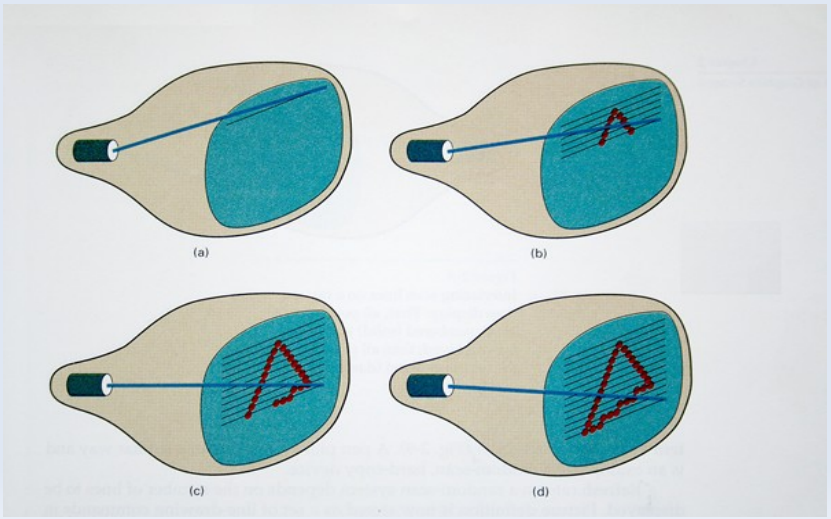
Computación gráfica: conceptos fundamentales

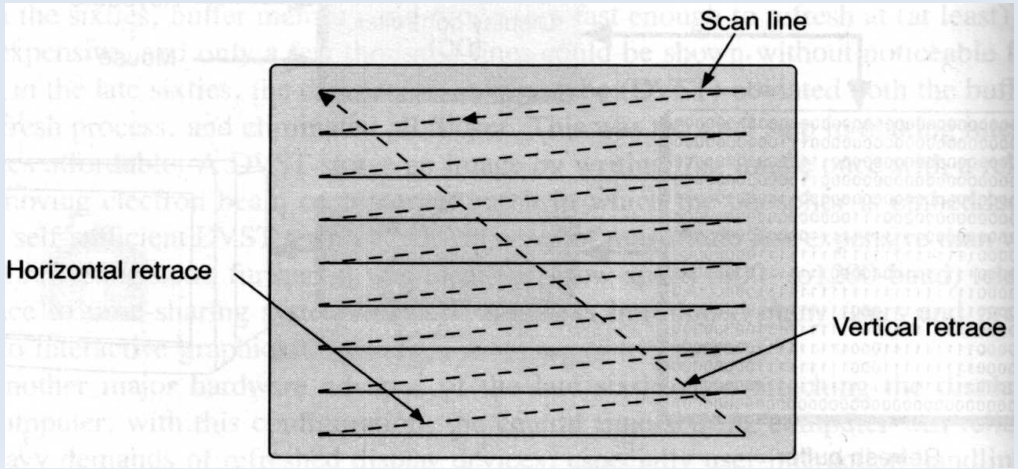
Contenido

- Display devices (Pantallas)
- Arquitectura de los sistemas raster-scan
- Cómo dibujar en el “frame buffer”?
  - Línea
  - Círculo

Display devices (pantallas)

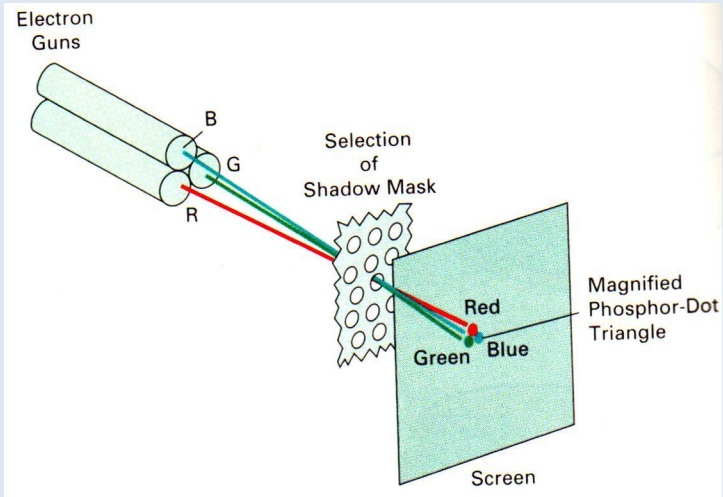
- Tubo de rayos catódicos (CRT)
  - Matriz de pixeles
  - Haz de electrones produce estimulación de fósforo (pixel)
  - Alta resolución, colores bien representados
  - Tasa de refresco: número de veces que se dibuja en la pantalla por segundo.
    - Entre 60 y 100 veces por segundo
    - Menos que 60 produce parpadeo





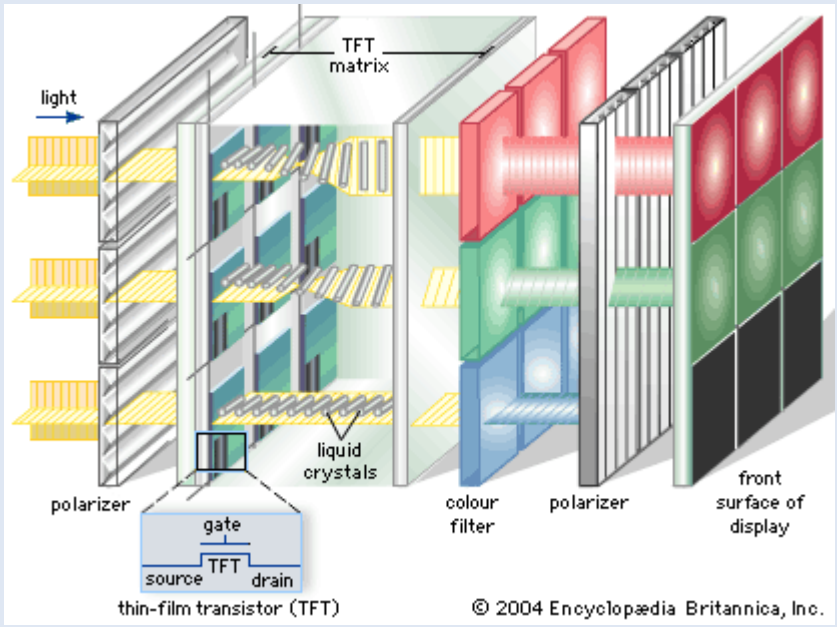
Display devices (pantallas)

