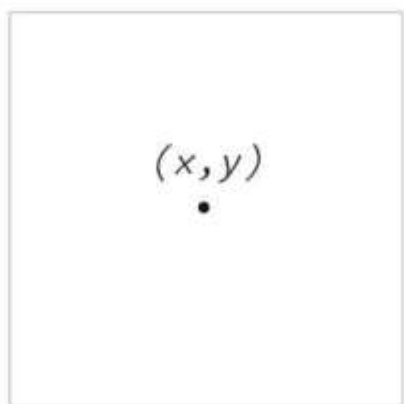
Introducción a la Computación Gráfica

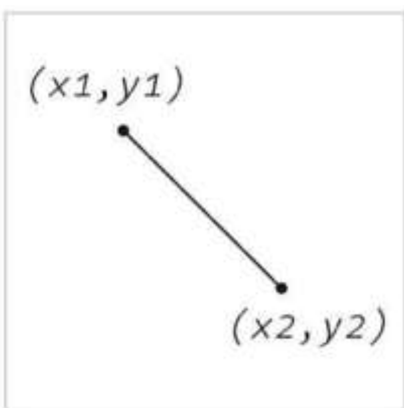
Ing. Gabriel Ávila, MSc.

En computación gráfica, se conoce como ***primitivas*** a las figuras más básicas que se pueden representar en un sistema gráfico.

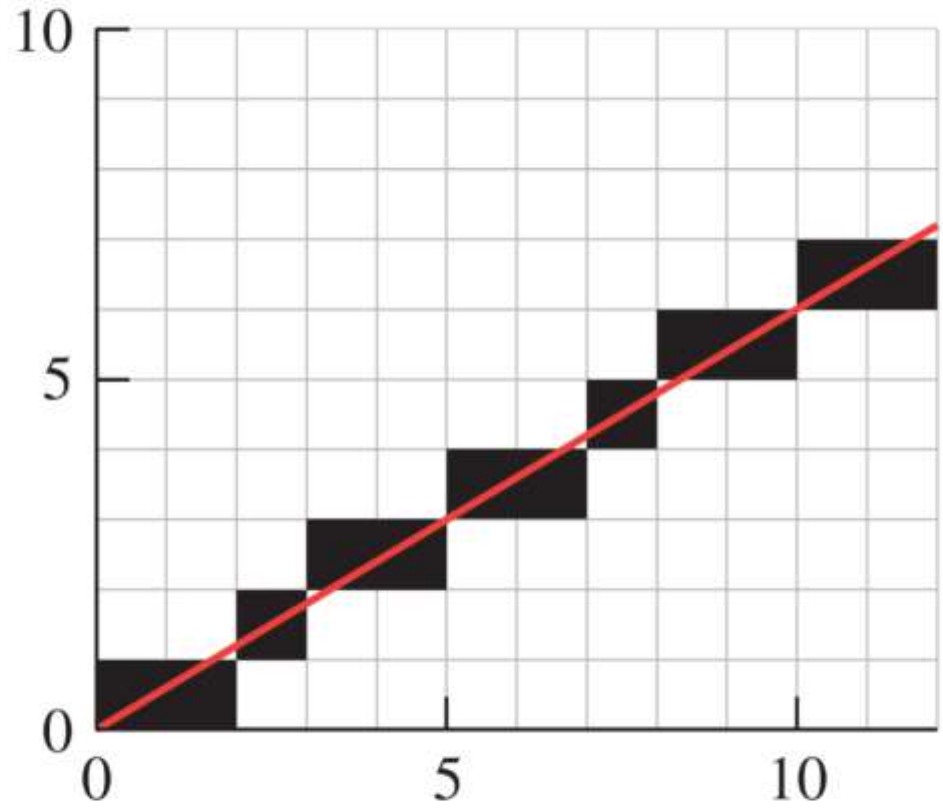
Se trata de objetos básicos, esenciales para la creación de objetos más complejos.

La más básica de las primitivas es el punto, a partir del cual se pueden graficar líneas, curvas, etc.





Primitivas gráficas



Líneas

PRIMITIVAS 2D

Basadas en la aproximación, pixel a pixel, de una recta.

Ecuación de la recta (punto-pendiente):

$$y(x) = mx + b$$

Dados dos puntos (x_0, y_0) y (x_f, y_f):

$$m = \frac{y_f - y_0}{x_f - x_0} \qquad b = y_0 - mx_0$$

Intercepto con el eje y

Líneas

Discretización:

Partiendo de la ecuación de la pendiente y reemplazando el diferencial de las posiciones en x y y:

$$m = \frac{y_f - y_0}{x_f - x_0} \rightarrow m = \frac{\delta y}{\delta x}$$

Se obtiene que las posiciones correspondientes sólo dependen de la pendiente (m):

$$\delta y = m\delta x \qquad \delta x = \frac{\delta y}{m}$$

Líneas

DDA ***(Digital Diferential Analizer)***: usado para la interpolación de variables, entre un punto de inicio y uno final. Usados en el proceso de rasterización (discretización) de líneas, triángulos y polígonos.

