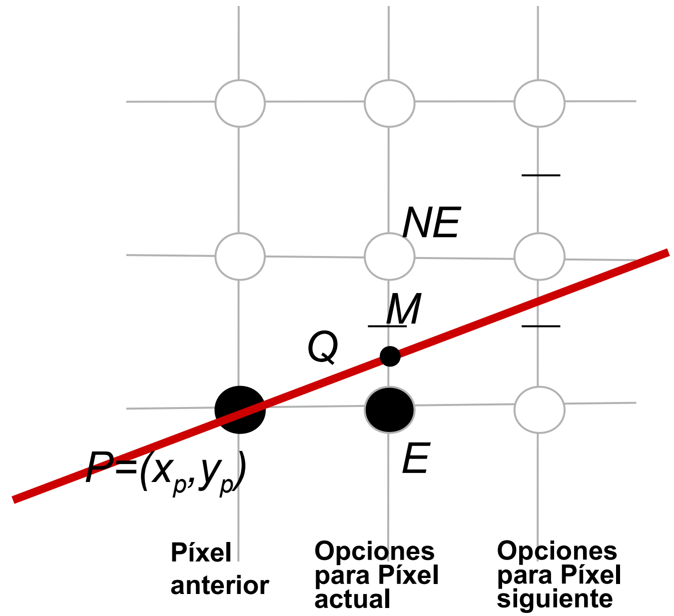
$\Rightarrow$ Aplicamos F al punto M ; $d = F(M) = F(x_p+1, y_p+½)$

Si $d > 0$ $\Rightarrow$ se elige el píxel *NE*

Si $d <= 0$ $\Rightarrow$ se elige el píxel *E*

Algoritmo de línea de punto medio

$d = a(x_p + 1) + b(y_p + ½) + c$



Llamemos d_{viejo} a este d

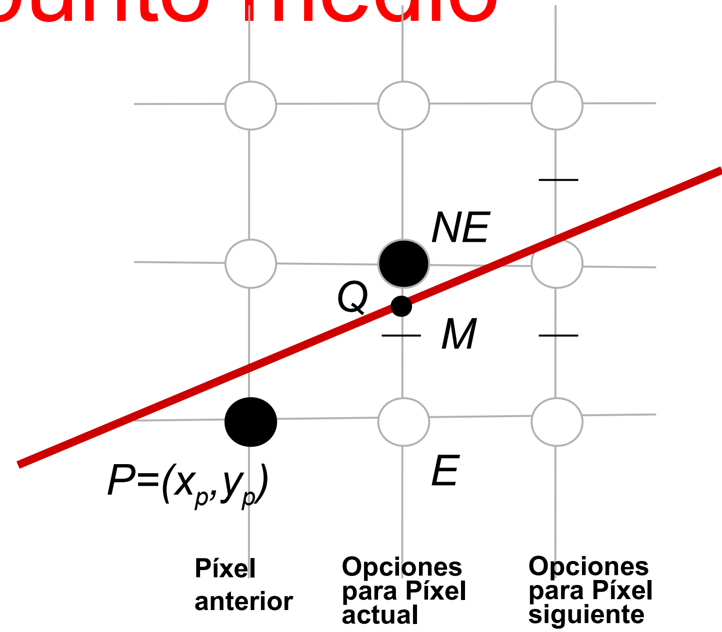
Si se eligió el píxel E, ¿cuál es el valor d_{nuevo}?

$d_{nuevo} = a(x_p + 2) + b(y_p + ½) + c$

$$\mathbf{d_{nuevo} = d_{viejo} + a = d_{viejo} + dy = d_{viejo} + \Delta_E}$$

Algoritmo de línea de punto medio

$d_{viejo} = a(x_p + 1) + b(y_p + ½) + c$



Si se eligió el píxel *NE*, ¿cuál es el valor d_{nuevo}?

$d_{nuevo} = a(x_p + 2) + b(y_p + {}^{3}/_{2}) + c$

$\mathbf{d_{nuevo} = d_{viejo} + (a+b) = d_{viejo} + (dy\text{-}dx) = d_{viejo} + \Delta_{NE}}$

Repaso: **Algoritmo de línea de punto medio**

Quiero dibujar el segmento entre (x_0,y_0) y (x_{fin},y_{fin}) que tienen coordenadas enteras => la recta tiene coeficientes enteros.

El primer punto medio está en

$F(x_0+1,y_0+½)=F(x_0,y_0)+a+{}^b/_2= a+{}^b/_2$

Dibujo (x_0,y_0)

Defino $d_{nuevo}= a + {}^b/_2$

p=0

Mientras no llegué a x_{fin}

Según el signo de d_{nuevo}

elijo y dibujo $(x_{p+1},y_{p+1}) \Rightarrow M \Rightarrow \Delta \Rightarrow d_{nuevo}$

$p=p+1$

Fin

Repaso: **Algoritmo de línea de punto medio**

El problema es que d_{inicio}= a + b/2, por lo que hay una fracción inicial que perjudica todos los cálculos posteriores.

¿Solución?: Multiplicar la función F por 2

F(x,y)=(ax + by + c)=2(ax + by + c)

Conclusiones:

Para cada paso, d_{nuevo} se calcula a partir de una suma entera.

El algoritmo se puede generalizar para líneas con pendientes que no estén entre 0 y 1.

Discretización de circunferencias

Discretización de circunferencias

Ecuación del círculo centrado en el origen:

$$X^2 + y^2 = R^2$$

Función explícita:

$$y=\pm sqrt(R^2-x^2)$$

y al discretizar queda:

$$y=round(\pm sqrt(R^2-x^2))$$

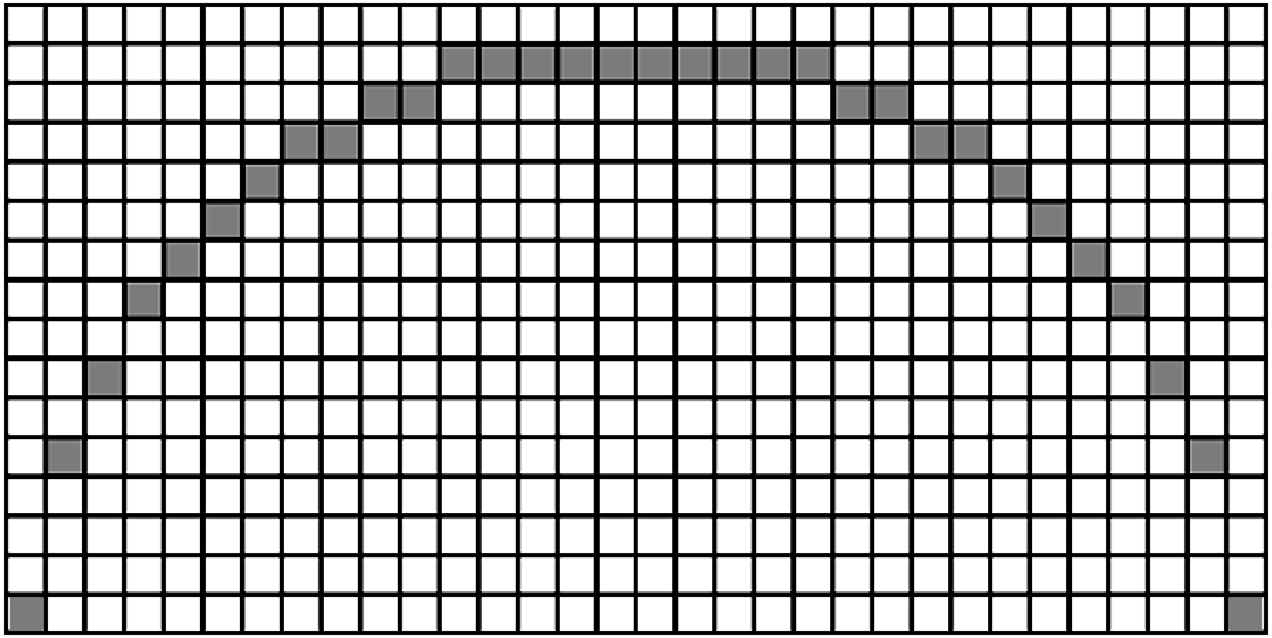
¿Problemas?

Problemas en la forma tradicional de discretización de circunferencias

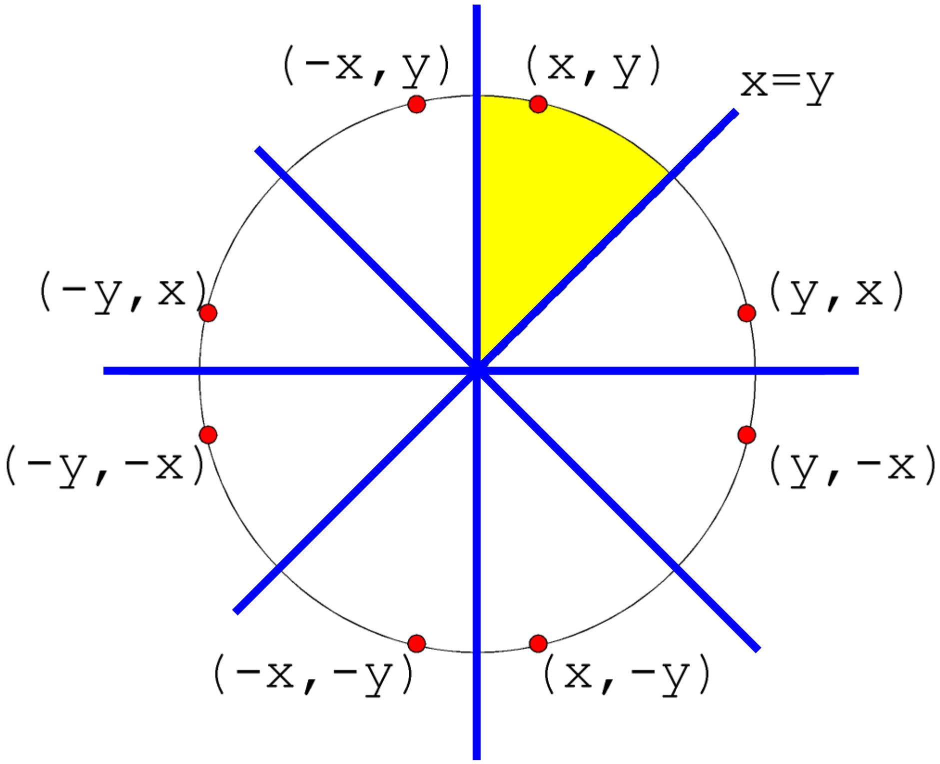
Precisa utilizar raíz cuadrada y redondeo