- Tubo de rayos catódicos (CRT): shadow mask
  - Tres “phosphor color dots” en cada pixel: Uno emite luz roja, otro verde y el último azul
  - Tiene tres “guns” una para cada “color dot”
  - La luz emitida por los tres fósforos genera un pequeño “spot” de color en cada pixel (nuestro ojo mezcla la luz emitida)
  - Configuraciones posibles de los fósforos: triangular o en línea



Display devices (pantallas)

- Pantalla de Cristal líquido (LCD):
  - Dibujo producido pasando luz polarizada a través de un cristal líquido que puede ser alineado para ya sea dejar pasar luz o bloquear su paso.
    - Passive matrix LCD (más antiguas)
    - Active matrix LCD: se usan transistores para controlar el voltage de cada pixel

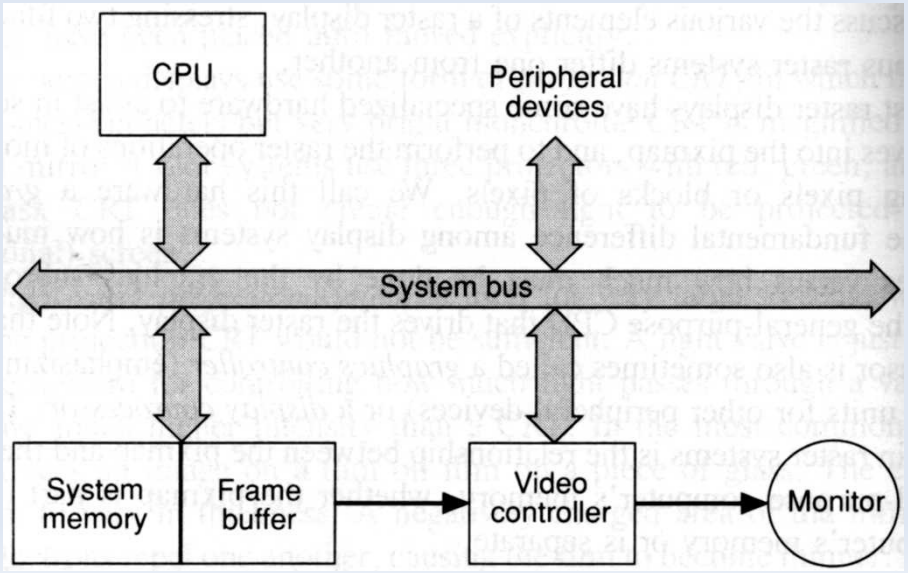


Display devices (pantallas)

- Pantalla de cristal líquido (LCD):
  - Matriz de celdas accesibles (pixeles)
  - Celdas contienen moléculas de cristal líquido que se alinean con la carga
  - No tiene parpadeo: la imagen debe ser actualizada solo si cambia (mayor persistencia).
  - Tasa de refresco: 60Hz, 120Hz o 240Hz ...
  - Más livianas que las CRT
  - Menos consumo de energía que las CRT
  - Mirar:
  - http://en.wikipedia.org/wiki/Liquid_crystal_display
  - http://www.topreviewshop.com/lcd_refresh_rates_explained_240hz_vs_120hz_vs_60hz

Arquitectura de los sistemas raster-scan

- Frame buffer: area de memoria usada para almacenar el dibujo
  - Posiciones en el frame buffer y las correspondientes posiciones de la pantalla se referencian en coordenadas cartesianas
- Video Controller
  - accesa el frame buffer para resfrescar la pantalla
  - obtiene los valores de los pixeles de áreas de memoria diferentes durante el ciclo de refresco
  - contiene la “look up table” que define la tabla de colores a usar



Arquitectura de los sistemas raster-scan

- Colores RGB (red, green, blue)
- Como especificar un color?
  - Esquema directo: Almacenar códigos de color directamente en el frame buffer
  - Ejemplo: frame buffer de 3 “bit planes” => 8 colores (el bit de la izquierda corresponde al rojo, el del medio al verde, y el bit derecho al azul)

TABLE 4-1

THE EIGHT RGB COLOR CODES FOR A THREE-BIT PER PIXEL FRAME BUFFER

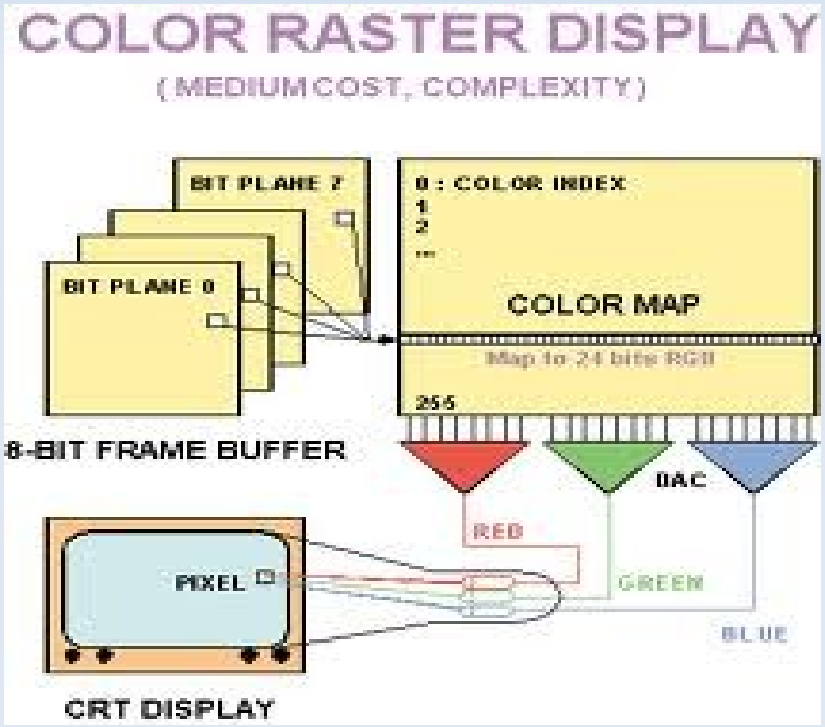
Color Code	Stored Color Values in Frame Buffer			Displayed Color
	RED	GREEN	BLUE	
0	0	0	0	Black
1	0	0	1	Blue
2	0	1	0	Green
3	0	1	1	Cyan
4	1	0	0	Red
5	1	0	1	Magenta
6	1	1	0	Yellow
7	1	1	1	White