Dado un diferencial δx=1 o δy=1, es más directo trabajar con las ecuaciones anteriores.

Para una línea con pendiente positiva:

Si $m \leq 1$, se hacen intervalos unitarios en x ($\delta x = 1$):

$$y_{k+1} = y_k + m$$

Si $m > 1$, se hacen intervalos unitarios en y ($\delta y = 1$):

$$x_{k+1} = x_k + \frac{1}{m}$$

Algoritmo DDA

1. Calcule manualmente, usando DDA, los puntos que conforman las líneas entre:

   (5, 10) y (10, 20)

   (10,20) y (18, 24)

2. Realice un programa en Processing que dibuje líneas utilizando el método explicado.
3. ¿Qué sucede si la pendiente es negativa?

Taller

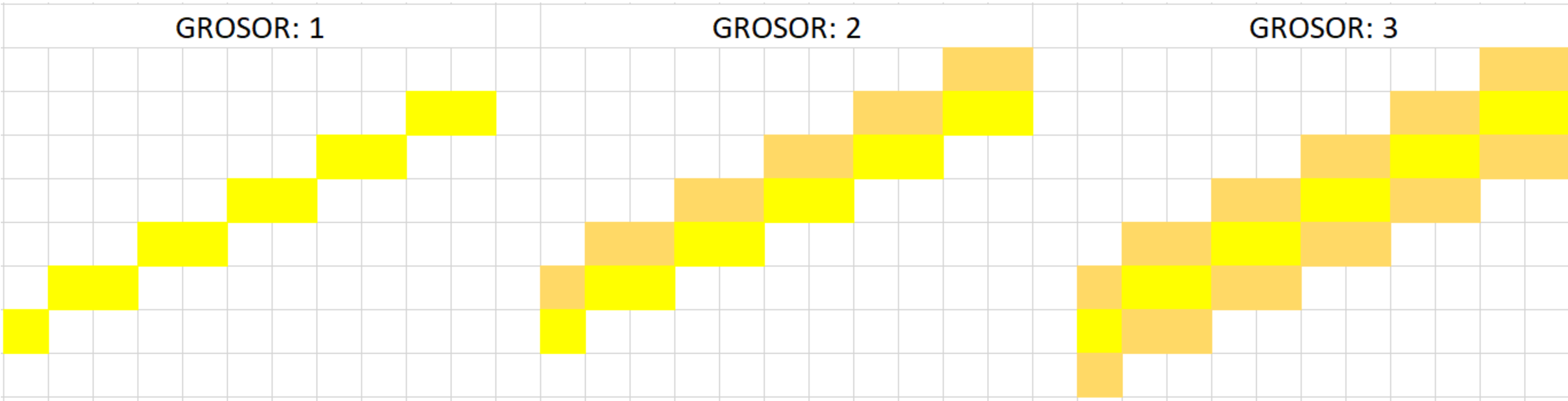
Se puede lograr grosor mediante diferentes métodos:

- Extensión vertical de pixeles ($|m| < 1$).
- Extensión horizontal de pixeles ($|m| \geq 1$).
- Uso de plumillas.

Grosor

Consiste en redibujar la línea, una cantidad determinada de pixeles por encima y por debajo.