Uso de punto flotante

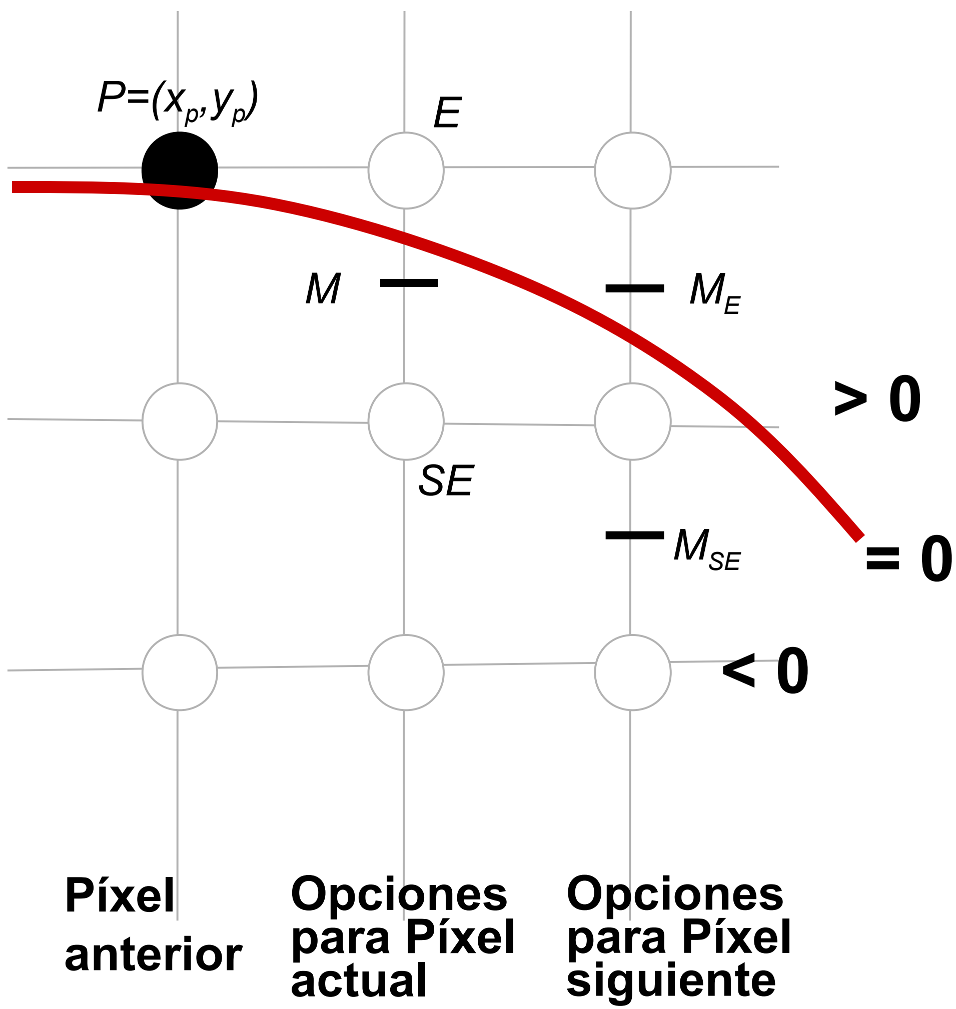
Dibujo de la circunferencia con huecos



Simetría de 8 lados



Algoritmo de círculo de punto medio



Algoritmo de círculo de punto medio

Se considera solo 45° de un círculo, de x=0 a x=y=R/sqrt(2) .

$$d_{viejo} = F(x_p+1, y_p-½) = (x_p+1)^2 + (y_p-½)^2 - R^2 = 0$$

Si $d_{viejo} < 0$ se escoge E y luego el punto medio es M_E

$$d_{nuevo} = F(x_p+2, y_p-½) = (x_p+2)^2 + (y_p-½)^2 - R^2$$

$$\mathbf{d_{nuevo} = d_{viejo} + (2x_p + 3) \Rightarrow \Delta_E = (2x_p + 3)}$$

Si $d_{viejo} >= 0$ se escoge SE y luego el punto medio es M_{SE}

$$d_{nuevo} = F(x_p+2, y_p-{}^3/_2) = (x_p+2)^2 + (y_p-{}^3/_2)^2 - R^2$$

$$\mathbf{d_{nuevo} = d_{viejo} + (2x_p - 2y_p + 5) \Rightarrow \Delta_{SE} = (2x_p - 2y_p + 5)}$$

Algoritmo de círculo de punto medio

Δ_E y Δ_{SE} varían en cada paso (en el caso lineal son ctes.).

Las operaciones para el cálculo de los d_{nuevo} son enteras.

Falta ver cómo comienza el algoritmo.

Si se parte del punto $(0,R)$, con R entero

el siguiente punto medio es $(1,R-½)$, o sea:

$$F(1,R-½)=1+(R^2-R+¼)-R^2= {}^5/_4 - R = d_{viejo}$$

¿Problema?

Solución: pasar a $h=d - ¼ \Rightarrow h_{viejo} = 1-R$

Esto funciona porque h comienza y continúa con valores enteros

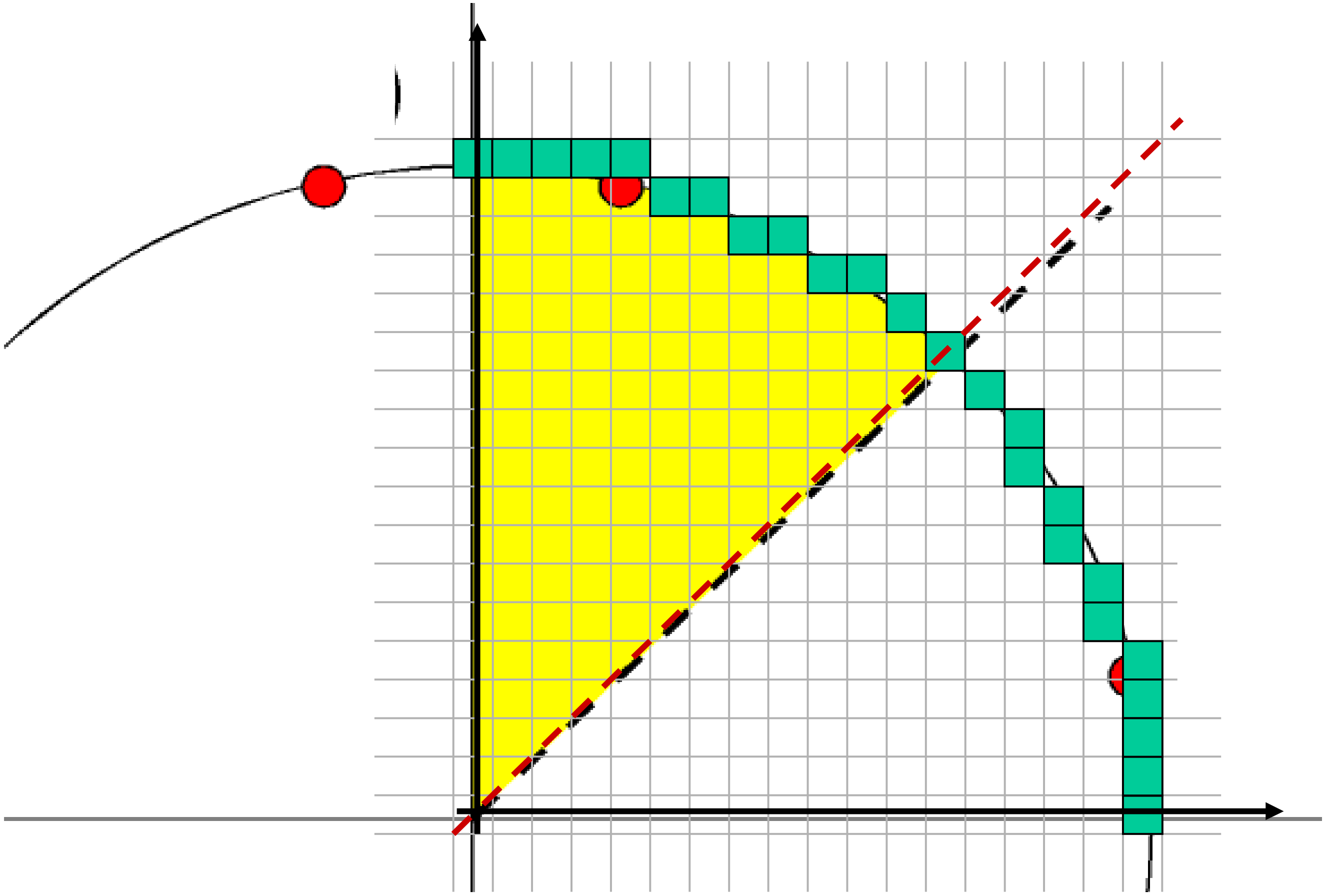
Algoritmo de círculo de punto medio

En lugar de preguntar por $d<0$ se pregunta por $h<-1/4$

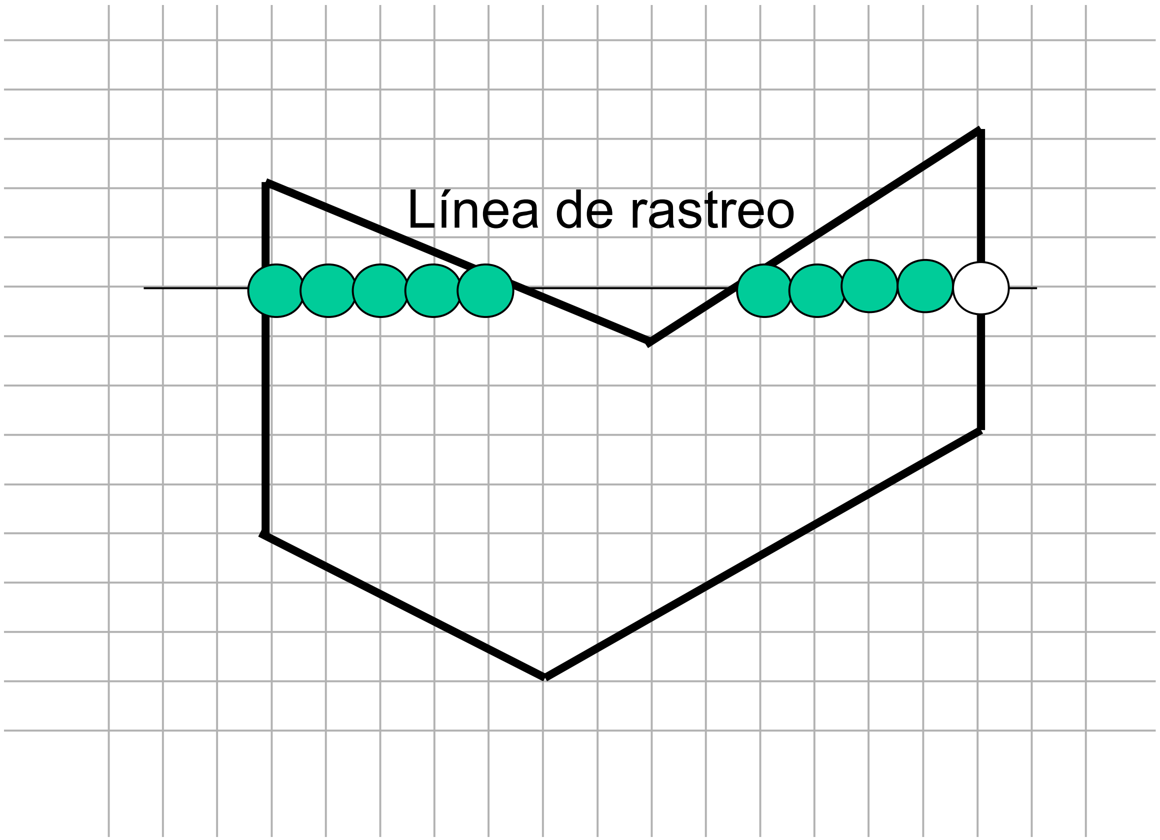
Como h se inicia con enteros y luego sigue con enteros, entonces resulta igual preguntar por $h<0$

Si un entero es menor que $-1/4$, también es menor que 0.