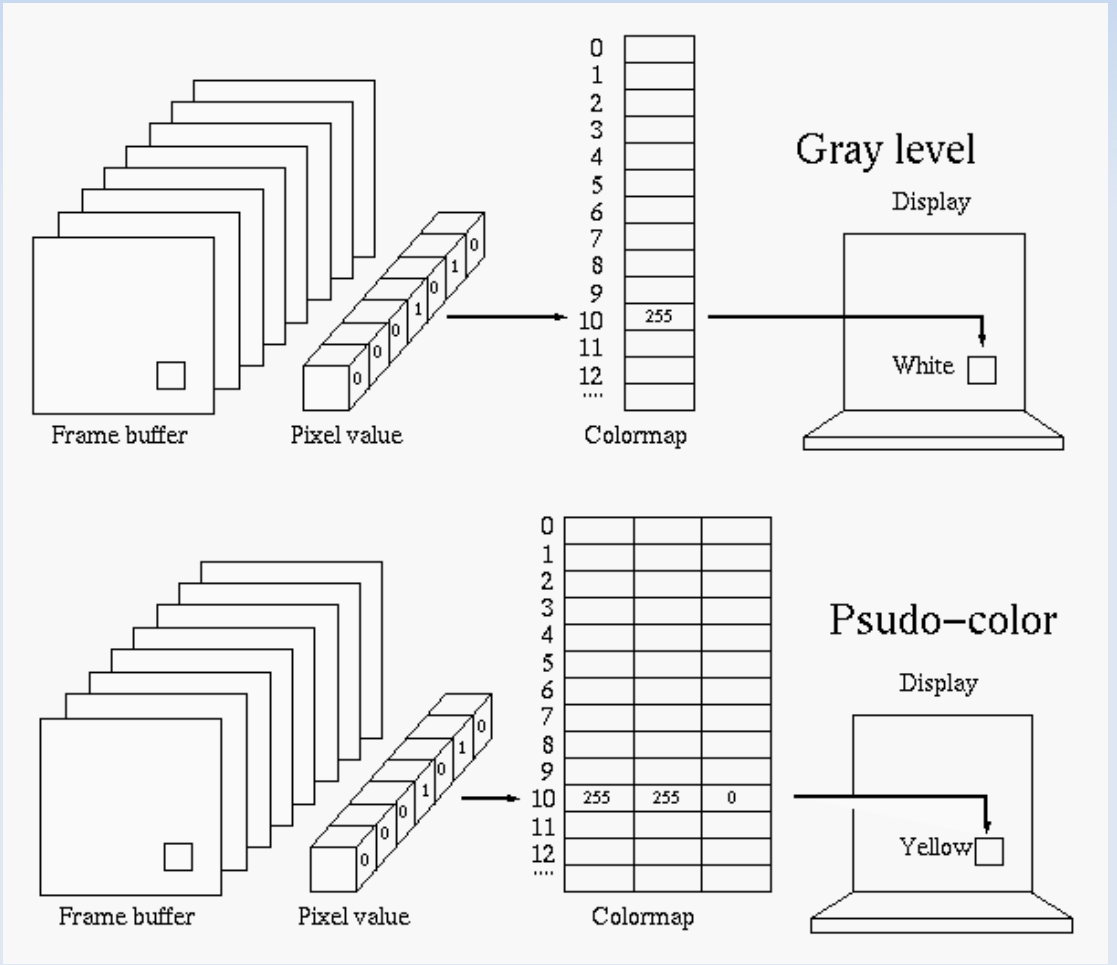
Arquitectura de los sistemas raster-scan

- Esquema directo:
  - Cómo agregar más colores: agregar más “bit planes”
  - Ventajas: simple
  - Desventajas: requerimiento de memoria alto
    - Ejemplo: 6-bit planes … cuántos colores? Cuánta memoria se necesita para el frame-buffer con resolución de 1024x1024?
    - Con 24-bit pixel y la misma resolución?

Arquitectura de los sistemas raster-scan

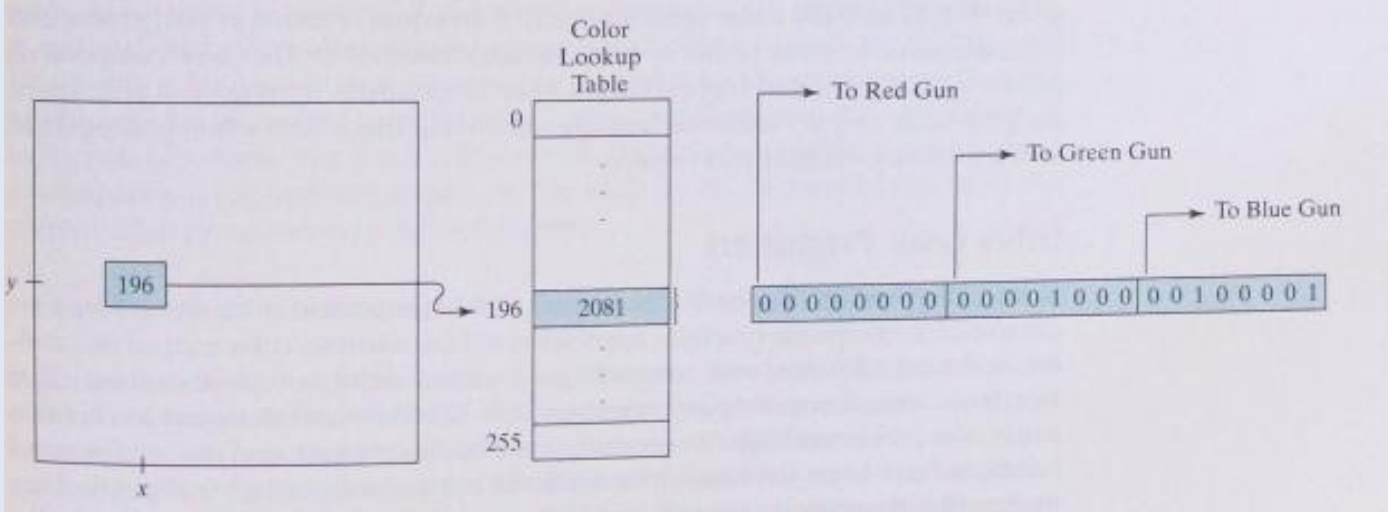
- Esquema indirecto: Almacenar el color a usar en una tabla separada (look up table) y usar el número almacenado en el frame buffer como índice a la tabla