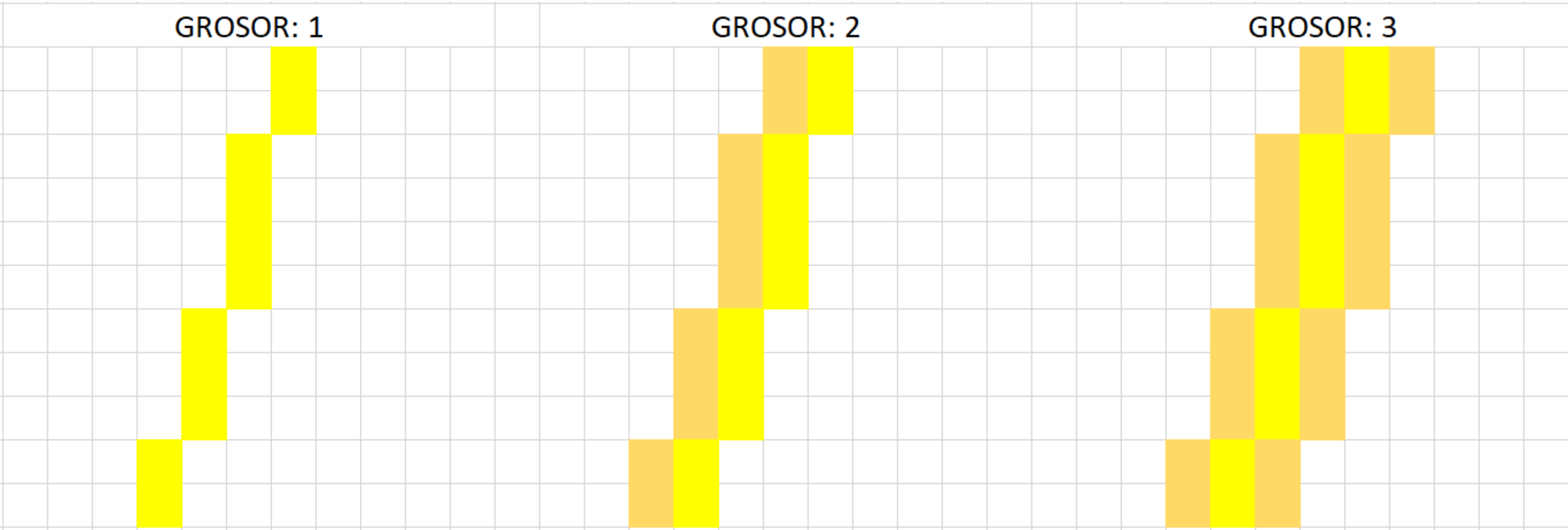
Esta cantidad está dada por el grosor de la línea.



Extensión vertical

Consiste en redibujar la línea, una cantidad determinada de pixeles a la izquierda y a la derecha.

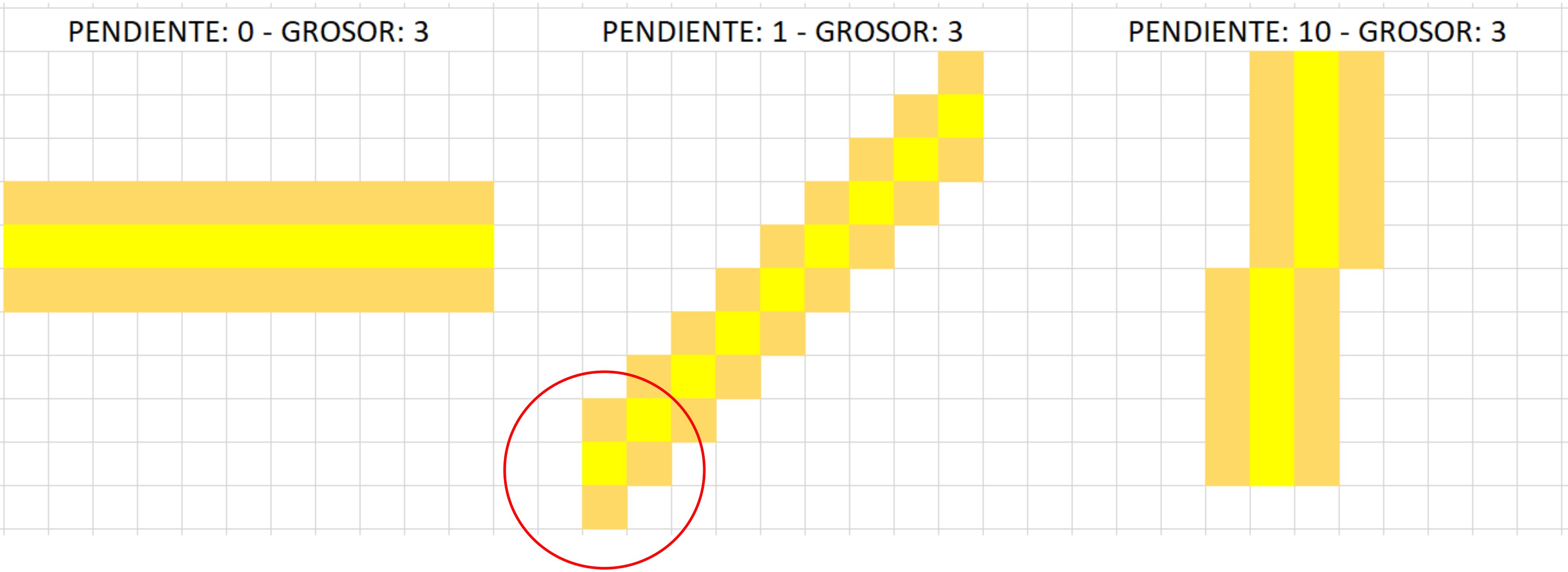
Esta cantidad está dada por el grosor de la línea.