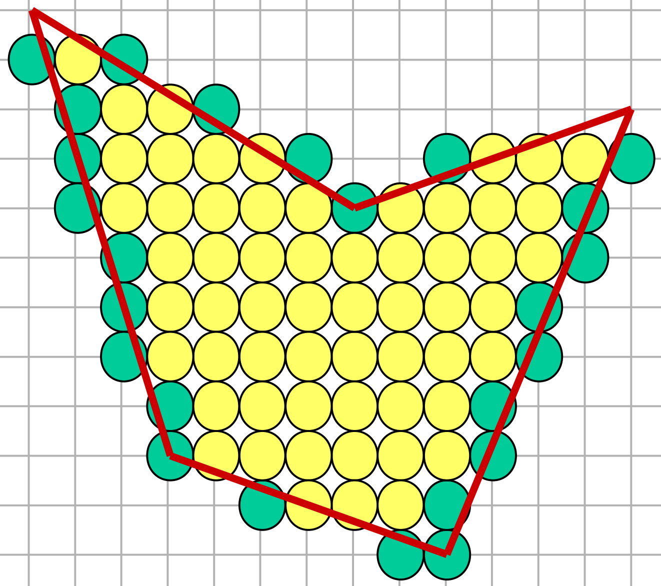
Algoritmo de círculo de punto medio



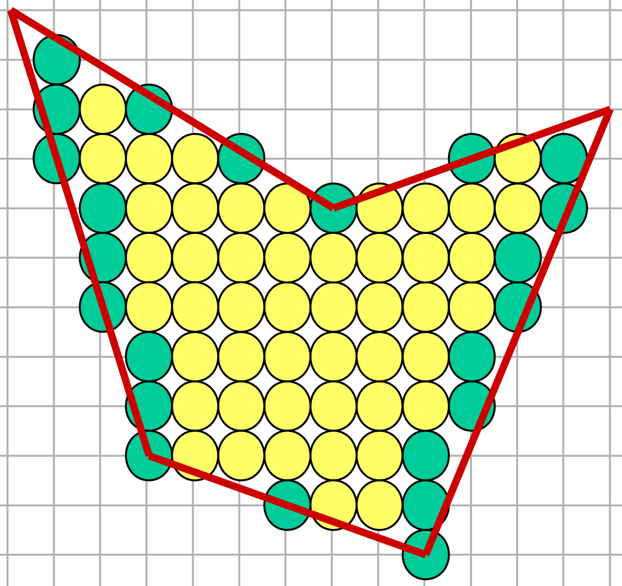
Rellenado de polígonos



Rellenado de polígonos



Se utiliza el algoritmo del punto medio



Solamente se dibujan los puntos interiores del polígono.

Rellenado de polígonos

1. Hallar las intersecciones de la línea de rastreo con todas las aristas del polígono

2. Ordenar las intersecciones aumentando la coordenada x

3. Colocar todos los píxeles entre pares de intersecciones que se encuentren dentro del polígono, utilizando la regla de paridad impar:

   La paridad es inicialmente par y cada intersección cambia la paridad; solo se dibuja cuando la paridad es impar.

Preguntas a responder para realizar el paso 3

Colocar todos los píxeles entre pares de intersecciones que se encuentren dentro del polígono, utilizando la regla de paridad impar

3.1 Dada una intersección con un valor X arbitrario y fraccionario, ¿cómo determinamos cuál de los dos píxeles a cada lado de la intersección es el interior?

3.2 ¿Cómo tratamos el caso especial de las intersecciones en coordenadas enteras de los píxeles?

3.3 ¿Cómo tratamos el caso especial del paso 3.2 para vértices compartidos?

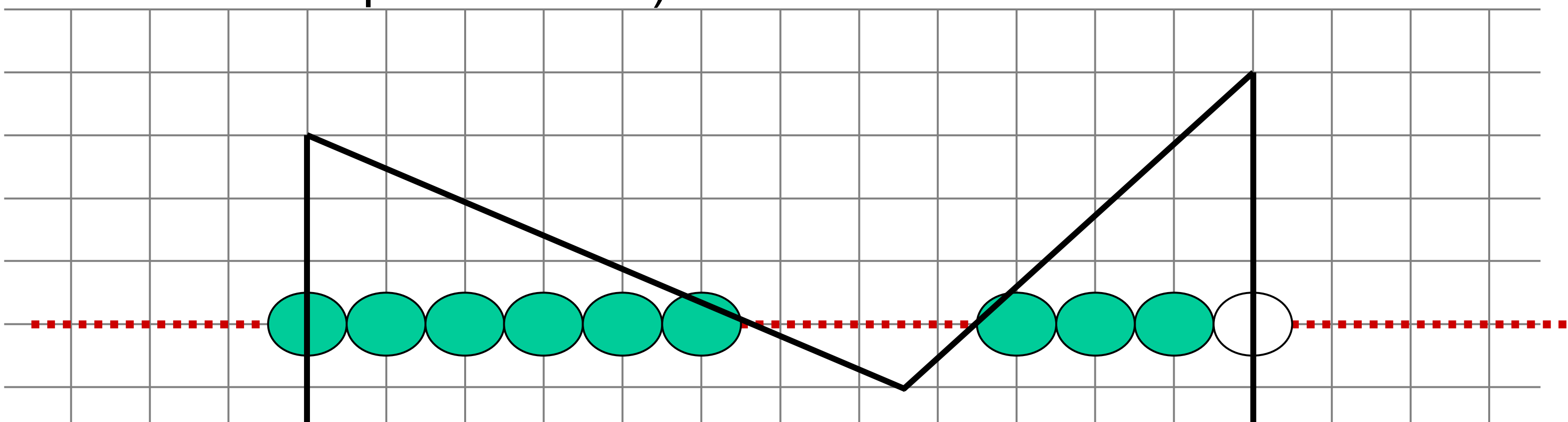
3.4 ¿Cómo tratamos el caso especial del paso 3.2 si los vértices definen una arista horizontal?

Preguntas a responder para realizar el paso 3

3.1 Dada una intersección con un valor x arbitrario y fraccionario, ¿cómo determinamos cuál de los dos píxeles a cada lado de la intersección es el interior?

Solución:

Si nos movemos hacia la derecha a una intersección fraccionaria, y estamos dentro del polígono, redondeamos hacia abajo en la coordenada x de la intersección (definiendo el píxel interior). Si estamos fuera, redondeamos hacia arriba la coordenada x (definiendo también un píxel interior).

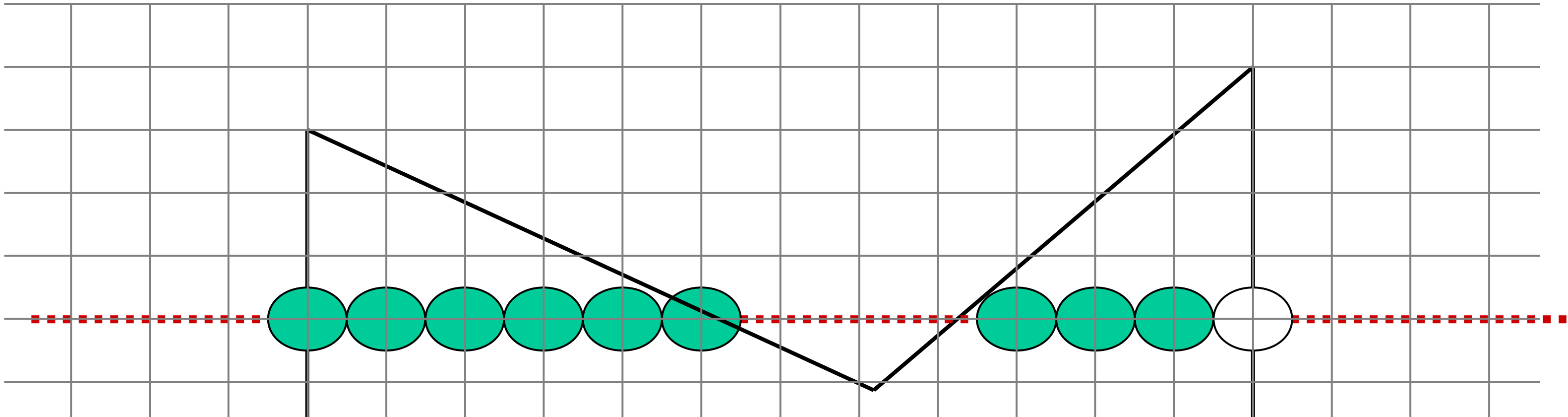


Preguntas a responder para realizar el paso 3

3.2 ¿Cómo tratamos el caso especial de las intersecciones en coordenadas enteras de los píxeles?

Solución:

Si el píxel del extremo izquierdo de un tramo tiene coordenada x entera, lo definimos como interior. Si el píxel de extremo derecho tiene coordenada x entera, lo definimos como exterior.

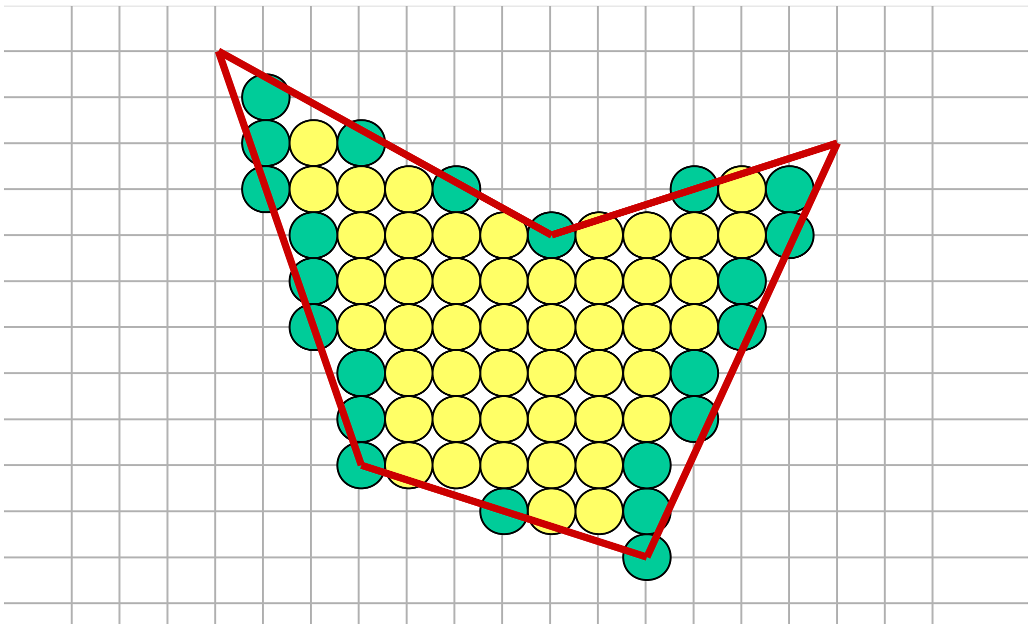


Preguntas a responder para realizar el paso 3

3.3 ¿Cómo tratamos el caso especial del paso 3.2 para vértices compartidos?