Arquitectura de los sistemas raster-scan

- Esquema indirecto: En la figura
  - Cada pixel puede tene run valor entre 0-255
  - Cada posición de la tabla tiene 24 bits para especificar unl color: Ej el índice 196 almacena el color 0x821
  - Provee flexibilidad para cambiar el color usado para un mismo índice, o cambiar la tabla completa
  - Cuánta memoria usamos frame-buffer + table?



Arquitectura de los sistemas raster-scan

- Esquema indirecto:
  - Uso de dos tablas de colores

False color

True color

Arquitectura de los sistemas raster-scan

- Display processor (graphic processor, display co-processor)

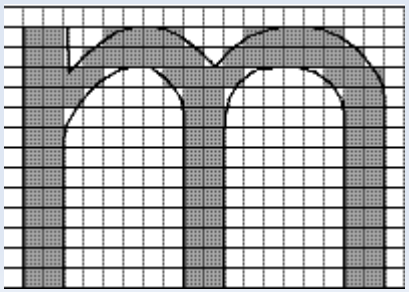
  - Liberar a la CPU de las operaciones gráficas

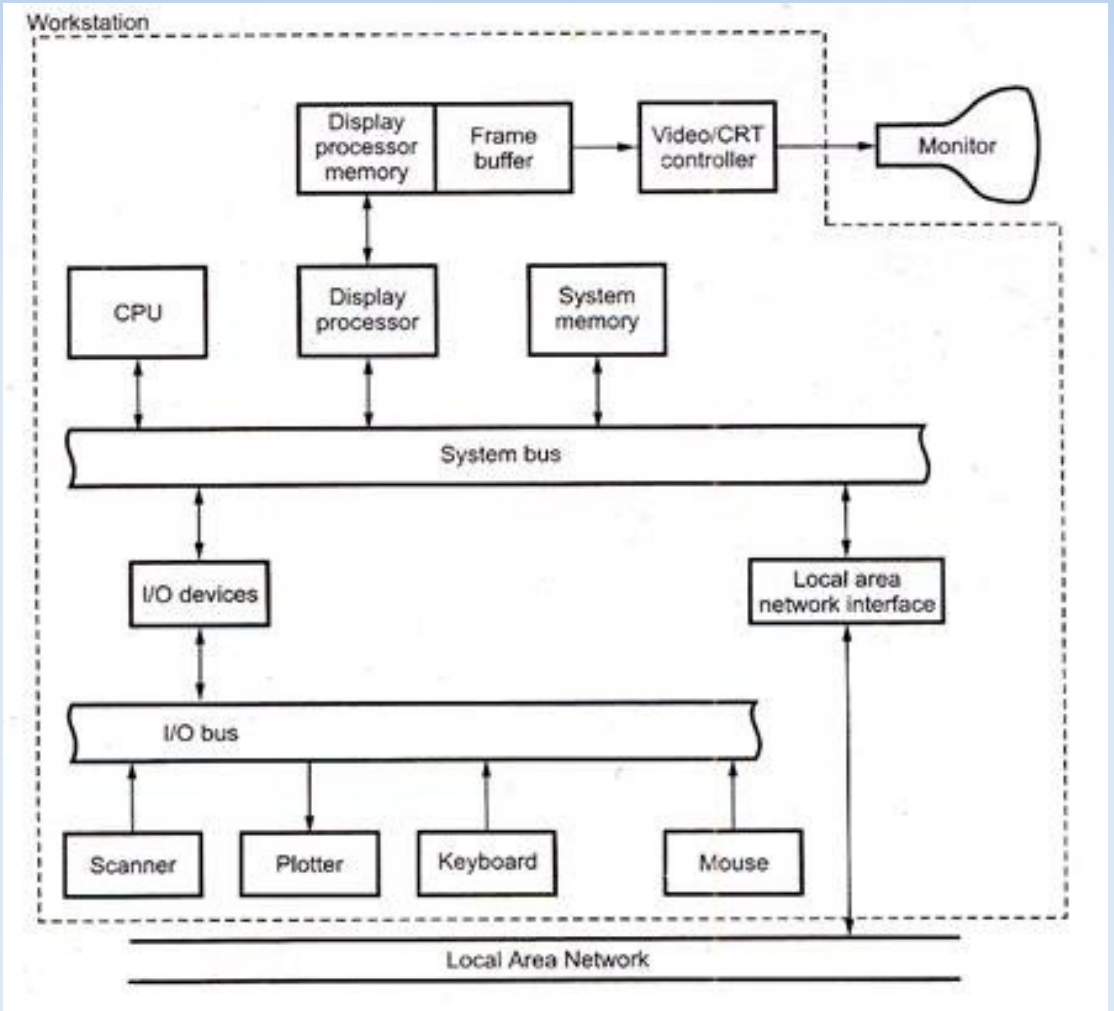
    a. trasformar el dibujo en conjunto de pixeles y almacenarlos en el frame buffer (scan-conversion, rasterización)

    + Ej: convertir una línea en en conjunto de puntos discretos

    b. generar estilos de línea, áreas rellenas, aplicar transformaciones a los objetos