Extensión horizontal

Inconvenientes:

- El grosor que se percibe de la línea depende de la pendiente.
- Los extremos de la línea dependen de la extensión utilizada.



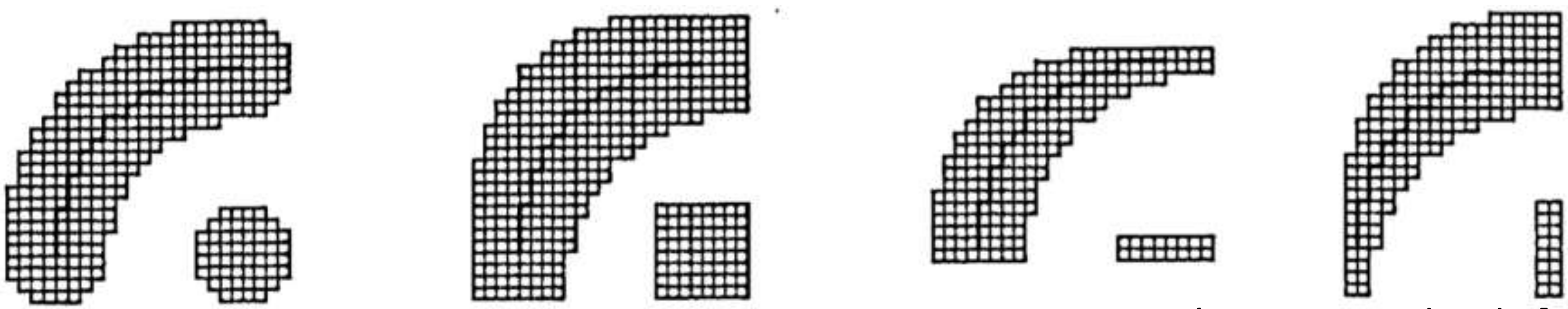
Extensiones

Plumillas - *Brushes*

Consiste en seguir el contorno de la línea, utilizando una plumilla predefinida. El mayor inconveniente con este método radica en que se redibujan zonas previamente dibujadas. A mayor área, mayores pixeles redibujados.

Plumillas:

Resultado:

