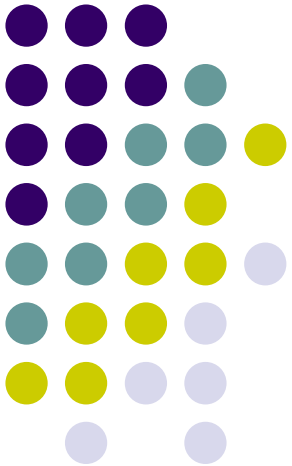
Computación Gráfica y Visualización
Tema 2: Dibujo de Primitivas 2D y Relleno

Julián Dorado

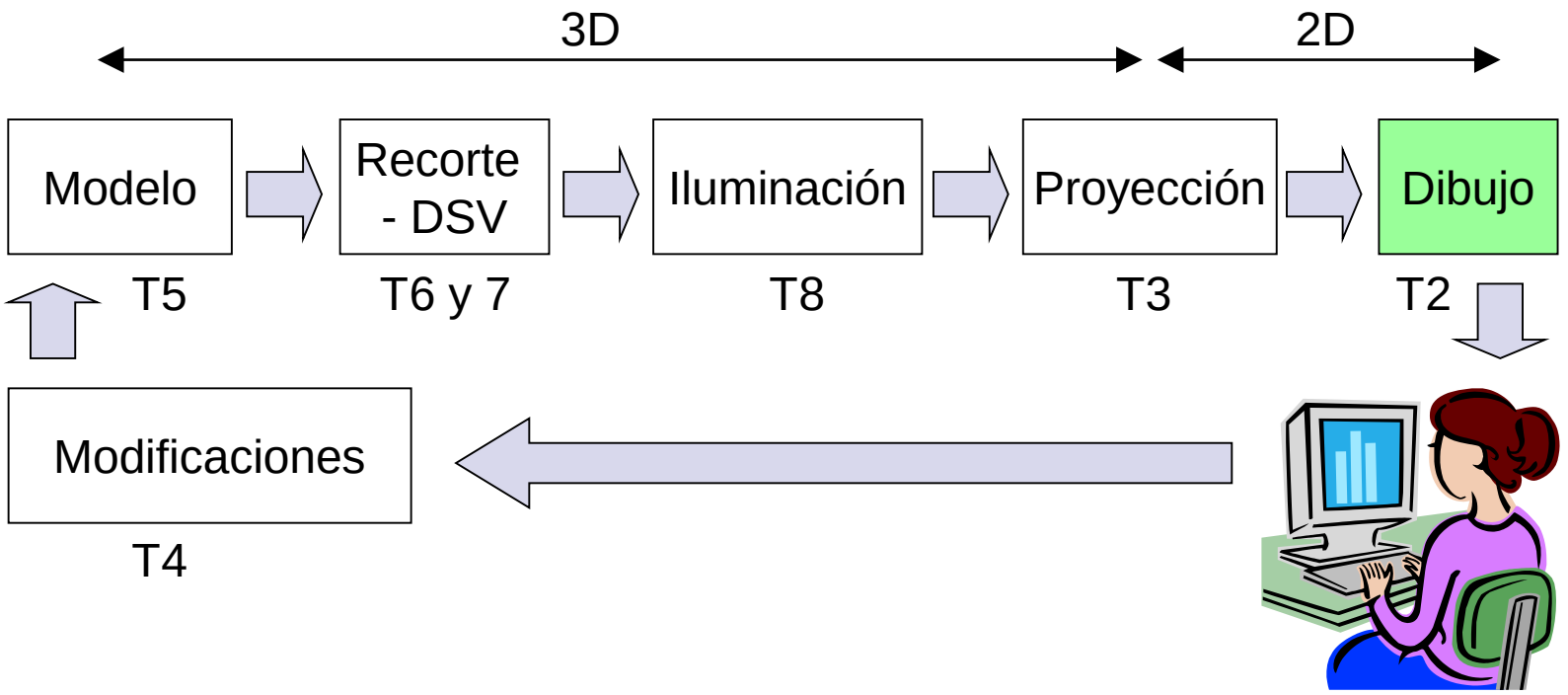
Departamento de Tecnologías de la Información y las Comunicaciones

Universidade da Coruña



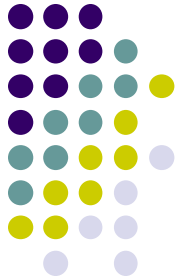
1.5 Marco conceptual para GC

- Proceso de visualización
  - Aplicación GC



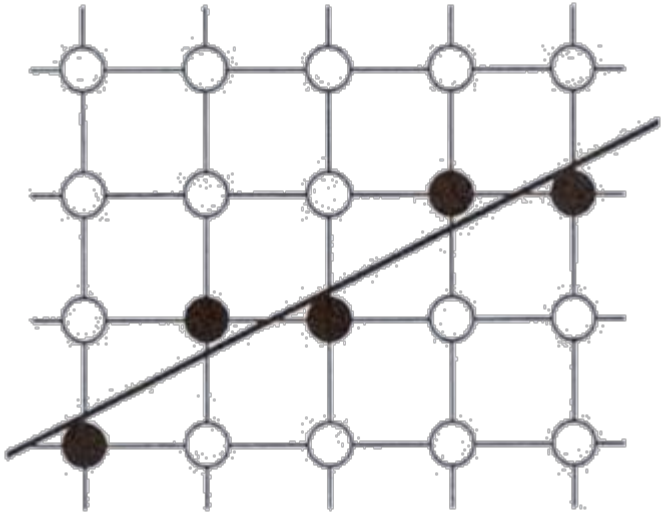
Índice

- Dibujo de líneas
  - Algoritmos
  - Algoritmo del punto medio
- Aliasing y Anti-aliasing
- Relleno de polígonos
  - Barrido
  - Inundación