Solución:

En el cálculo de paridad, contamos solo los vértices y_{min} de las aristas y no los vértices y_{max}.

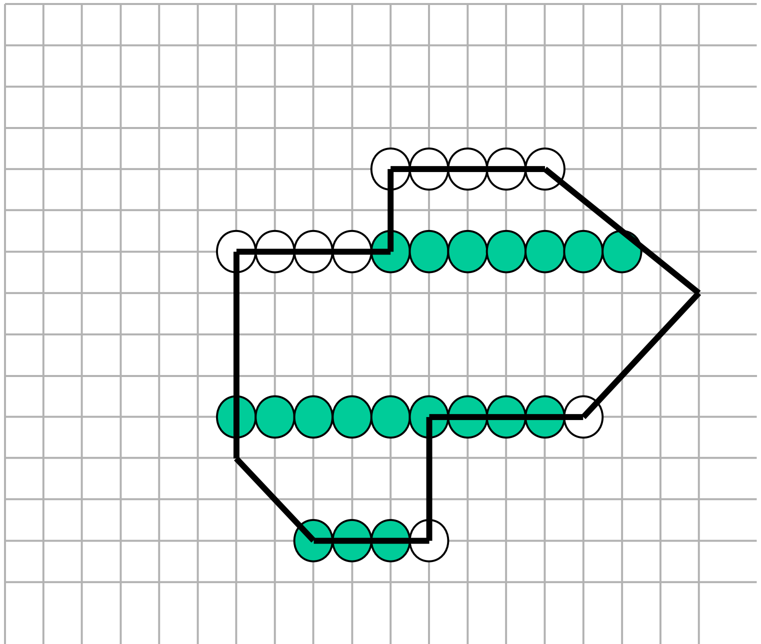


Preguntas a responder para realizar el paso 3

3.4 ¿Cómo tratamos el caso especial del paso 3.2 si los vértices definen una arista horizontal

Solución:

No se cuentan los vértices de las aristas horizontales.
No son ni y_{min} ni y_{max}.

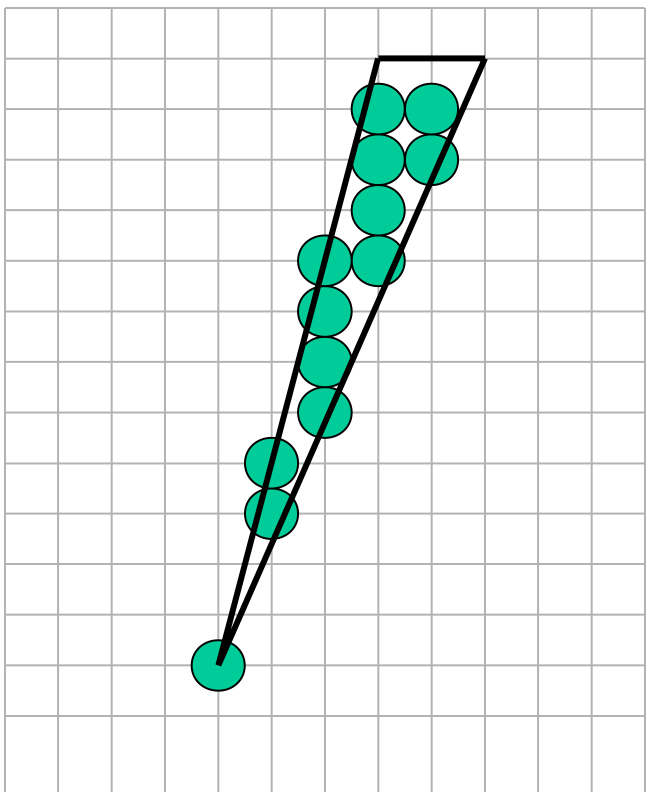


Astillas

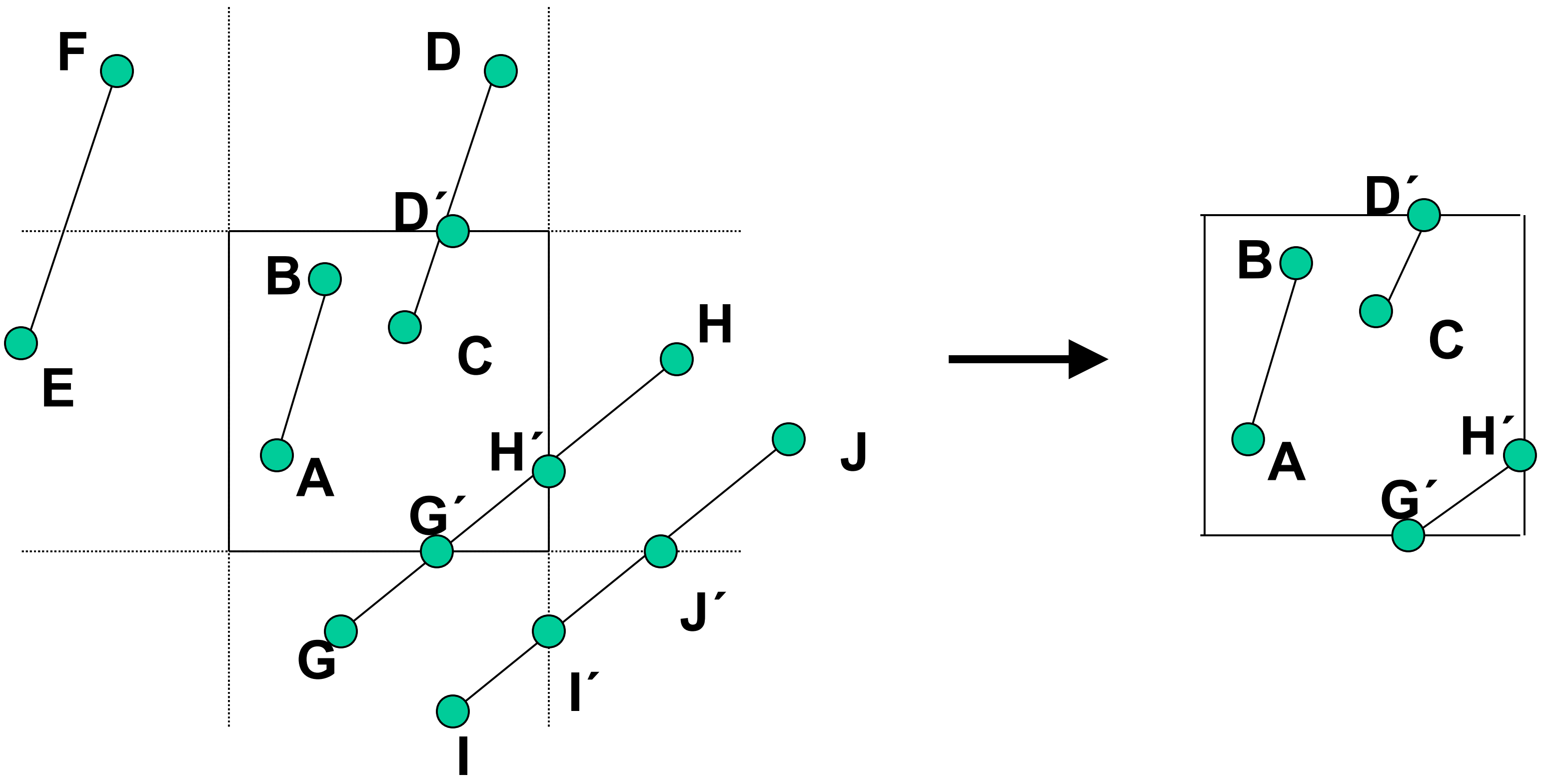
Cuando hay polígonos con aristas tan cercanas que crean una astilla.

Pueden haber líneas de rastreo sin pintar.

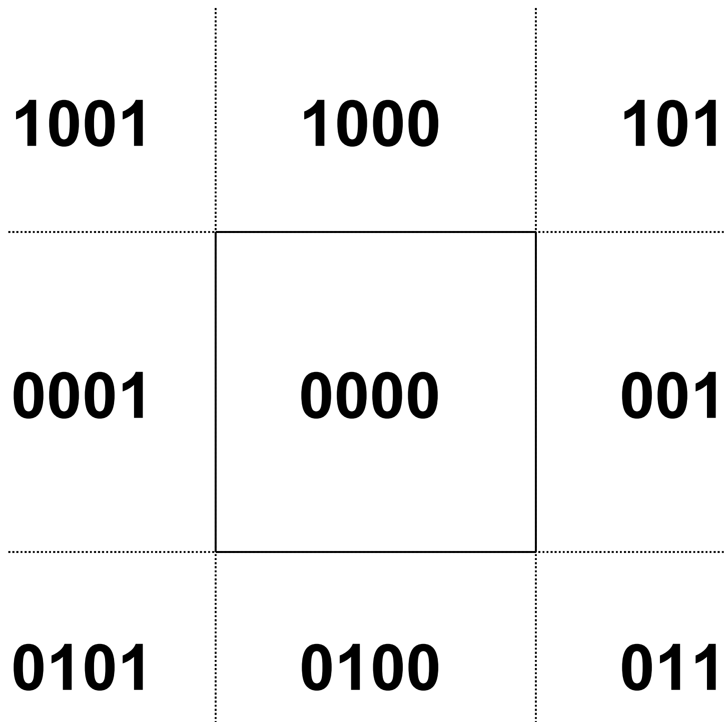
Se puede solucionar si en lugar de 2 valores posibles para el píxel, se permiten valores de intensidad que varíen como función de la distancia entre el centro del píxel y la primitiva.