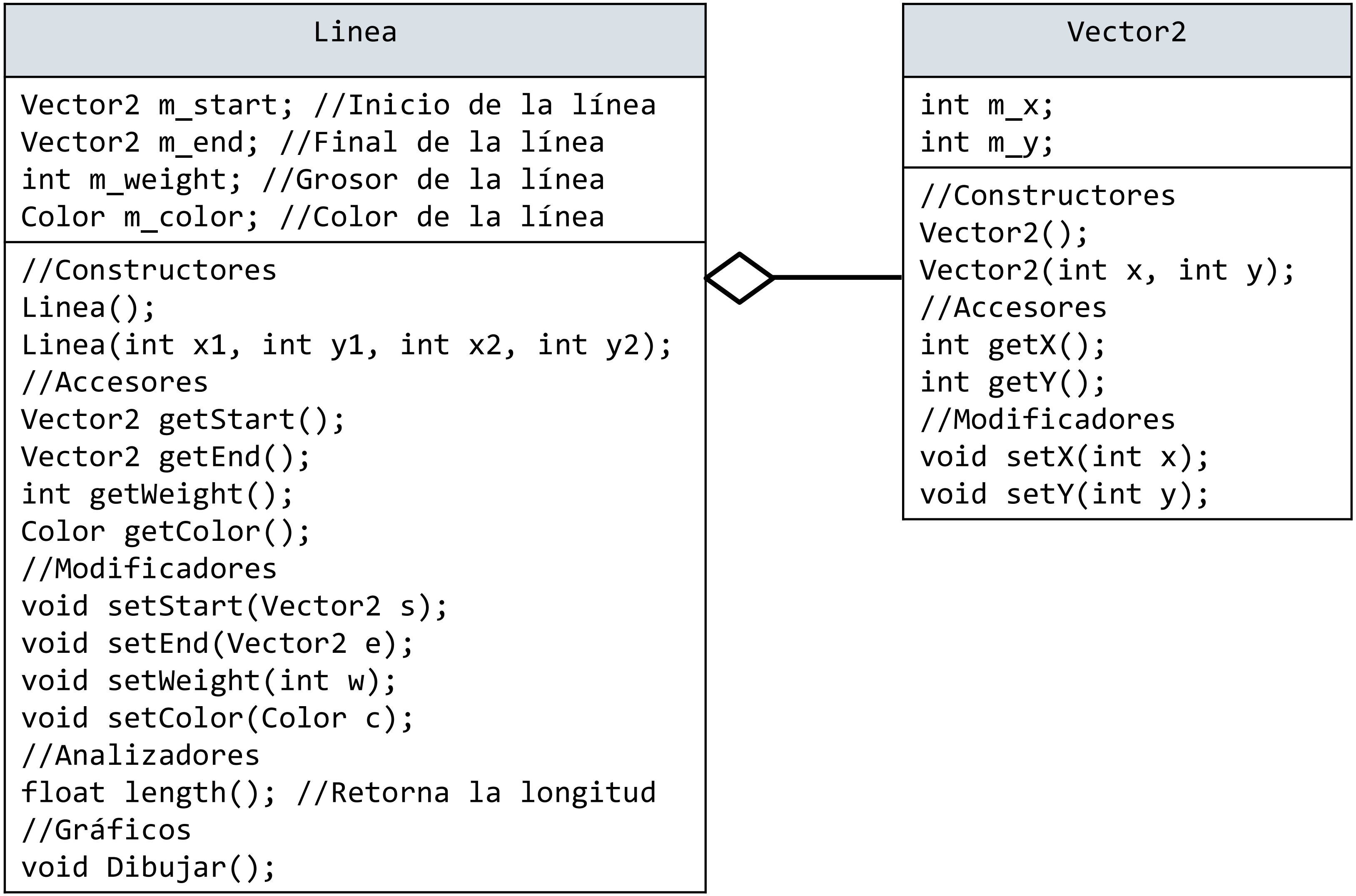


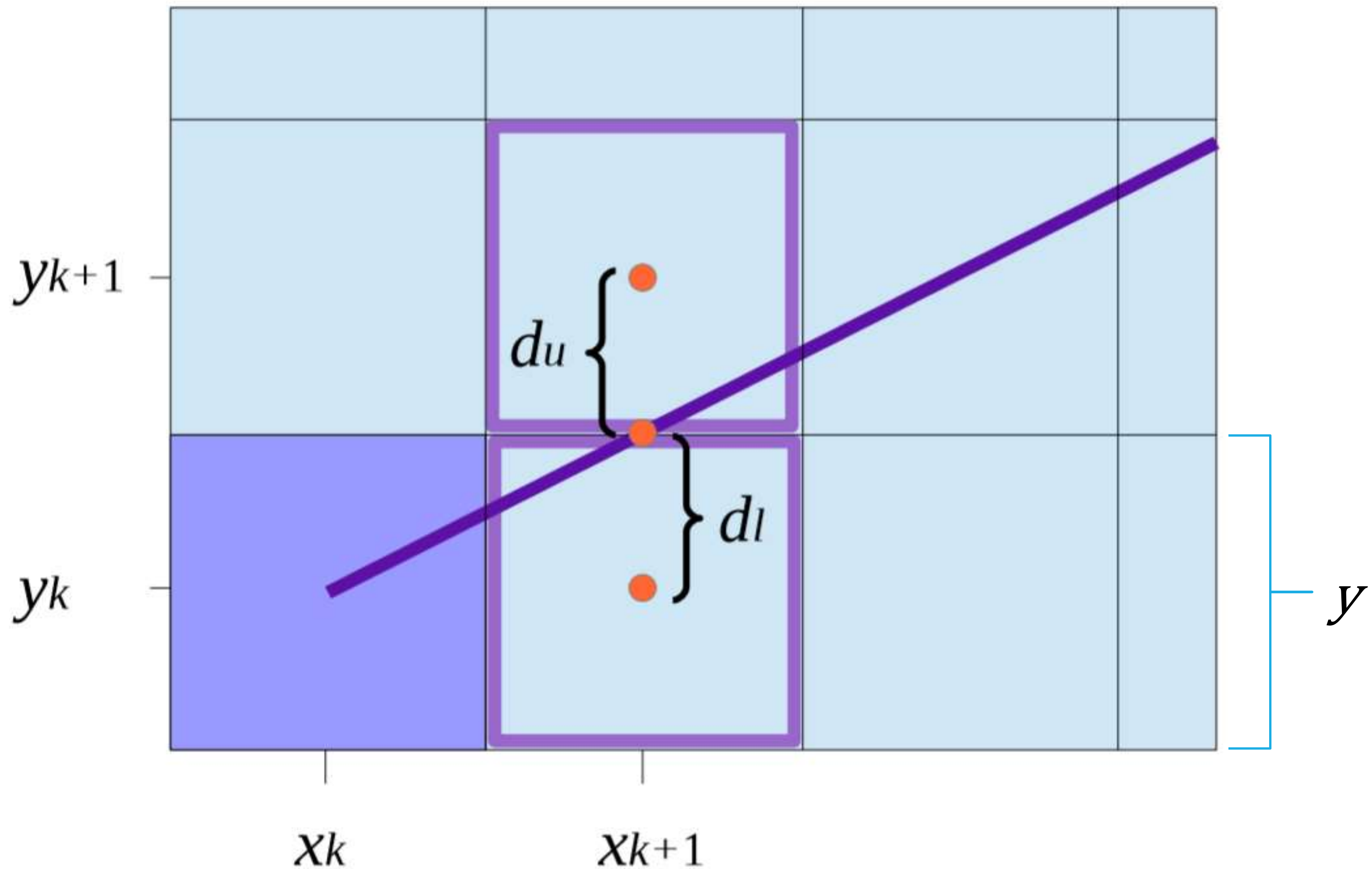
Imágenes tomadas de [2]