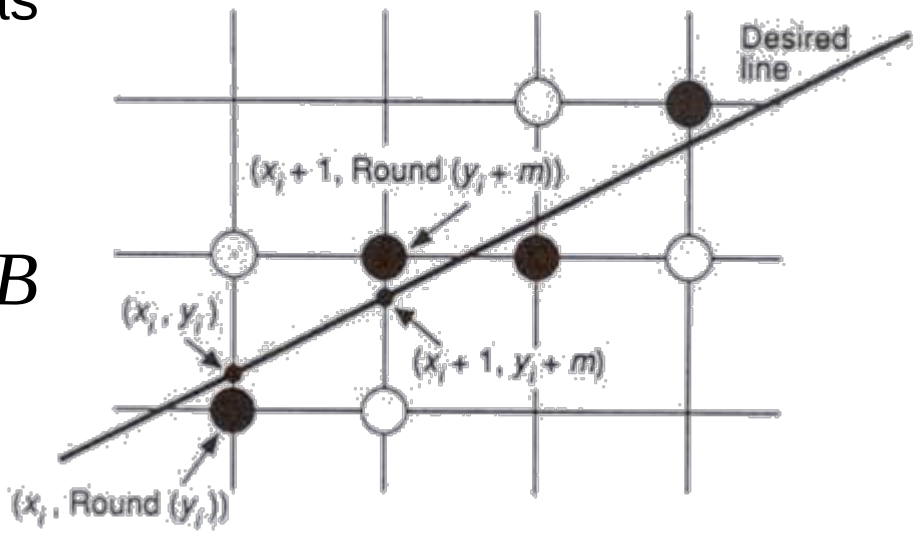
2.1 Dibujo de Líneas

- Dibujo vectorial vs bitmap
  - Calcular posiciones más próximas a línea ideal
- Características
  - Brillo constante
  - Rapidez
    - Cálculo enteros y sumas
  - Centrado en línea
  - Minimizar efecto escalera



2.1 Dibujo de Líneas

- Algoritmos de dibujado de líneas
  - Cálculo directo
  - Incremental
  - Punto medio
- Cálculo directo $y_i = mx_i + B$
  - Ecuación implícita
  - Línea (P_0,P_1)
  - Proceso
    - $x_i=x_0$
    - Repetir
      - $x_i=x_i+1$ –> calcular y_i –> Round(y_i)
    - Hasta $x_i=x_1$
  - Problemática
    - Operaciones en punto flotante (suma y producto) y redondeo



$$m = \frac{dy}{dx} = \frac{y_1 - y_0}{x_1 - x_0}$$

2.1 Dibujo de Líneas

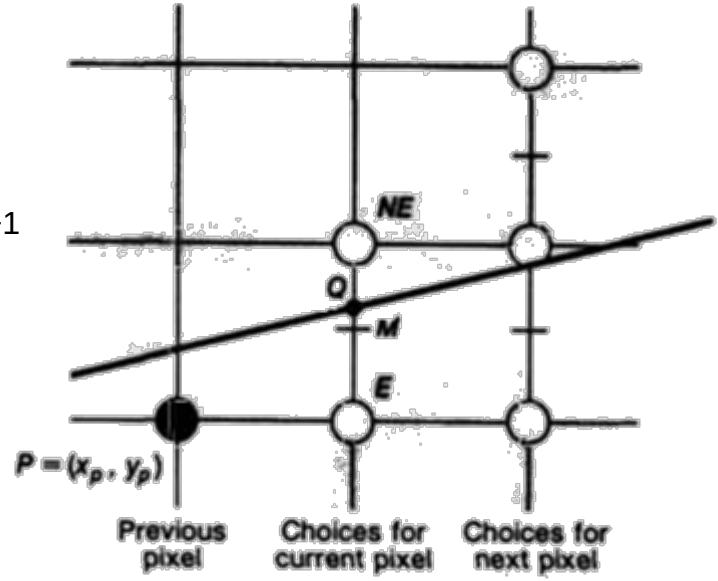
- Algoritmos de dibujado de líneas
  - Incremental

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

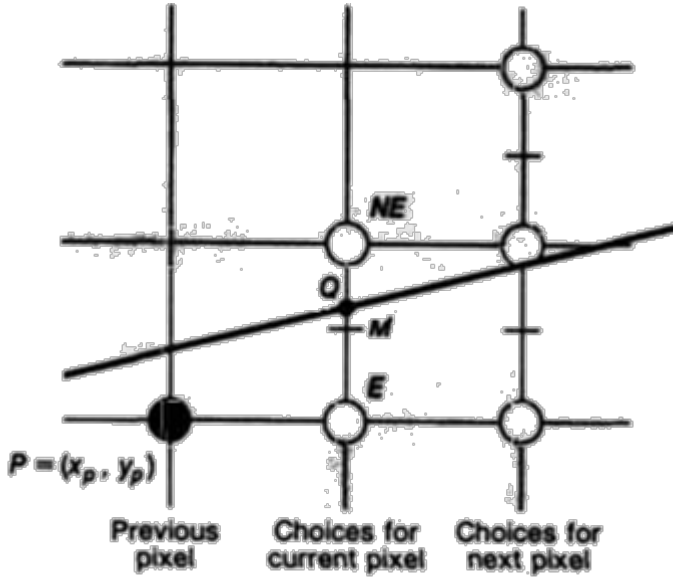
    - Si $\Delta x=1$
    - Entonces
      - $X_{i+1}=x_i+1$ –> calcular y_i $\quad y_{i+1} = y_i + m$
    - Condiciones
      - Válido para $|m|<1$
      - Para $m>1$ usar $y_{i+1}=y_i+1$ y calcular x_i y x_{i+1}
      - Usar condiciones especiales para líneas horizontales y verticales
    - Problemática
      - Precisión en el cálculo de m (punto flotante)
      - Se produce un error acumulativo en líneas de gran longitud

2.1 Dibujo de Líneas

- Algoritmos de dibujado de líneas
  - Algoritmo del Punto Medio
    - Bresenham, años 60
    - Objetivos
      - Evitar redondeos y valores en punto flotante
      - Trabajar únicamente con enteros y de forma incremental
    - Funcionamiento
      - Desde un pixel P ya seleccionado
      - Escoger entre dos siguientes
        - Pendiente [0-1] pixel E o NE
      - Q es la intersección de la línea con x_{p+1}
      - M es el punto medio entre E y NE
      - Estudiar la distancia de Q con E y NE
        - Escoger la distancia menor