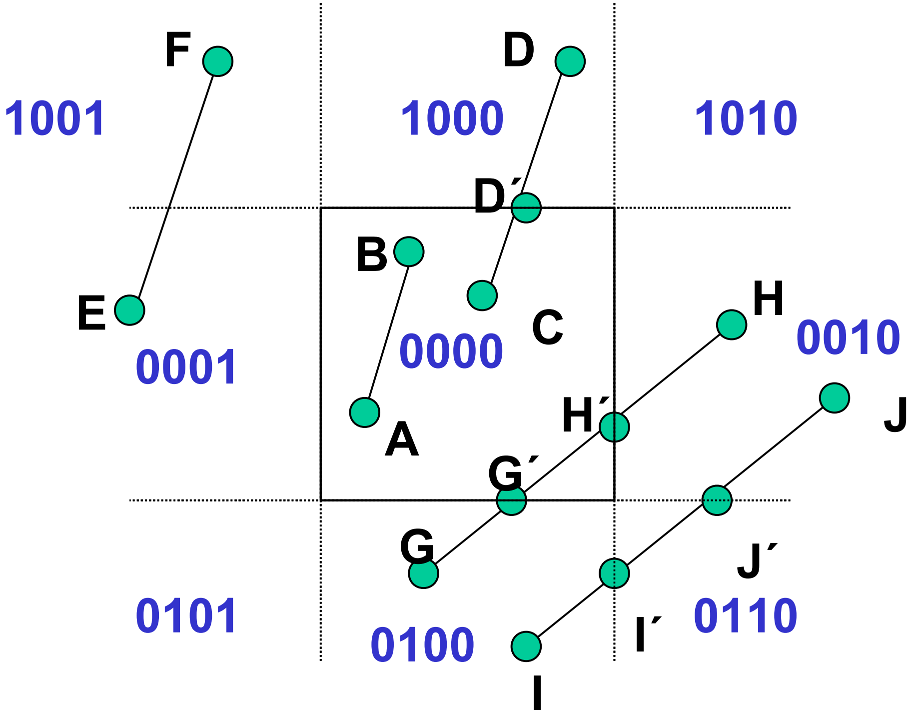
Algoritmo de recorte de líneas de Cohen-Sutherland



Asignación de códigos de región



Algoritmo de recorte de líneas de Cohen-Sutherland



E & F = 0001 & 1001 = 0001

C & D = 0000 & 1000 = 0000

A & B = 0000 & 0000 = 0000

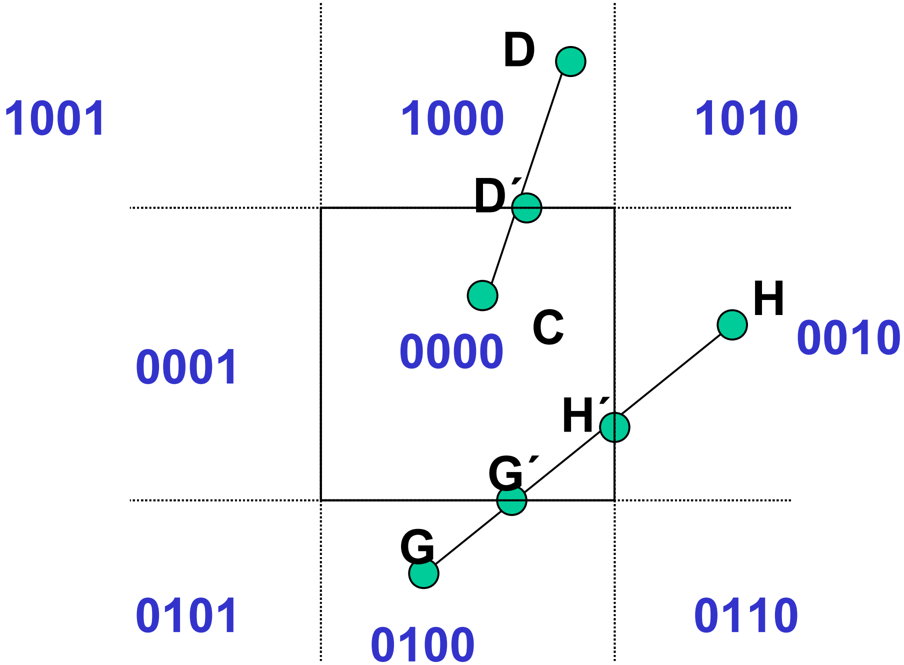
G & H = 0100 & 0010 = 0000

I & J = 0100 & 0010 = 0000

Se asigna un número a cada extremo de un segmento y se le aplica la operación lógica AND (bit a bit).

Si da $\neq$ 0000 $\Rightarrow$ se rechaza la línea trivialmente

Algoritmo de recorte de líneas de Cohen-Sutherland

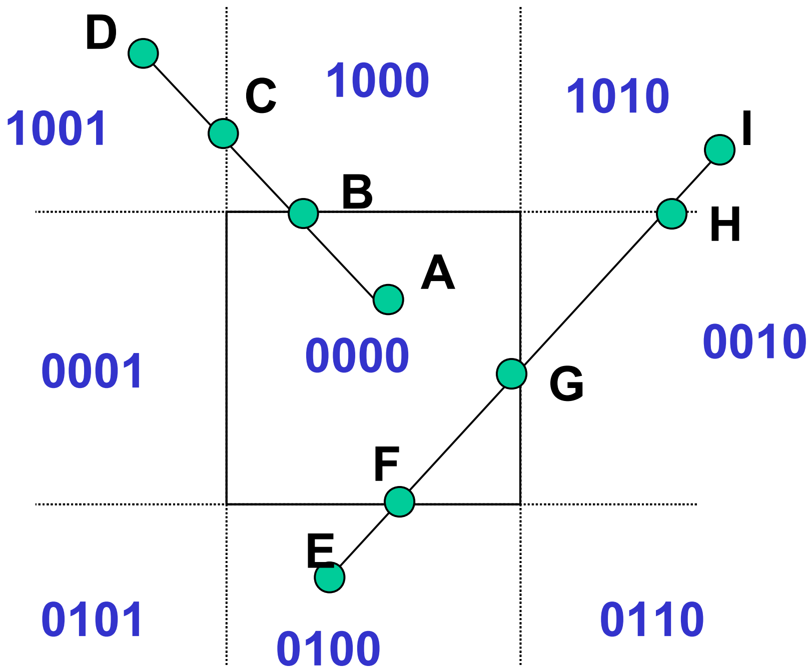


Si no se puede rechazar trivialmente la línea, se la divide en 2 segmentos / uno o ambos se puedan descartar.

Se divide considerando las aristas que crucen la línea. Eso se conoce a través de los códigos de los puntos extremos.

El orden de las aristas es arriba, abajo, derecha, izquierda.

Algoritmo de recorte de líneas de Cohen-Sutherland

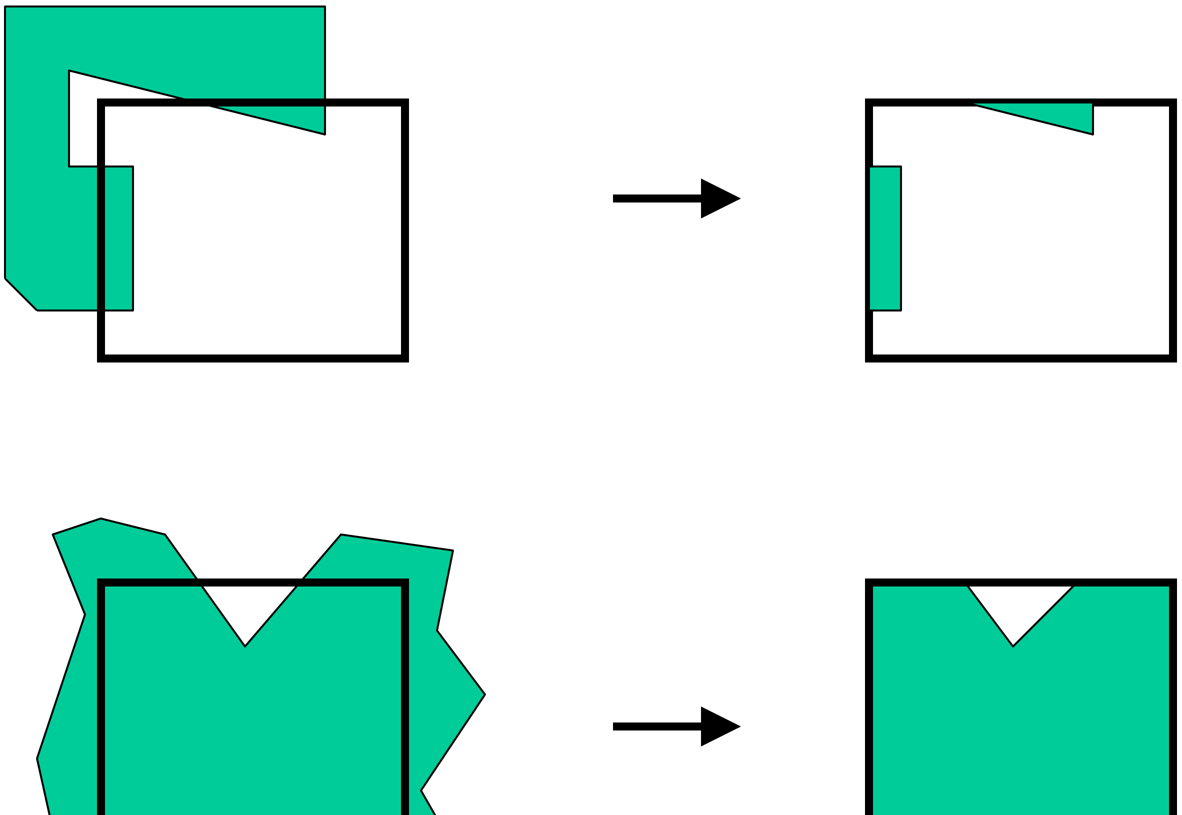


Si no se puede rechazar trivialmente la línea, se la divide en 2 segmentos / uno o ambos se puedan descartar.