1. Investigar el algoritmo de Bresenham.

2. Implementar la clase Línea:
   - Dibujar una línea dados 2 puntos.
   - Dibujar una línea, dada su pendiente y punto de corte.
   - Dibujar una línea con diferentes grosores.

Tareas


Linea
Vector2 m_start; //Inicio de la línea
Vector2 m_end; //Final de la línea
int m_weight; //Grosor de la línea
Color m_color; //Color de la línea
//Constructores
Linea();
Linea(int x1, int y1, int x2, int y2);
//Accesores
Vector2 getStart();
Vector2 getEnd();
int getWeight();
Color getColor();
//Modificadores
void setStart(Vector2 s);
void setEnd(Vector2 e);
void setWeight(int w);
void setColor(Color c);
//Analizadores
float length(); //Retorna la longitud
//Gráficos
void Dibujar();
Vector2
int m_x;
int m_y;
//Constructores
Vector2();
Vector2(int x, int y);
//Accesores
int getX();
int getY();
//Modificadores
void setX(int x);
void setY(int y);



Tomado de [1]

Algoritmo de Bresenham

A partir de la ecuación de la recta, el punto $\boldsymbol{y}$ se calcula como:

$$\boldsymbol{y = m(x_k + 1) + b}$$

Separación inferior d_l:

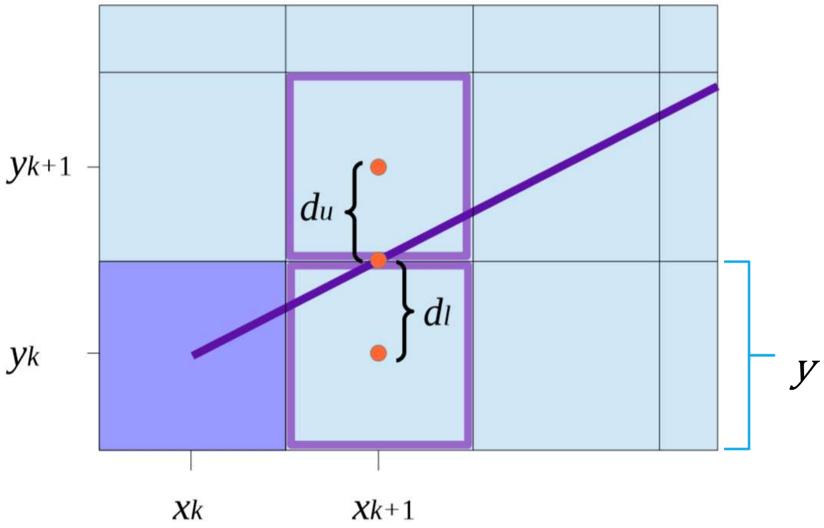
$$d_l = y - y_k$$

$$\boldsymbol{d_l = m(x_k + 1) + b - y_k}$$

Separación superior d_u:

$$d_u = y_{k+1} - y$$

$$\boldsymbol{d_u = y_{k+1} - m(x_k + 1) - b}$$



Algoritmo de Bresenham

$$d_l - d_u = 2m(x_k + 1) - 2y_k + 2b - 1$$

$$d_l - d_u = 2m \cdot x_k + 2m - 2y_k + 2b - 1$$

$$\boldsymbol{d_l - d_u = 2m \cdot x_k - 2y_k + c}$$

Teniendo en cuenta que:

$$m = \frac{\Delta y}{\Delta x}, \qquad c = 2m + (2b - 1) = 2\Delta y + \Delta x(2b - 1)$$

El ***criterio de decisión***, se basa en las separaciones calculadas:

$$\boldsymbol{p_k = \Delta x(d_l - d_u) = 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c}$$

Algoritmo de Bresenham

Criterio de decisión:

$$\boldsymbol{p_k = 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c}$$

$p_k < 0 \rightarrow$ Se toma el pixel inferior (y_k)

$p_k \geq 0 \rightarrow$ Se toma el pixel superior ($y_k + 1$)