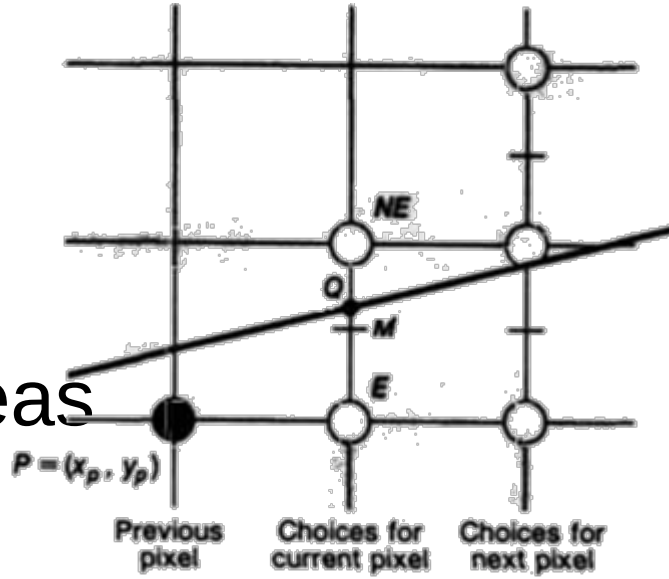


2.1 Dibujo de Líneas



- Algoritmos de dibujado de líneas
  - Algoritmo del Punto Medio
    - Funcionamiento
      - Función explícita de la línea F(x,y) = ax+by+c = 0
        - Ejemplo y=x -> F(x,y) = x-y = 0
      - Para un punto en (x,y) si F(x,y)
        - =0 el punto (x,y) está en la línea
        - >0 el punto (x,y) está por debajo de la línea
        - <0 el punto (x,y) está por encima de la línea
      - Se define una variable de decisión $d=F(M)=F(x_p+1,y_p+1/2)$
        - Si d > 0 se pinta ?
        - Si d < 0 se pinta ?
      - $d_{ini} = a(x_0+1)+b(y_0+1/2)+c = (ax_0+by_0+c)+a+1/2b = a+1/2b$
        - Multiplicar por 2 los valores de d_{ini} ΔE ΔNE
        - Sólo se estudia el signo, da igual la magnitud

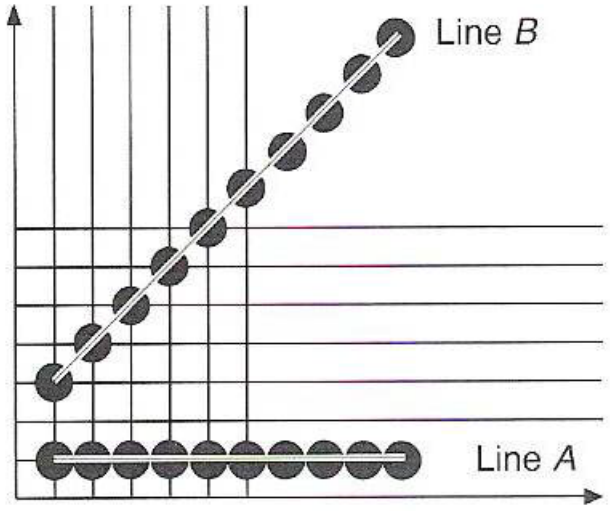
2.1 Dibujo de Líneas



- Algoritmos de dibujado de líneas
  - Algoritmo del Punto Medio
    - Funcionamiento
      - Cálculo incremental de d
        - Se escoge E -> $d_{nueva} = F(M_E) = F(x_p+2, y_p+1/2)$
        - $\Delta E = d_{nueva} - d_{vieja} = a$
        - $d = d + a$
        - Se escoge NE -> $d = F(M_{NE}) = F(x_p+2, y_p+3/2)$
        - $\Delta NE = d_{nueva} - d_{vieja} = a + b$
        - $d = d + a + b$
    - Calcular para el resto de pendientes
      - ¿Qué direcciones se escogen? ¿cuáles son los Δ?

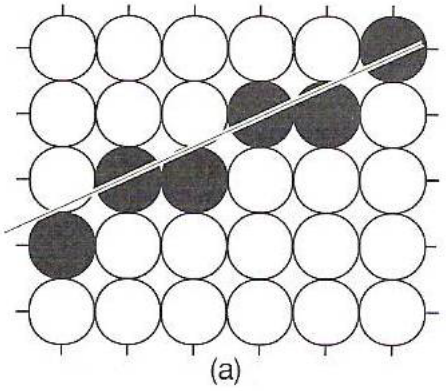
2.1 Dibujo de Líneas

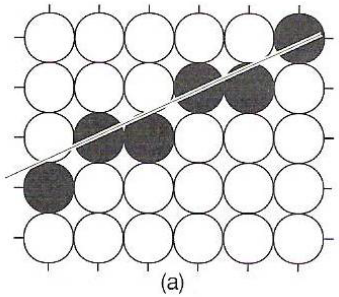
- Algoritmos de dibujado de líneas
  - Algoritmo del Punto Medio
    - Ventajas
      - Cálculo incremental, de enteros y sólo con sumas
    - Consideraciones
      - El punto de inicio P_0 está a la izquierda de P_1
      - La intensidad de la línea varía en función de la pendiente
        - Línea A: 10 pixels y longitud 1
        - Línea B: 10 pixels y longitud $\sqrt{2}$

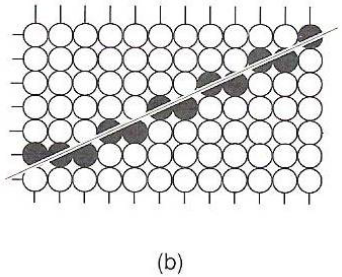


2.2 Aliasing y Anti-aliasing

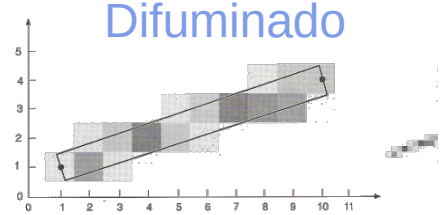
- Primitivas dibujadas en un dispositivo con posiciones discretas (píxels) produce efecto escalera o aliasing
  - Cambios de línea (a)
  - Técnicas de anti-aliasing
    - Multiplicar la resolución (b)
      - Píxels x4 solución cara
      - Doble número de saltos de la mitad de altura
      - Retrasa el problema
    - Supersampling
    - Filtering
    - Muestreo de área
      - Algoritmo de Gupta-Sproull