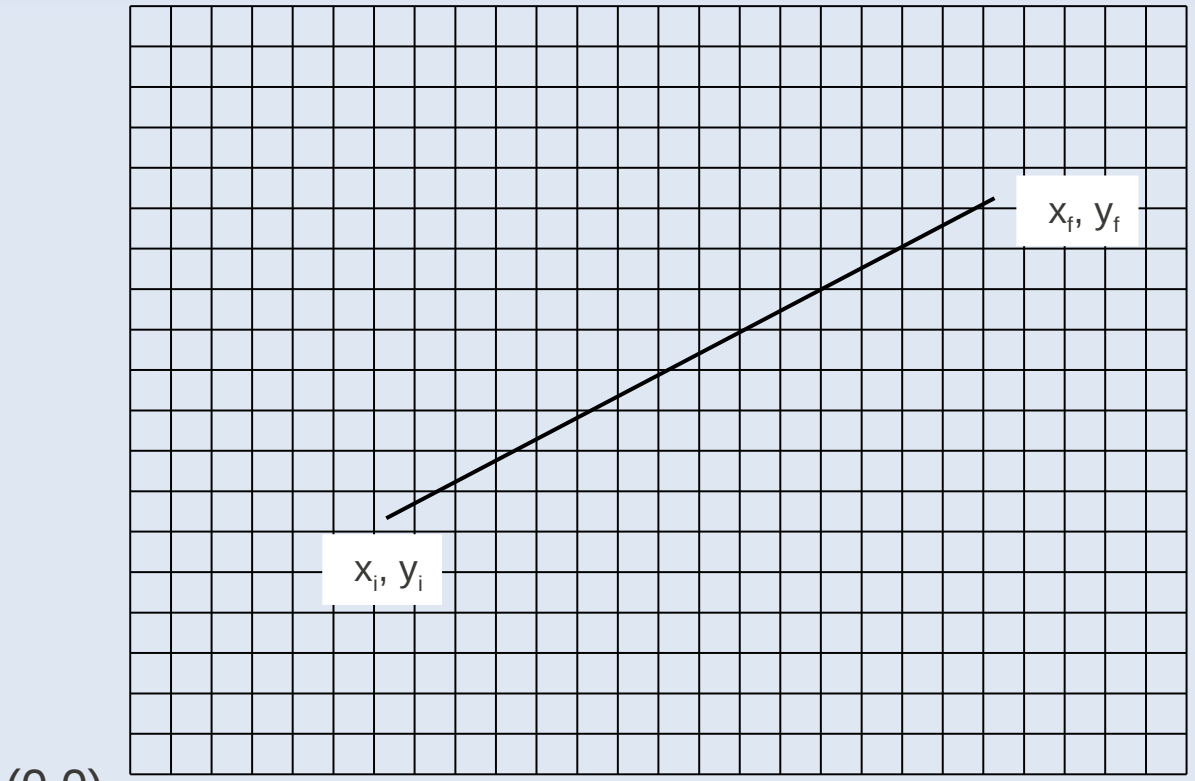
    c. obtener información de los dispositivos de entrada. Ej: mouse





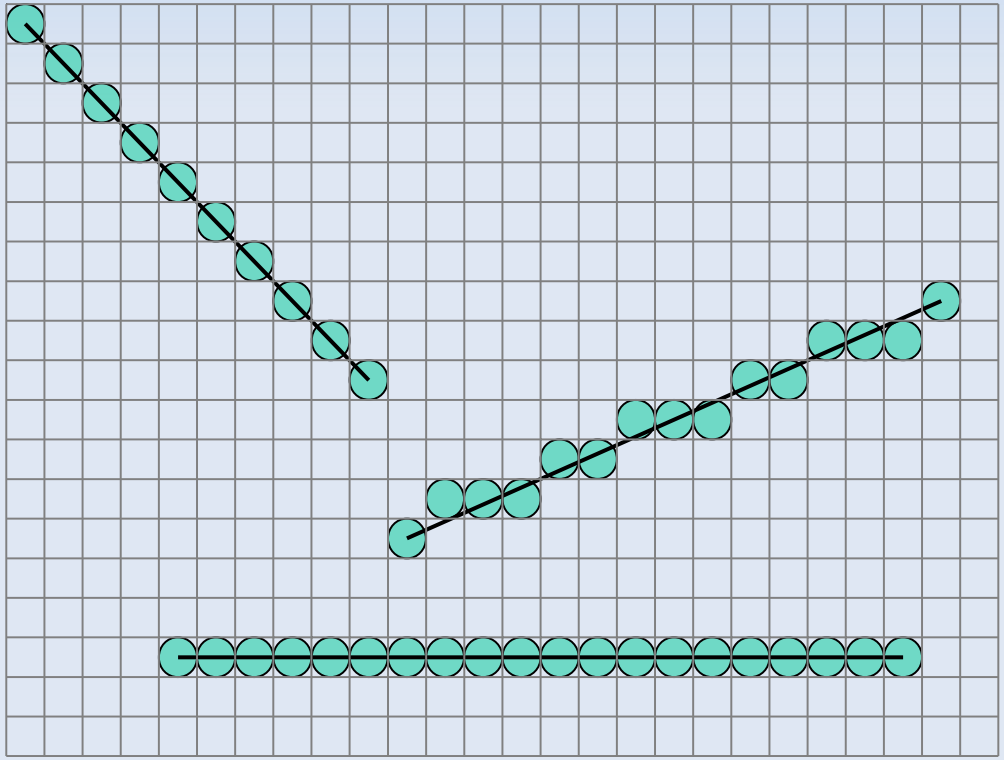
Cómo dibujar en el frame buffer? Una línea

- Arreglo 2D de pixeles
- Asignación individual de intensidad/color para cada pixel
- Pixeles centrados en coordenadas enteras
- *setPixel(x, y)*