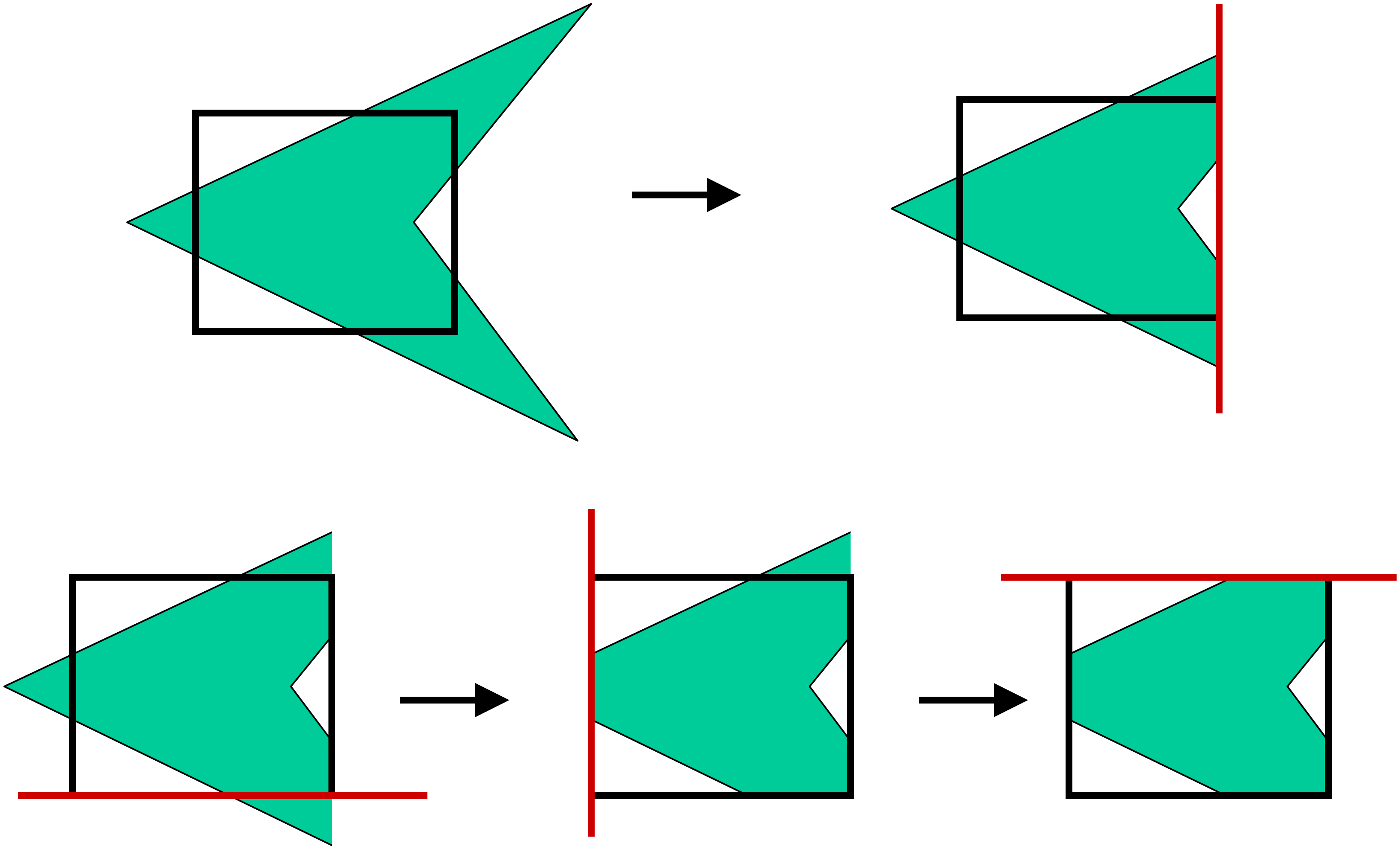
AD → DB y BA
Ignoramos DB y a B le asignamos 0000

IE → IH y HE. Ignoramos IH. H → 0010
HE → HF y FE. Ignoramos FE. F →0000
HF → HG y GF. Ignoramos HG. G →0000

Recorte de Polígonos

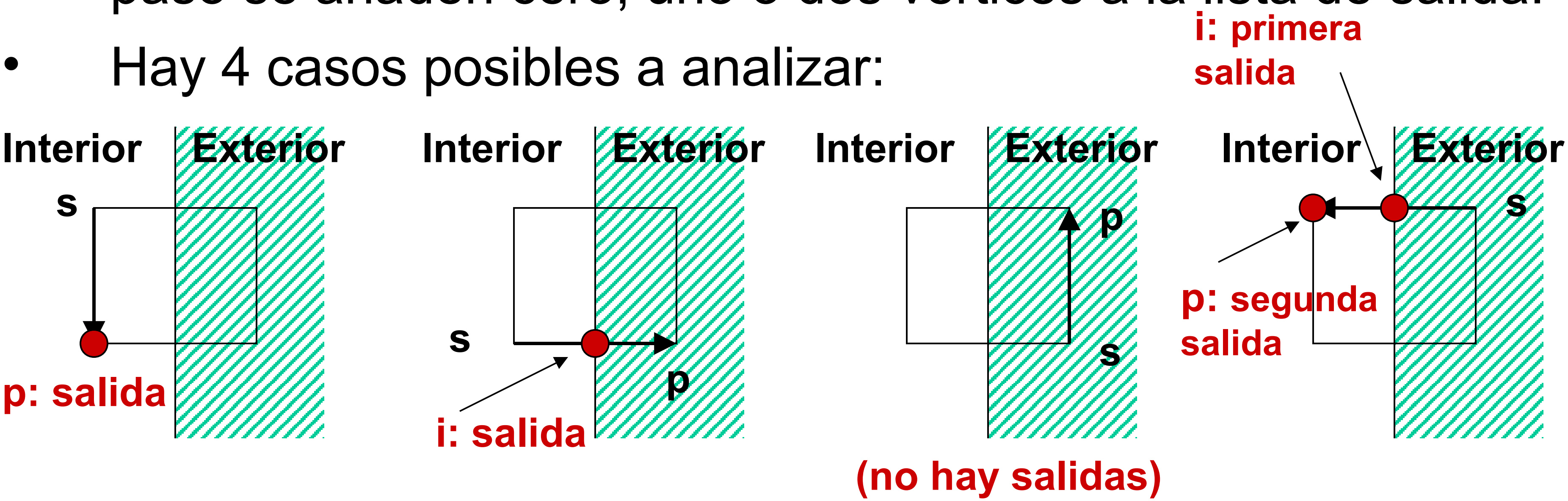


Algoritmo de Sutherland-Hodgman



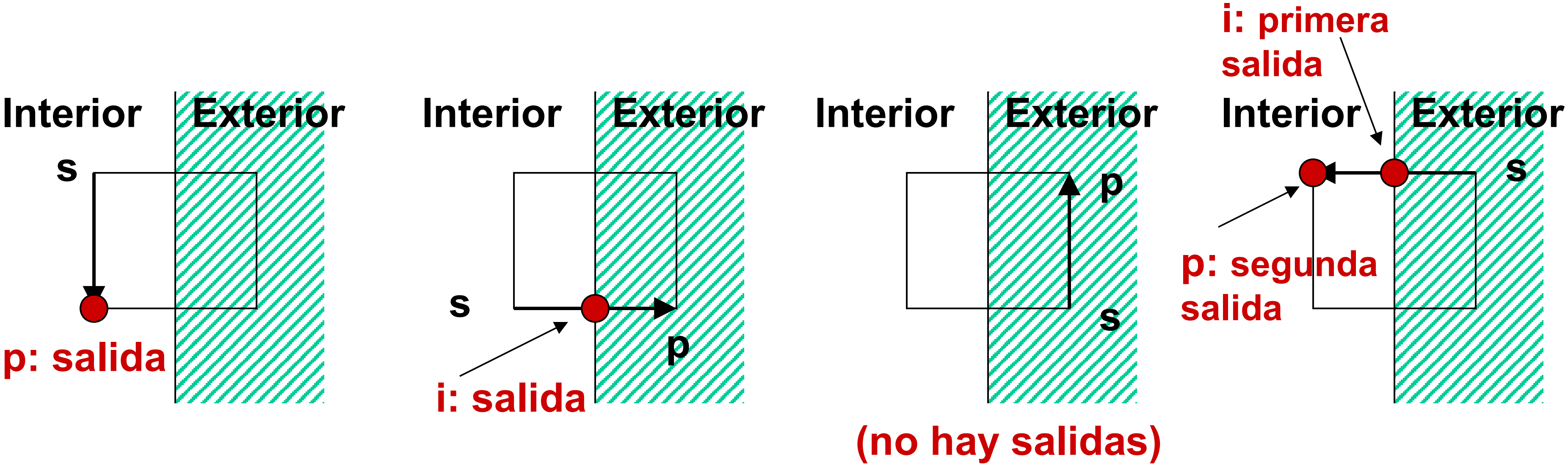
Algoritmo de Sutherland-Hodgman

- El polígono se compone de una serie de vértices $v_1, v_2, \ldots, v_n$
- Las aristas se forman de v_i a v_{i+1} y de v_n a v_1
- El algoritmo recorta el polígono con respecto a una sola arista de la ventana por vez.
- El algoritmo recorre el polígono arista por arista. En cada paso se añaden cero, uno o dos vértices a la lista de salida.
- Hay 4 casos posibles a analizar:



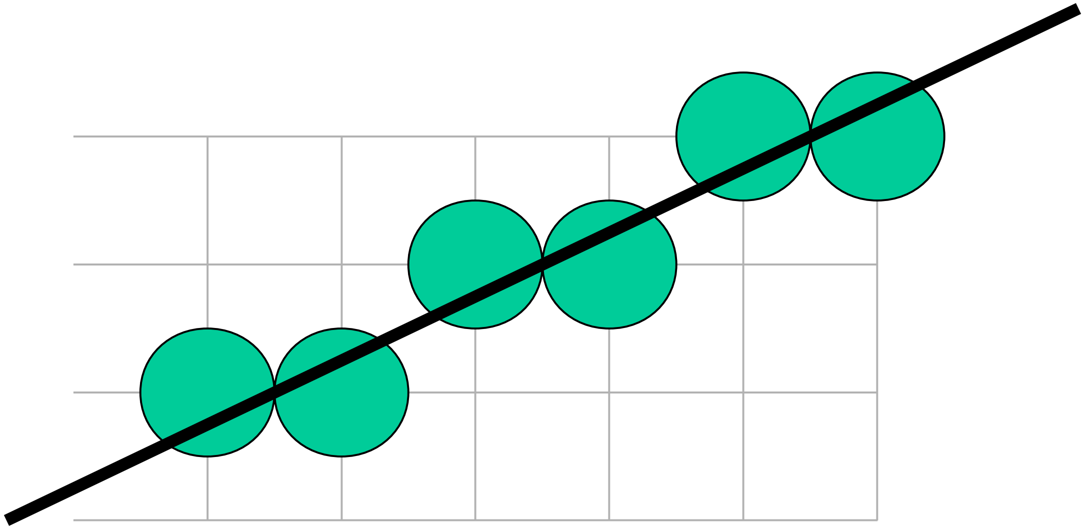
Algoritmo de Sutherland-Hodgman

1. La arista está completamente dentro de las fronteras de recorte ⇒ se agrega el vértice *p* a la lista de salida.
2. Se agrega el punto de intersección *i* con la frontera se agrega a la lista de salida.
3. Ambos vértices se hallan fuera de las fronteras, por lo que no hay salida.
4. El punto de intersección *i* y el vértice *p* se añaden a la lista de salida.