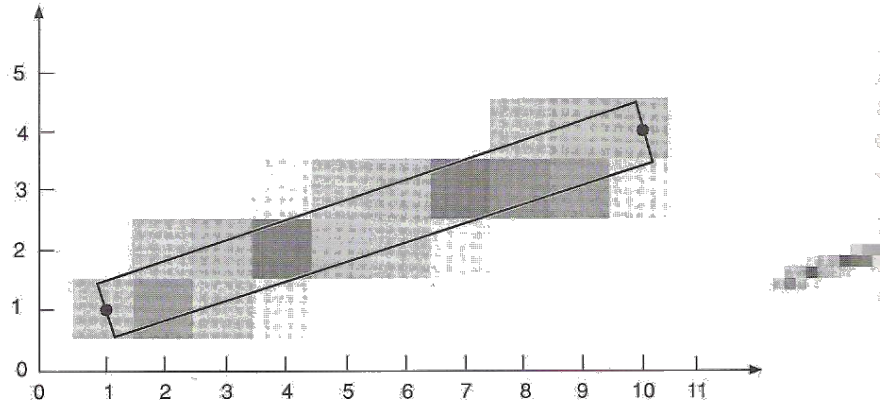


Difuminado



2.2 Aliasing y Anti-aliasing

- Difuminado
  - Las primitivas ocupan un área sobre los pixels
  - Procesar todos los pixels del área
    - Sólo líneas horizontales y verticales afectan a un solo píxel por fila o columna
  - La intensidad de cada píxel es proporcional al porcentaje de píxel cubierto por la línea
  - Esta línea con bordes difusos queda mejor en la distancia

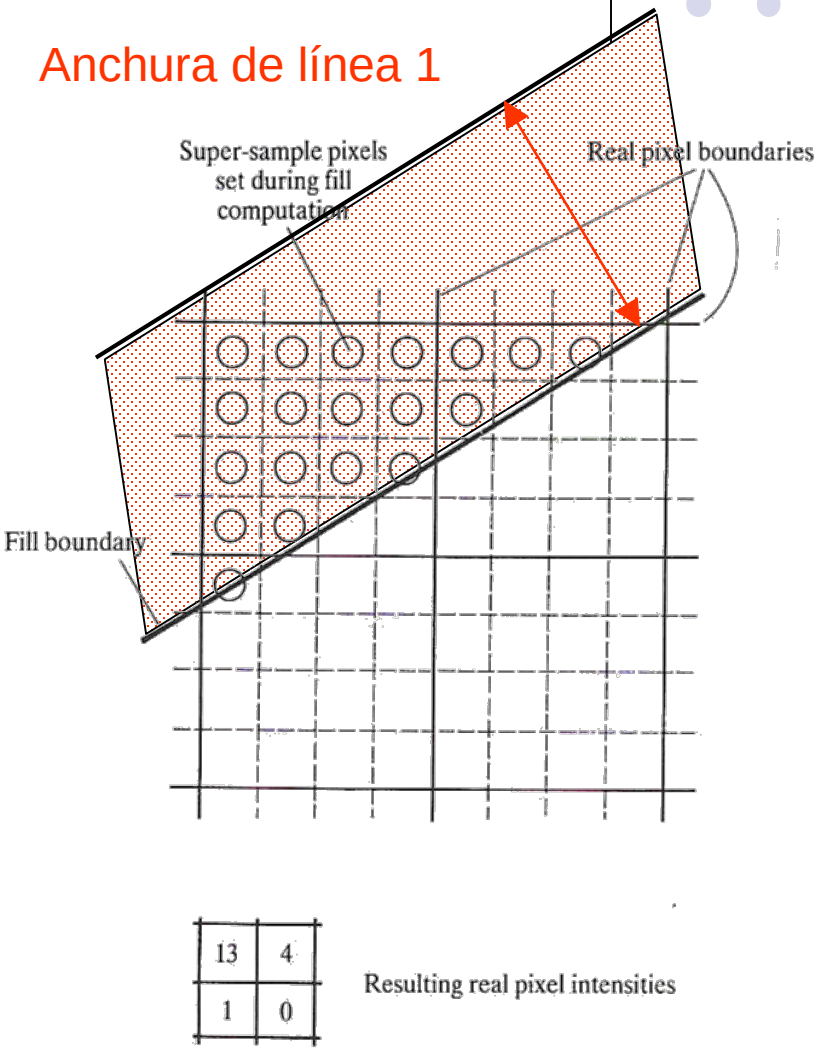


2.2 Aliasing y Anti-aliasing

- Muestreo de área
  - Anti-aliasing de fuentes en Windows

2.2 Aliasing y Anti-aliasing

- Supersampling
  - Sobremuestreo
  - Dividir los pixels
    - Resolución multiplicada x4
  - Cálculo
    - Incrementado x16
  - Intensidad del píxel final proporcional a los pixels dibujados
  - Difumina los bordes



2.2 Aliasing y Anti-aliasing

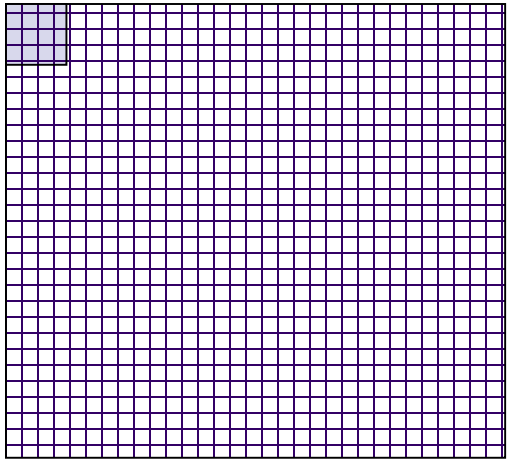
- Filtrado o Filtering
  - Utiliza un kernel de convolución
    - Filtro paso bajo

x-1,y-1	x,y-1	x+1,y-1
x-1,y	x,y	x+1,y
x-1,y+1	x,y+1	x+1,y+1

Kernel Paso bajo

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

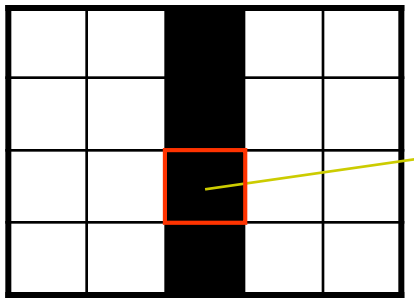
$$F(x,y) = \sum_{i,j=1}^{-1} F(x-i, y-j)K(-i,-j)$$