Para los pasos sucesivos,

$$\boldsymbol{p_{k+1} = 2\Delta y \cdot x_{k+1} - 2\Delta x \cdot y_{k+1} + c}$$

Restando y simplificando:

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

Despejando:

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

Algoritmo de Bresenham

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

Donde $(\boldsymbol{y_{k+1} - y_k})$ es 0 o 1, dependiendo del criterio de decisión $\boldsymbol{p_k}$:

$p_k < 0 \rightarrow (y_{k+1} - y_k) = 0$

$p_k \geq 0 \rightarrow (y_{k+1} - y_k) = 1$

En el primer parámetro $\boldsymbol{p_0}$ se calcula como:

$$\boldsymbol{p_0 = 2\Delta y - \Delta x}$$

Algoritmo de Bresenham

Paso 1: Obtenga los dos puntos extremos de la línea. El punto izquierdo será $(\boldsymbol{x_0}, \boldsymbol{y_0})$.

Paso 2: Grafique $(\boldsymbol{x_0}, \boldsymbol{y_0})$.

Paso 3: Calcule las constantes $\boldsymbol{\Delta x}$, $\boldsymbol{\Delta y}$, $\boldsymbol{2\Delta y}$ y $\boldsymbol{2\Delta y - 2\Delta x}$.

Paso 4: Obtenga el primer parámetro de decisión:

$$\boldsymbol{p_0 = 2\Delta y - \Delta x}$$

Paso 5: Para cada Xi a lo largo de la línea, empezando por k=0, verifique:

- Si $p_k < 0$

$$\boldsymbol{p_{k+1} = p_k + 2\Delta y} \text{ y se mantiene } y_k$$

De lo contrario:

$$\boldsymbol{p_{k+1} = p_k + 2\Delta y - 2\Delta x} \text{ y se usa } y_{k+1}$$

Paso 6: Repita el paso $(\boldsymbol{\Delta x - 1})$ veces, hasta llegar al punto final.

Algoritmo de Bresenham

En una hoja de papel cuadriculada, graficar la línea que hay entre los puntos:

(7, 8) y (15, 13)

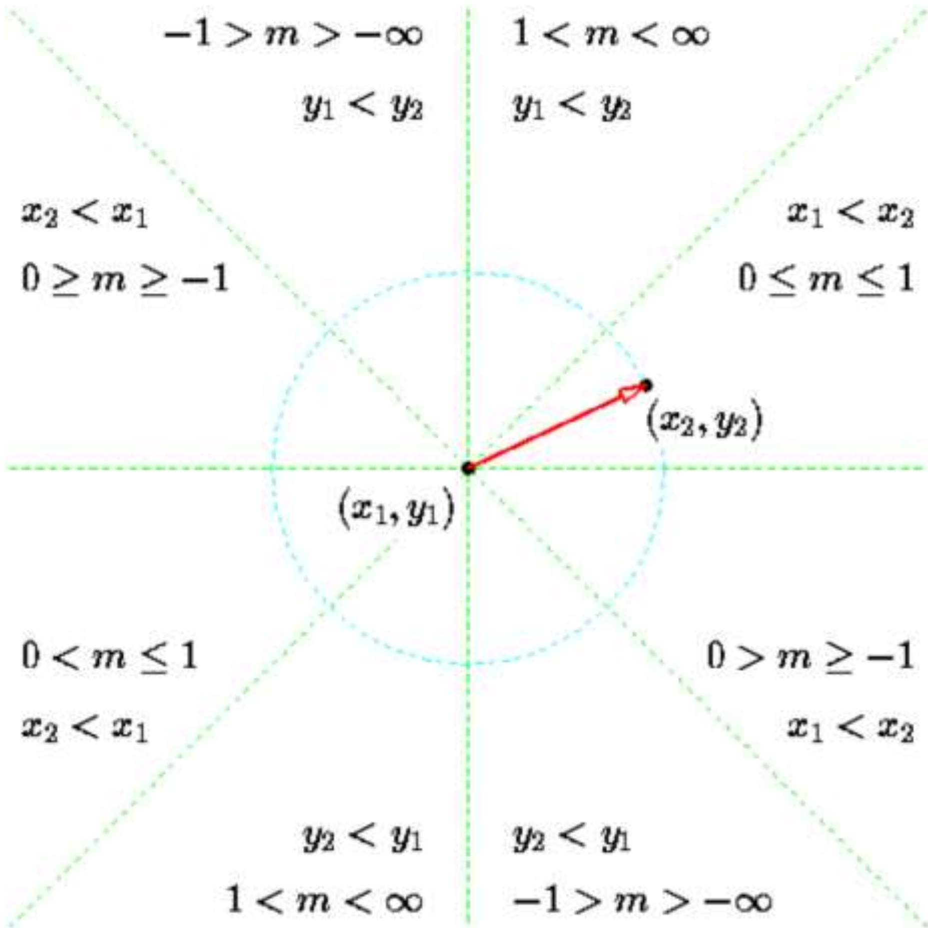
(5, 2) y (8, 8)

Utilizando DDA y Bresenham.