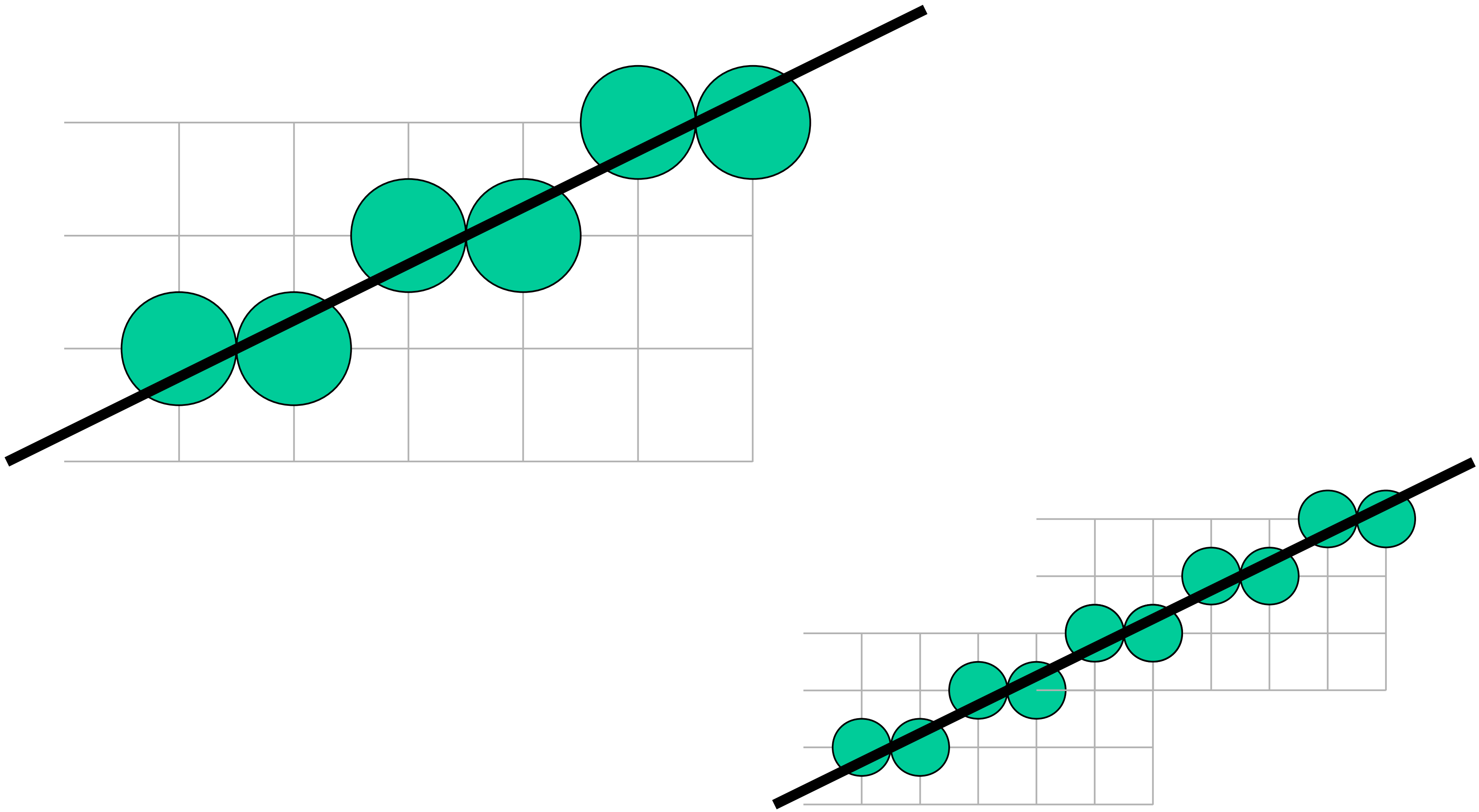


Eliminación de Artefactos de Discretización (*Antialiasing*)

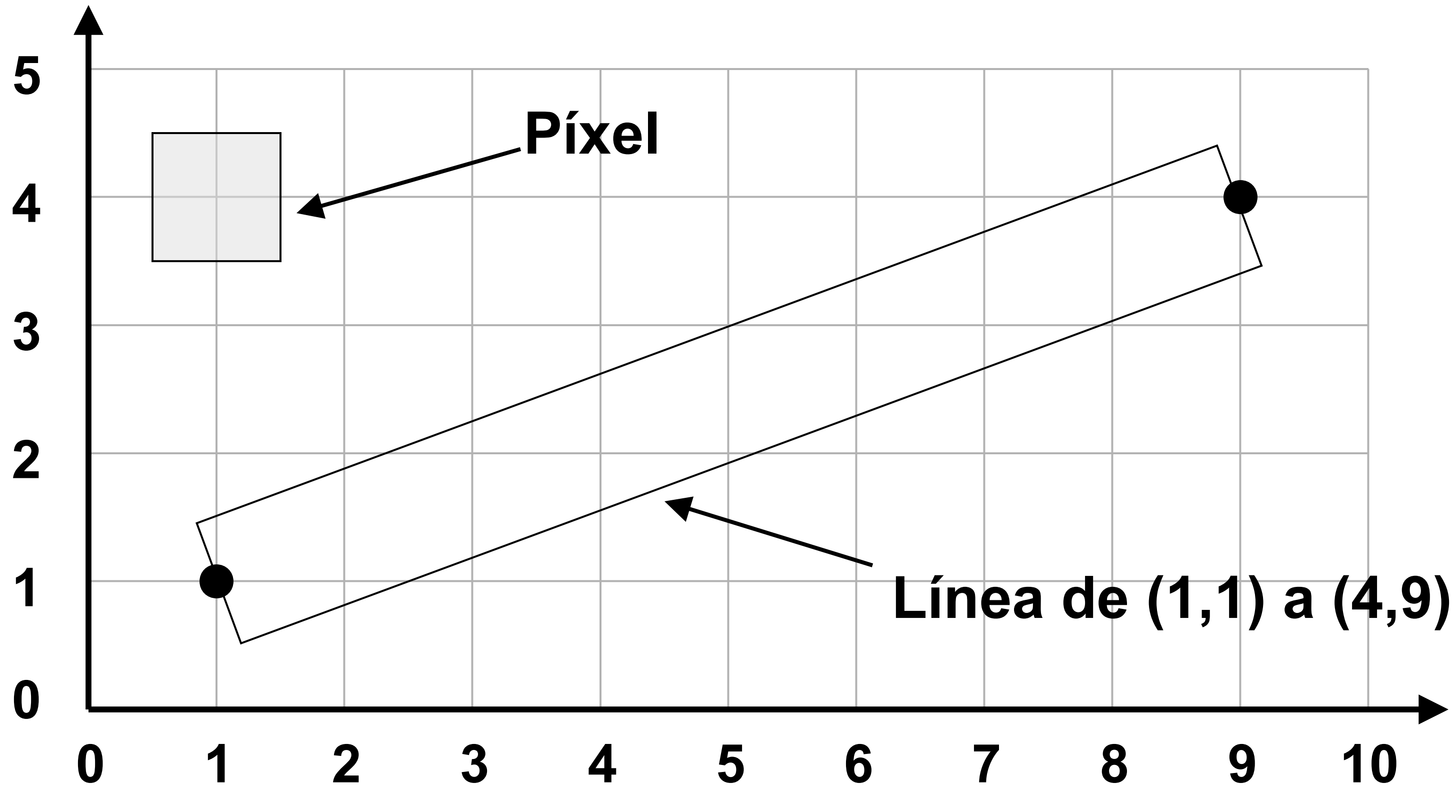
- Las aristas a veces sufren el efecto de "**serramiento**" o "**escalonamiento**". Esto es un efecto del fenómeno de *aliasing.*



Solución 1: Aumento de la resolución



Solución 2: Muestreo de área no ponderada

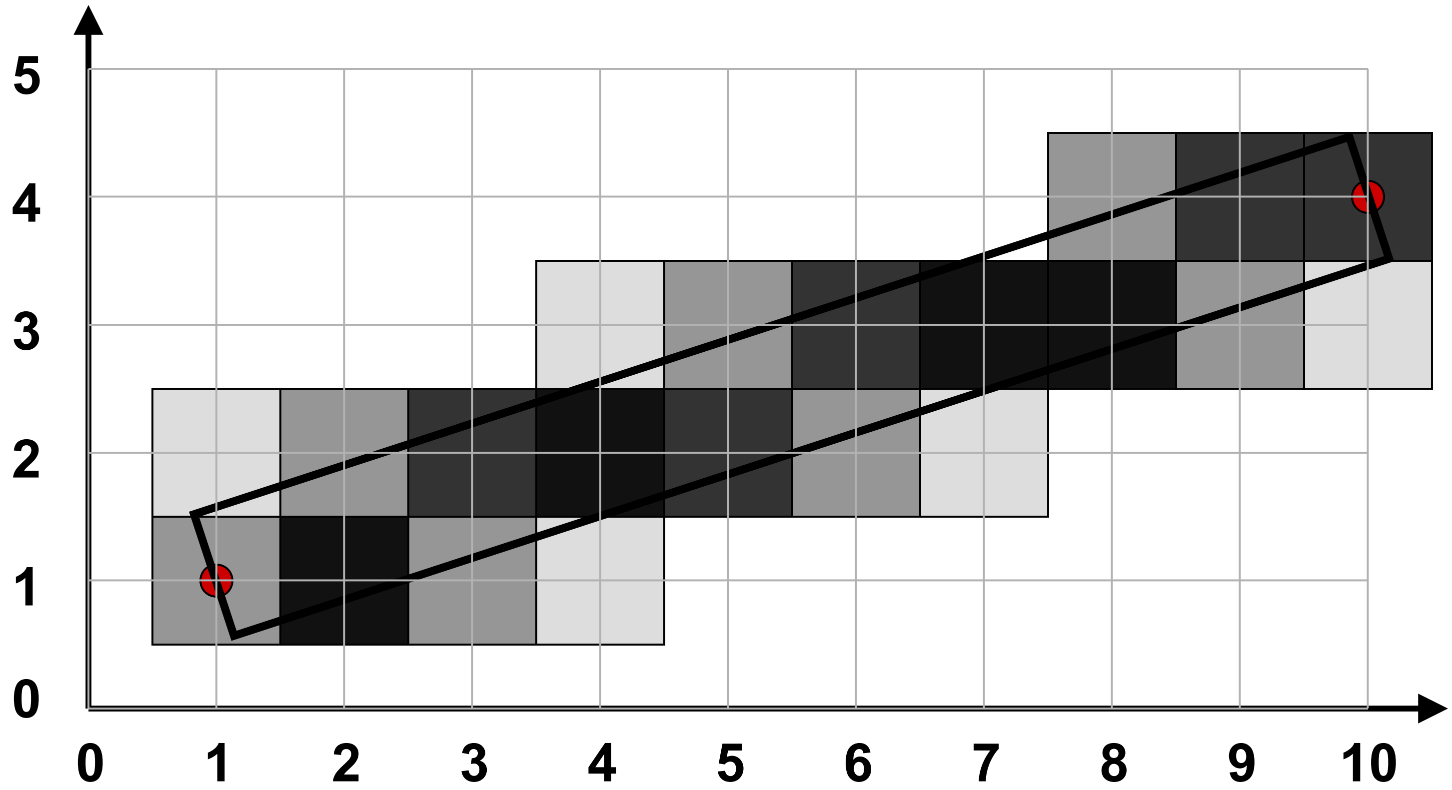


La línea se considera como un rectángulo de color que cubre una porción de la malla.

Los píxeles se los consideran como un embaldosado de azulejos.