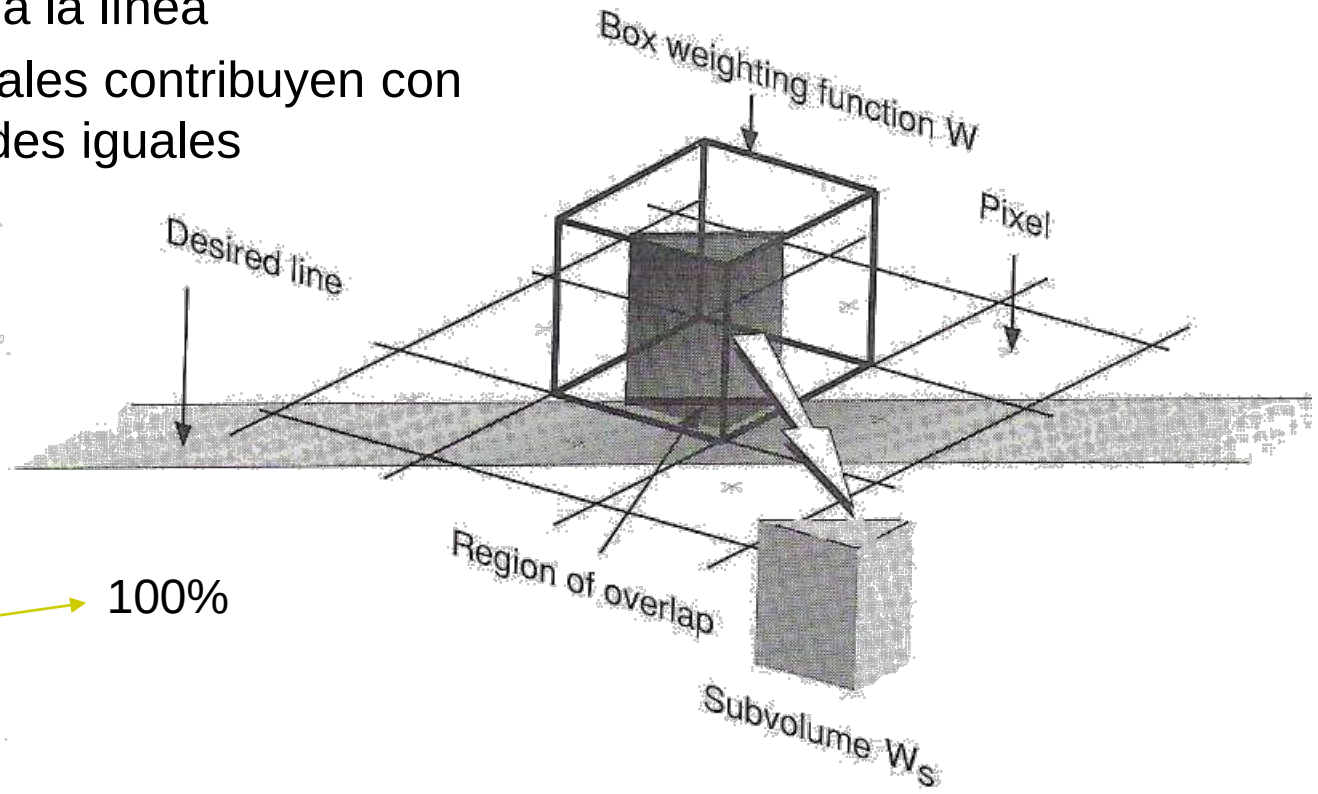
    - Difumina los bordes
    - No se tratan
      - Primera y última fila
      - Primera y última columna

2.2 Aliasing y Anti-aliasing

- Muestreo de área
  - Sin ponderación
    - Pixel se intensifica sólo si está solapado a la línea
    - Áreas iguales contribuyen con intensidades iguales