Cómo dibujar en el frame buffer? Una línea

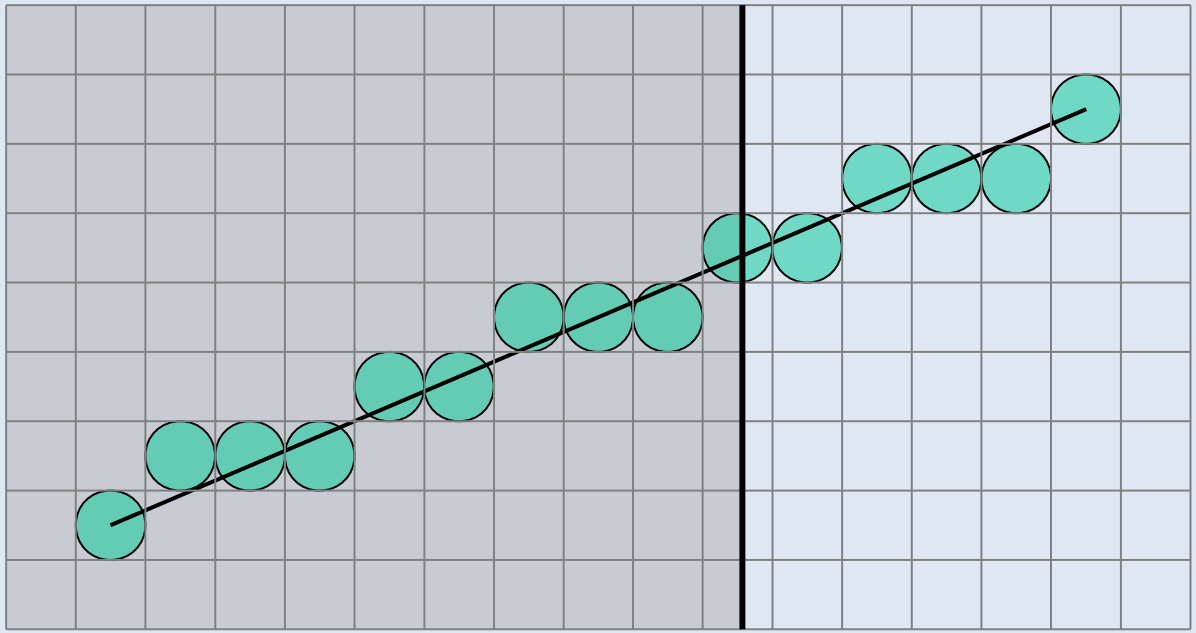
- Dados los puntos extremos de una línea (x_i, y_i) y (x_f, y_f) identificar las posiciones a dibujar
- Qué requisitos se desea cumplir?



Mirar: http://www.netgraphics.sk/

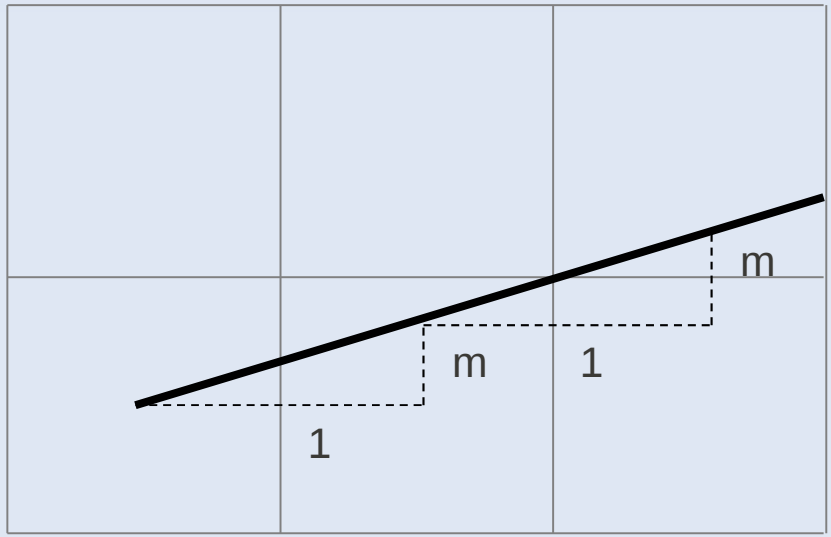
Cómo dibujar en el frame buffer? Una línea

- Algoritmo ingenuo:
  - y = f(x) = mx+b
  - para cada x en [x_i, x_f], *setPixel(x, round(y(x)))*



Cómo dibujar una línea?

- Algoritmo DDA (digital differential analizer):
  - Para cualquier intervalo en x $\delta y = m\,\delta x$ y, de manera similar $\delta x = \delta y / m$
  - Supongamos que: 0 <= m <= 1, $\delta x = 1$,
    - $y_{k+1} = y_k + m$
  - Algoritmo incremental:
    - $x = x_i$
    - $y = y_i$
    - setPixel(x, round(y))
    - While $x < x_f$
      - x = x + 1
      - y = y + m
      - setPixel(x, round(y))