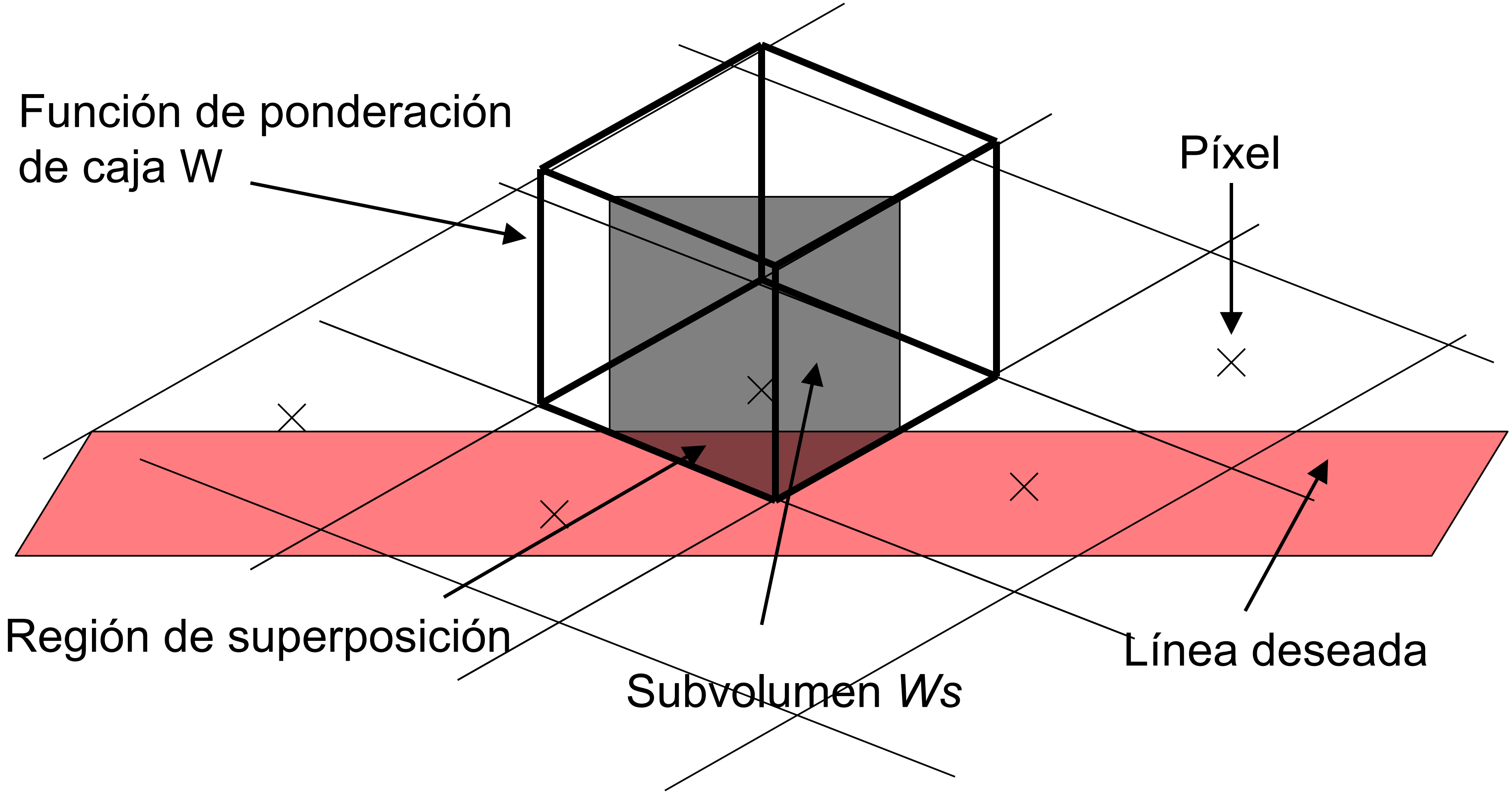
Por columna no hay en general un solo píxel pintado.

Solución 2: Muestreo de área no ponderada



Según el porcentaje del píxel cubierto por la línea, es el porcentaje del color negro.

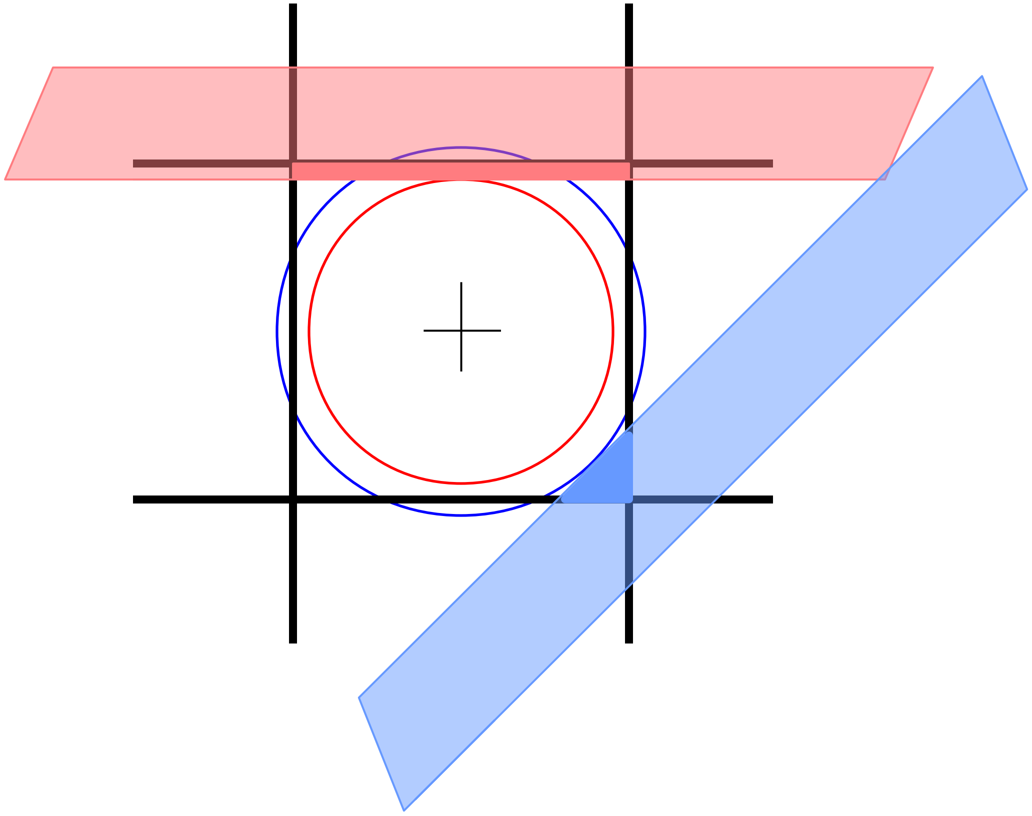
Solución 2: Muestreo de área no ponderada



El cubo tiene volumen = 1

Según el volumen Ws, es la tonalidad de gris del píxel.

Solución 2: Muestreo de área no ponderada



Áreas iguales, a distinta distancia del centro del píxel, se consideran de igual forma en área no ponderada, y de distinta forma en área ponderada

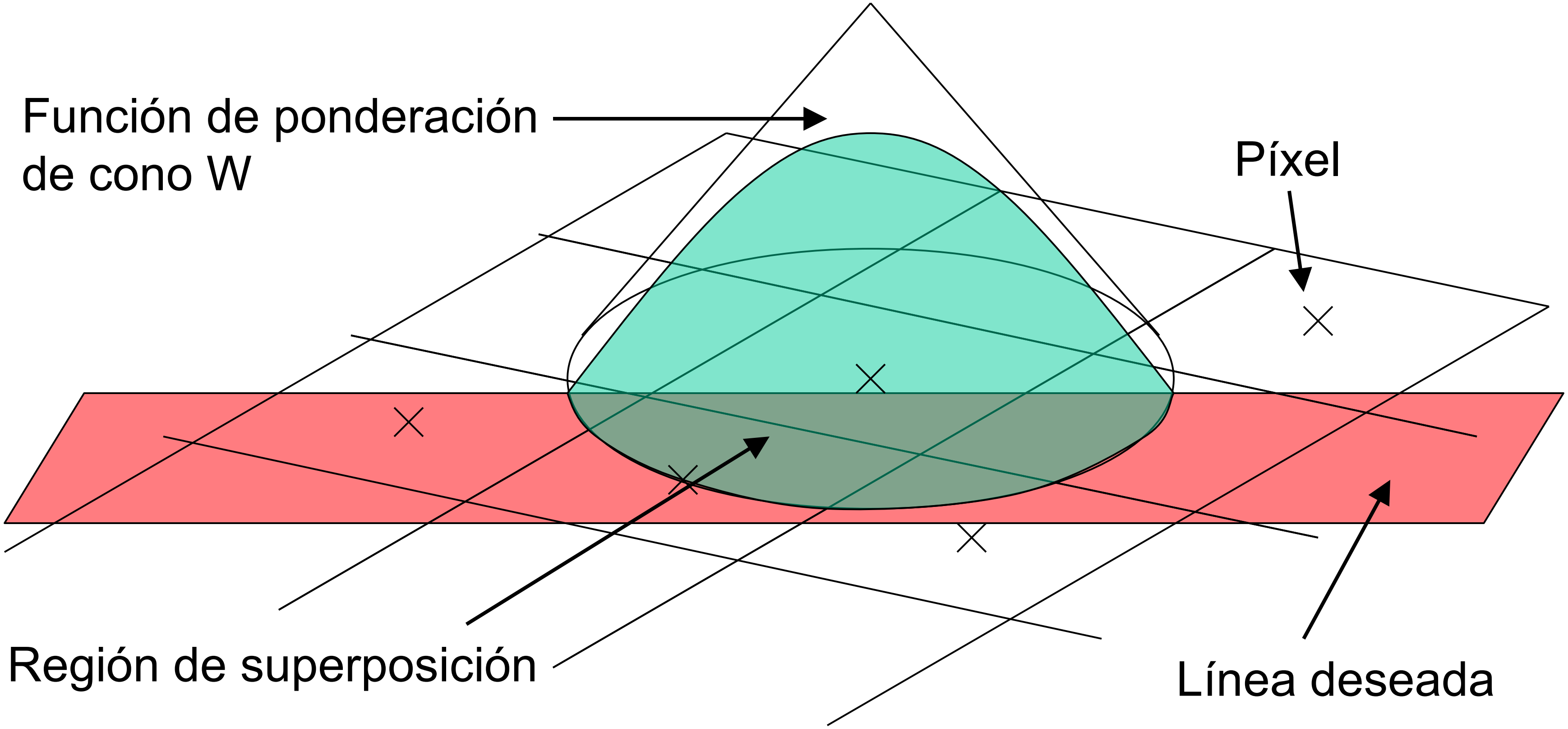
Solución 3: Muestreo de área ponderada

No ponderada versus ponderada

Área no ponderada: La intensidad del píxel se define exclusivamente por el área muestreada.

Área ponderada: La intensidad del píxel está en función del área y de la distancia de la misma al centro del píxel.

Solución 3: Muestreo de área ponderada



Según el porcentaje del píxel cubierto por la línea, es el porcentaje del color negro.