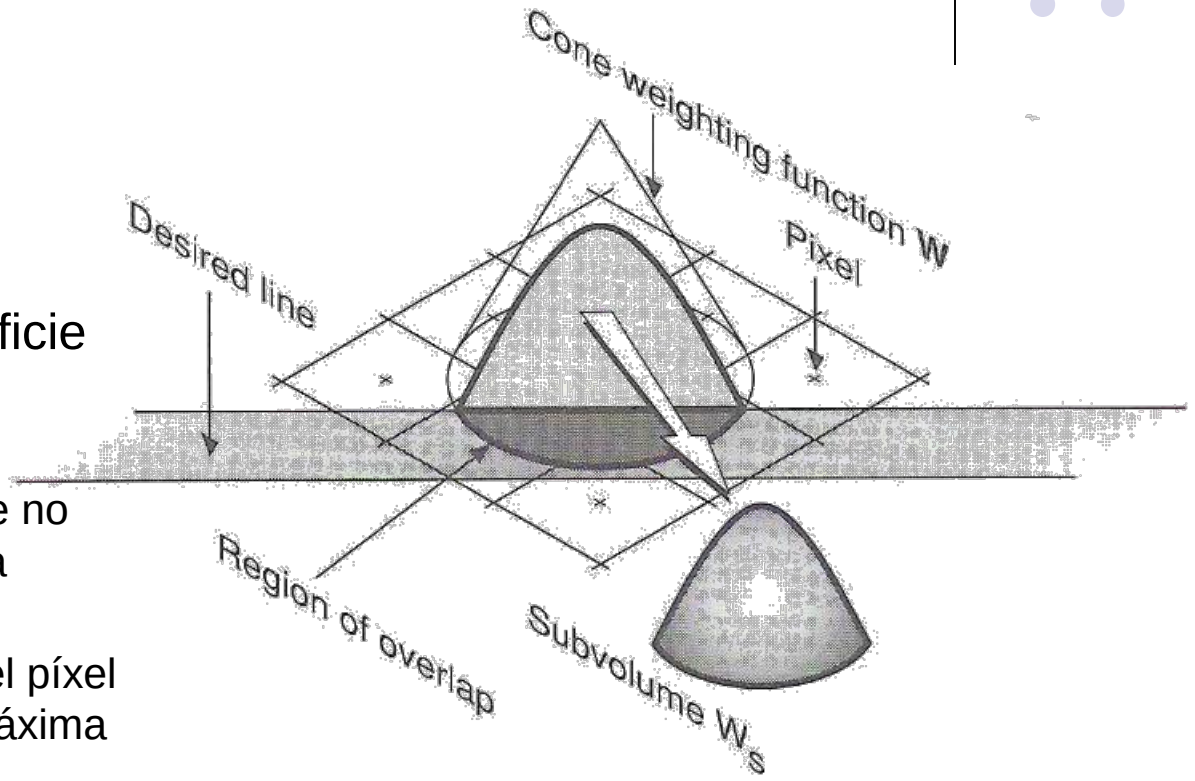


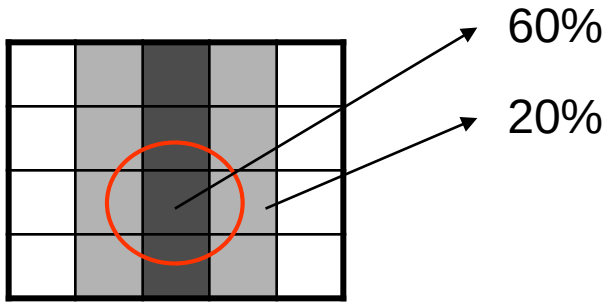
100%



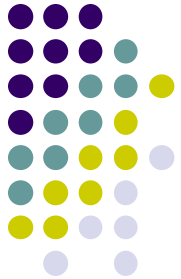
2.2 Aliasing y Anti-aliasing

- Muestreo de área
  - Con ponderación
    - Área de influencia abarca más superficie que el píxel
      - Píxel se puede intensificar aunque no solape con la línea
      - Si la línea pasa exactamente por el píxel no va a tener la máxima intensidad
        - Problema de diferencias de intensidad en líneas dependiendo de la pendiente



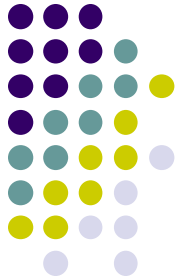


2.4 Relleno de polígonos



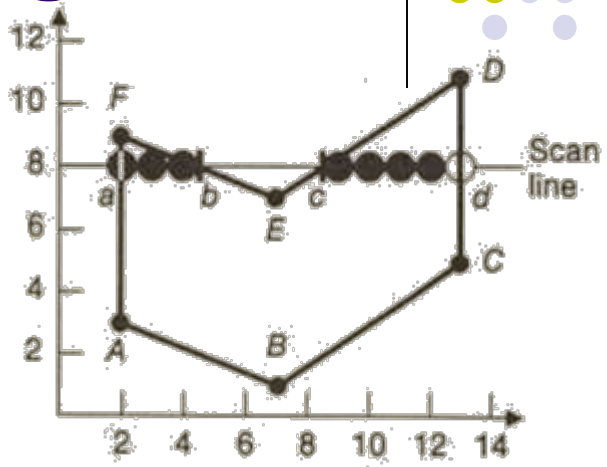
- Tareas:
  - Decidir de qué color pintar esos pixels
  - Decidir qué pixels hay que rellenar
- Métodos
  - Barrido
    - Se dispone del modelo del polígono
  - Inundación
    - No se dispone del modelo
      - Necesario un punto semilla
- Problema general
  - Aliasing con los bordes

2.4 Relleno de polígonos



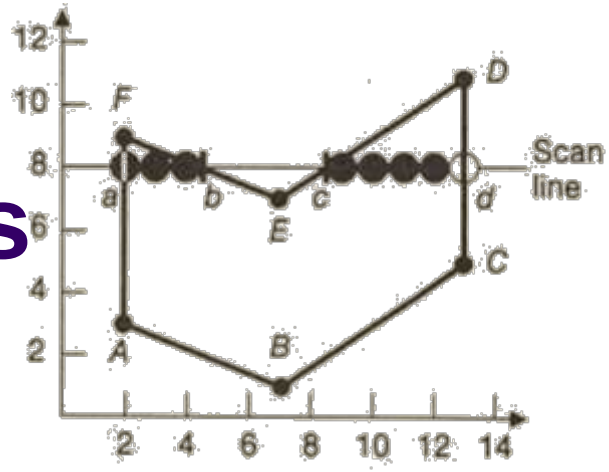
- Color de los píxels
  - Color sólido
  - Técnicas de Iluminación
    - Calcular a partir de
      - la normal del polígono
      - la posición de la fuente de iluminación
    - Distintos algoritmos (tema de Iluminación)
  - Relleno mediante patrones
    - Mapeado de texturas (tema de Iluminación)
      - Coordenadas u,v sobre coordenadas x,y,z

2.4 Relleno de polígonos

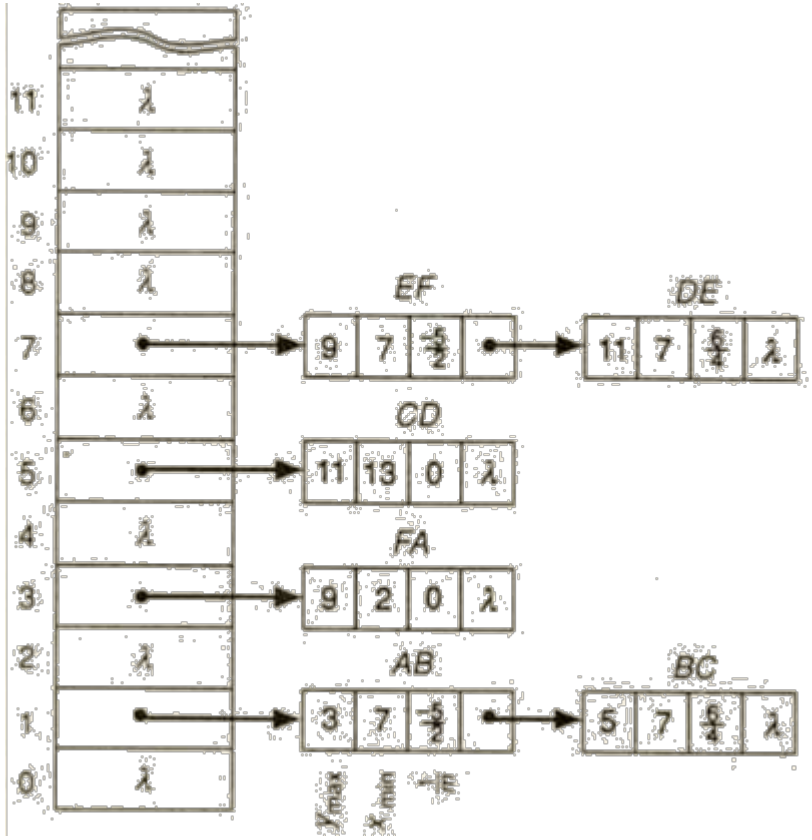


- Métodos de barrido
  - Se definen líneas de barrido
  - Recorren el polígono
    - Pasando por las aristas
  - Determinar en cada momento si la línea de barrido está dentro o fuera del polígono
    - Encontrar las intersecciones de la línea de barrido
    - Ordenar las intersecciones en eje x
    - Definir un bit de paridad que cambie al cruzar una arista
      - Pintar si es 1 y no pintar si es 0
        - Inicializar a 0
      - En vértices, no considerar los vértices y_{max}