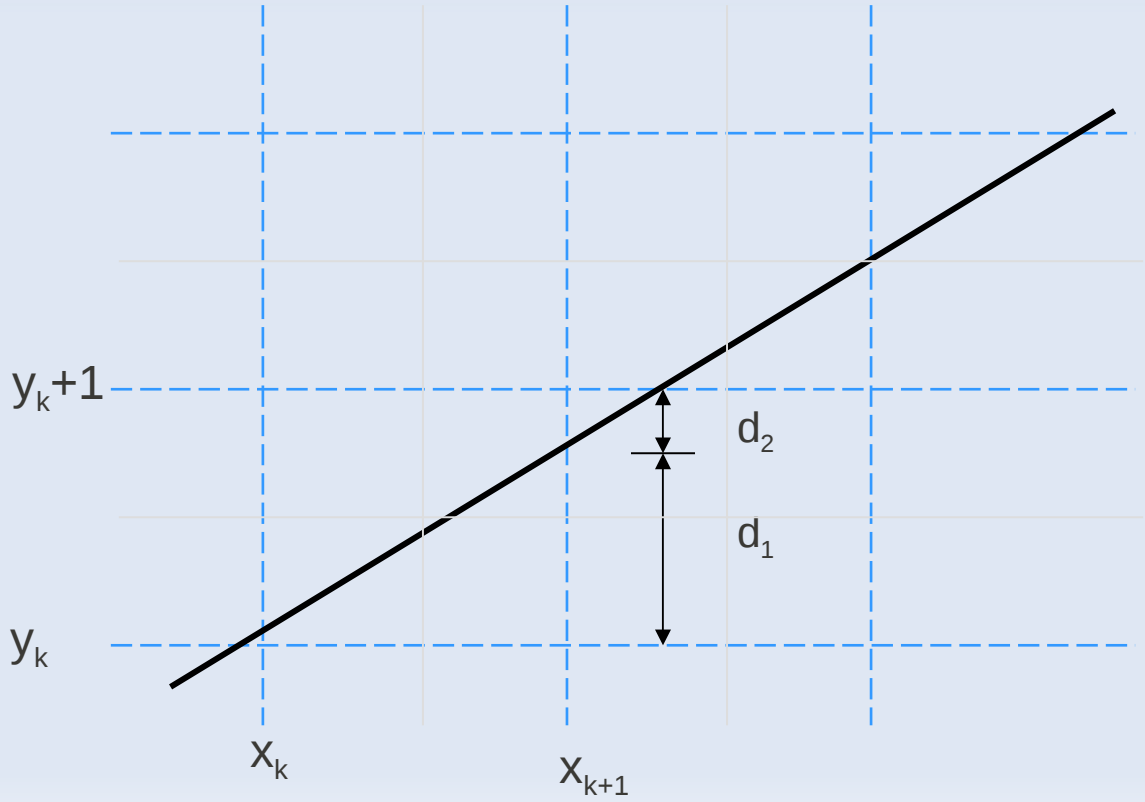


Qué problemas puede tener?

Propuesto: completarlo

Cómo dibujar una línea?

- Algoritmo de Bresenham
  - Usa solo operatoria incremental entera
  - Calcula un valor de decisión entero para elegir qué pixel usar



Cómo dibujar en el frame buffer?

- Algoritmo de Bresenham (0 < m < 1)
  - $y = m(x_k + 1) + b$
  - $d_{lower} = y - y_k = m(x_k + 1) + b - y_k$
  - $d_{upper} = (y_k + 1) - y = y_k + 1 - m(x_k + 1) - b$
  - $d_{lower} - d_{upper} = 2\mathrm{m}(x_k + 1) - 2\mathrm{y}_k + 2\mathrm{b} - 1$
  - Dado que m = δy / δx
  - $p_k = \delta x (d_{lower} - d_{upper}) = 2\,\delta y\, x_k - 2\,\delta x\, y_k + c$
  - En el paso k+1 $p_{k+1} = 2\,\delta y\, x_{k+1} - 2\,\delta x\, y_{k+1} + c$
  -
  - Restando $p_{k+1} - p_k = 2\,\delta y\,(x_{k+1} - x_k) - 2\,\delta x\,(y_{k+1} - y_k)$

Cómo dibujar una línea?

- Algoritmo de Bresenham (|m| < 1)
  - $y = m(x_k + 1) + b$
  - $d_{lower} = y - y_k = m(x_k + 1) + b - y_k$
  - $d_{upper} = (y_k + 1) - y = y_k + 1 - m(x_k + 1) - b$
  - $d_{lower} - d_{upper} = 2\mathrm{m}(x_k + 1) + b - 2\mathrm{y}_k + 2\mathrm{b} - 1$
  - Dado que m = $\delta y / \delta x = (y_f - y_i)/(x_f - x_i)$

$$p_k = \delta x (d_{lower} - d_{upper}) = 2\,\delta y\, x_k - 2\,\delta x\, y_k + c$$

  - En el paso k+1

$$p_{k+1} = 2\,\delta y\, x_{k+1} - 2\,\delta x\, y_{k+1} + c$$

  - Restando

$$p_{k+1} - p_k = 2\,\delta y\,(x_{k+1} - x_k) - 2\,\delta x\,(y_{k+1} - y_k)$$

Cómo dibujar una línea?

- Algoritmo de Bresenham (m < 1)

$$p_{k+1} = p_k + 2\delta y (x_{k+1} - x_k) - 2\delta x (y_{k+1} - y_k)$$

  - Pero dado que $x_{k+1} - x_k = 1$

$$p_{k+1} = p_k + 2\delta y - 2\delta x (y_{k+1} - y_k)$$

  - Y dependiendo del signo de p_k, $(y_{k+1} - y_k)$ es 0 o 1

  - Valor inicial de $p_0 = 2\delta y - \delta x$

Cómo dibujar una línea?

Recordar que si $p < 0$ => solo se incremente x
Si $p >= 0$ se incrementa x e y

- lineBresenham(x0, y0, xf, yf)
  - setPixel(x0, y0)
  - p := 2δy - δx
  - x := x0; y := y0
  - while x < xf
    - if p<0
      p := p + 2δy
      else
      y := y+1
      p := p + 2δy - 2δx
    - x := x+1
    - setPixel(x, y)

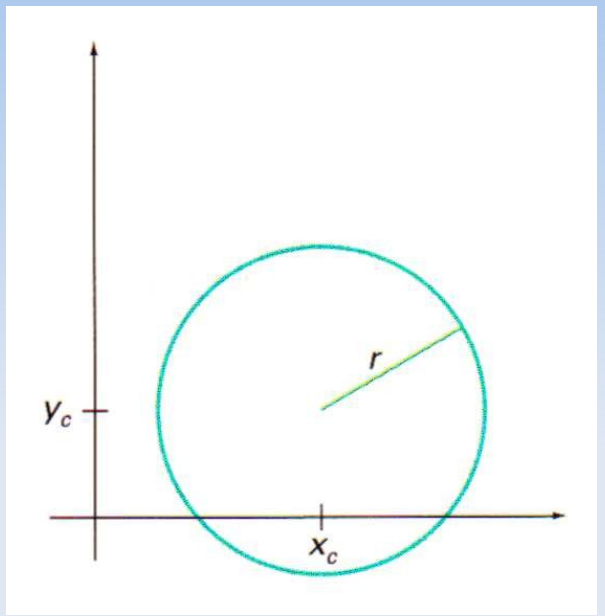
Propuesto: extenderlo a los otros casos.

Cómo dibujar un círculo?