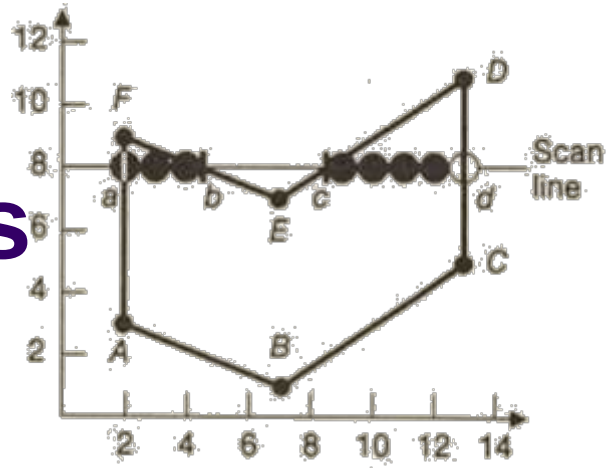
2.4 Relleno de polígonos



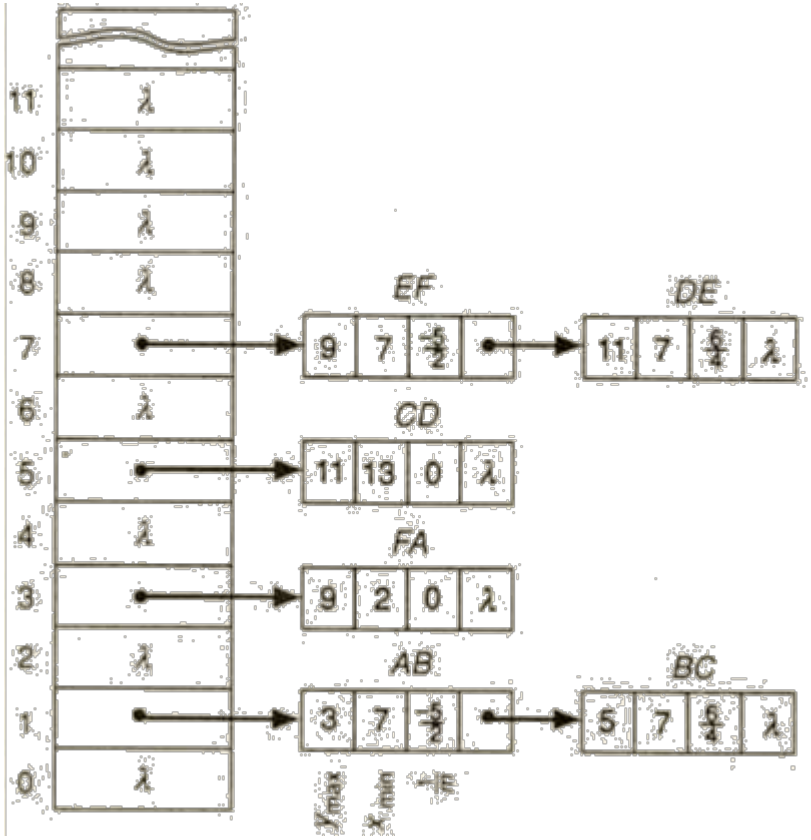
- Métodos de barrido
  - Encontrar las intersecciones de la línea de barrido
    - Lista de bordes
      - Lista por valores en y
      - Cada borde
        - Valor máximo en y
        - Valor mínimo en x
        - Pendiente



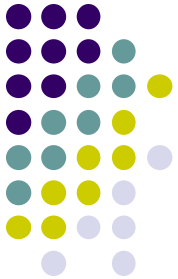
2.4 Relleno de polígonos



- Métodos de barrido
  - Encontrar las intersecciones de la línea de barrido
    - Pasos
      - Comenzar en $y_{mín}$
      - Lista de bordes inicial
      - Aumentar y
        - Insertar nuevos bordes
        - Calcular x para cada borde
        - Ordenarlos en x
        - Aplicar bit de paridad
        - Rellenar
        - Borrar bordes en y_{max}
        - Volver

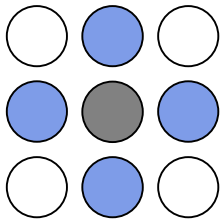


2.4 Relleno de polígonos

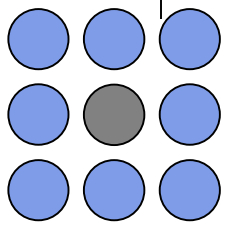


- Métodos de inundación
  - Se utilizan cuando no hay definición de los objetos
  - Procedimientos necesarios
    - Inicio
      - Identificación del punto de inicio (semilla)
        - Realizado, normalmente, por el usuario
    - Modificación
      - Decidir el color del punto inundado
        - Color de inundación
    - Propagación
      - Escoger los puntos vecinos en los que continuar el proceso
        - Vecindad
        - Por fondo o por borde
    - Final
      - Decidir que no hay más puntos a los que aplicar el proceso

2.4 Relleno de polígonos



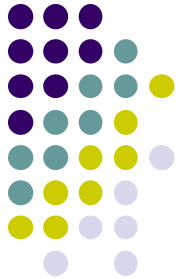
4 conectada



8 conectada

- Métodos de inundación
  - Vecindad
    - Tipos:
    - Importante para el proceso de inundación
  - Tipos de propagación
    - Por fondo
      - Propagar mientras el color de fondo sea el de la semilla
        - Se suele definir un valor de tolerancia en porcentaje
        - Textura de metal
    - Por borde
      - Propagar hasta que se llegue a un color de borde
        - Da igual el color de fondo
        - Textura de madera

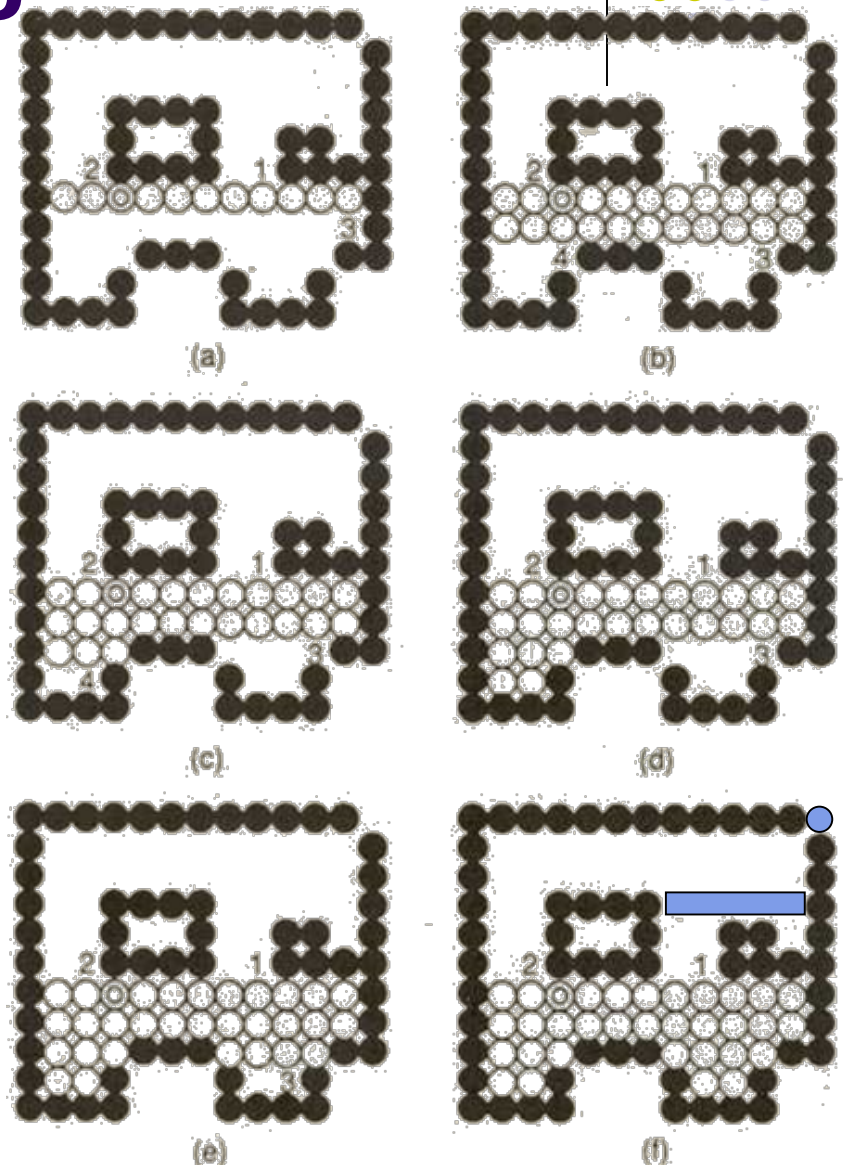
2.4 Relleno de polígonos



- Métodos de inundación
  - Algoritmo de propagación
    - Inundación Recursiva
      - Lanzar de nuevo el proceso de inundación en los pixels resultantes del proceso de propagación (4-8 vecinos)
      - Ventaja
        - Algoritmo simple
      - Desventaja
        - El número de procesos crece exponencialmente hasta que satura el sistema (pila de procesos)
    - Inundación por Barrido

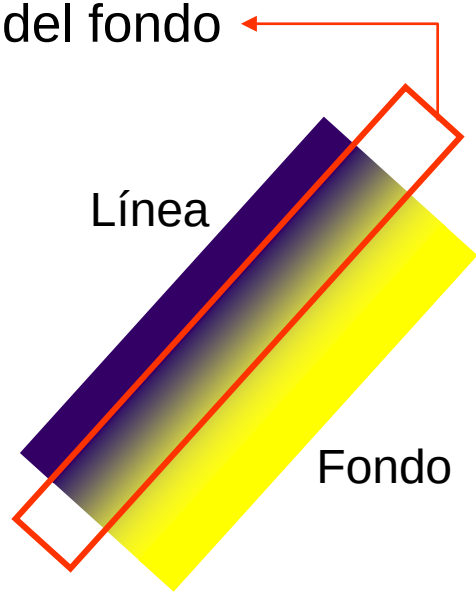
2.4 Relleno de polígonos

- Métodos de inundación
  - Algoritmo de propagación
    - Inundación Recursiva
    - **Inundación por Barrido**
      - Define líneas de barrido
      - Parte de una semilla
      - Recorre la línea de la semilla rellenando pixels que cumplen las condiciones de propagación
      - Busca vacíos (bordes derechos)
        - Arriba y abajo
        - Añade las posiciones en una pila
      - Usa los valores de la pila como semillas siguientes
      - Finaliza al vaciar la pila
      - Hay que revisar siempre arriba y abajo
      - Diferencia en vecindad 4 y 8



2.4 Relleno de polígonos

- Métodos de inundación
  - Soft-filling o relleno de bordes difusos
    - Se aplica a bordes con anti-aliasing ya aplicado
    - Zona de anti-aliasing
      - El color del borde se mezcla con el color del fondo
    - En la inundación
      - Cambiar color de fondo
      - En zona de anti-aliasing
        - Sustituir la parte de color de fondo
        - Dejar la parte de color de línea



2.4 Relleno de polígonos

- Métodos de inundación
  - Soft filling o relleno de bordes difusos
    - Color
      - F de fondo
      - L de línea
      - En la zona de anti-aliasing cada píxel
        - C = tF + (1 - t)L siendo 0 <= t <= 1
    - Calcular
      - Se conoce F y L
      - Coger un punto de color C
      - Despejar t en una componente RGB
        - Tiene que darse F_{RGB} <> L_{RGB}
          - En esa componente
        - Mejor hacerlo en varias componentes
      - Recalcular en todos los pixels
        - $C_{nuevo} = tF_{nuevo} + (1 - t)L$