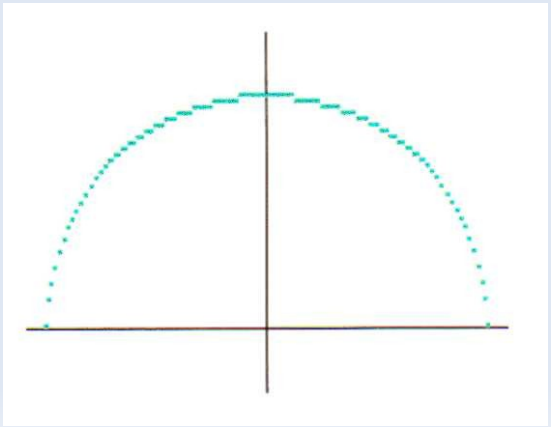
- Por qué no usar la conocida fórmula:

$$(x - x_c)^2 + (y - yc)^2 = r^2$$

$$y = y_c \pm \sqrt{r^2 - (x_c - x)^2}$$



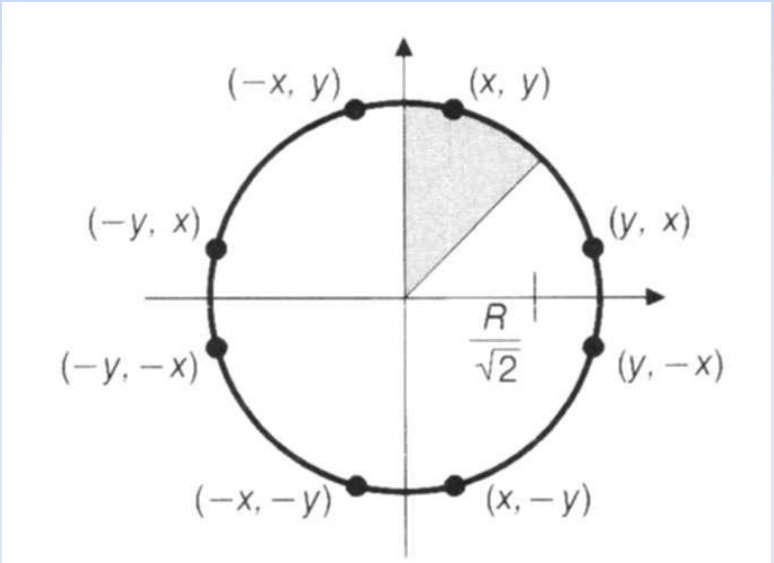
  - Usa operaciones “pesadas”
  - El espacio entre pixeles dibujados no uniforme



Cómo dibujar un círculo?

- Usando un espaciado angular, un círculo puede ser dibujado con puntos equidistantes

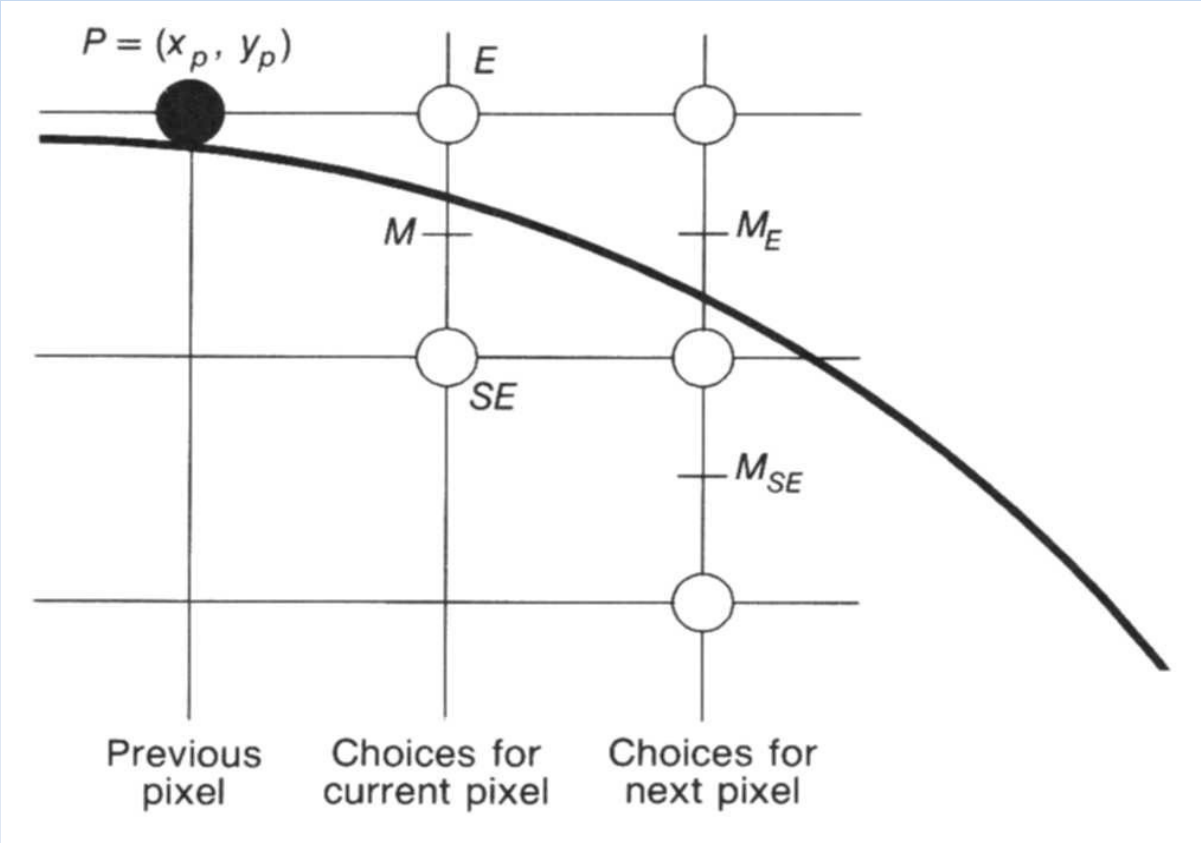
$$x = x_c + r\cos\theta$$
$$y = y_c + r\,\mathrm{sen}\,\theta$$



- Simetría en el cálculo del círculo: se calcula una posición en un octante y se obtiene los siete puntos restantes

Cómo dibujar un círculo?

- MidPoint Circle algorithm



- Propuesto: mirar la implementación