Ejercicio

El algoritmo básico de Bresenham funciona para uno de 8 octantes.

Para los demás es necesario aplicar simetría o modificar el algoritmo.



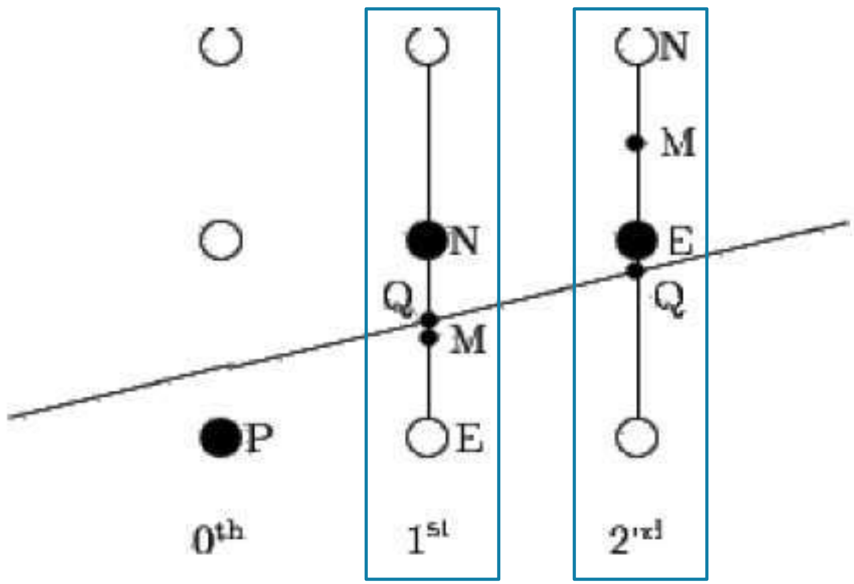
Tomado de: https://www.cs.helsinki.fi/group/goa/mallinnus/lines/bresenh.html

Octantes

Algoritmo del punto medio

Se trata de una modificación al algoritmo de Bresenham.

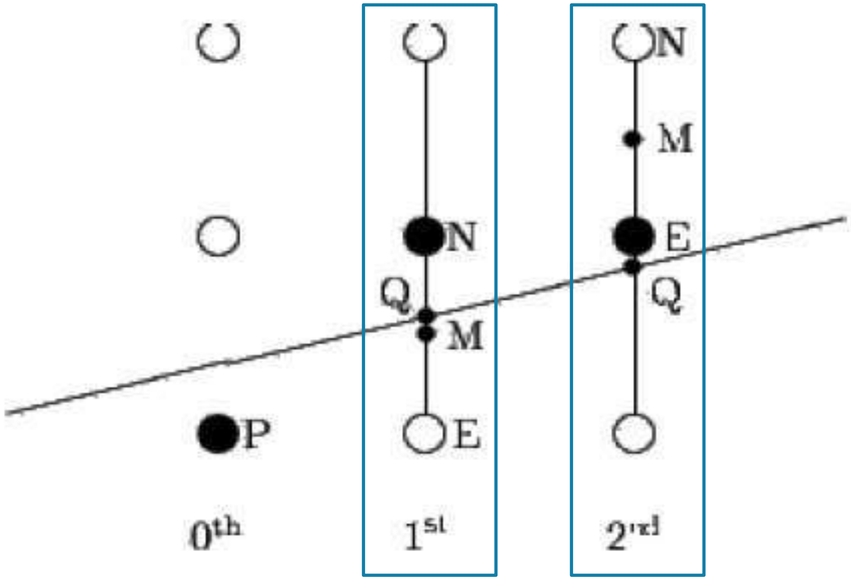
A partir de un punto **P**, para una línea con pendiente $0 \leq m \leq 1$, se desea saber si el siguiente punto a graficar es **N** o **E**. El criterio de evaluación será el punto de intersección **Q**, dependiendo de si está más cerca de **N** o de **E**.



Tomado de: https://www.tutorialspoint.com/computer_graphics/line_generation_algorithm.htm

Para determinar esto, se calcula el punto medio: $\mathrm{M}\left(x+1, y+\frac{1}{2}\right)$

Si el punto de intersección **Q** de la línea con la vertical para cada punto **E** o **N** está por debajo de **M**, tome **E** como siguiente punto, de lo contrario, tome **N**.



Tomado de: https://www.tutorialspoint.com/computer_graphics/line_generation_algorithm.htm

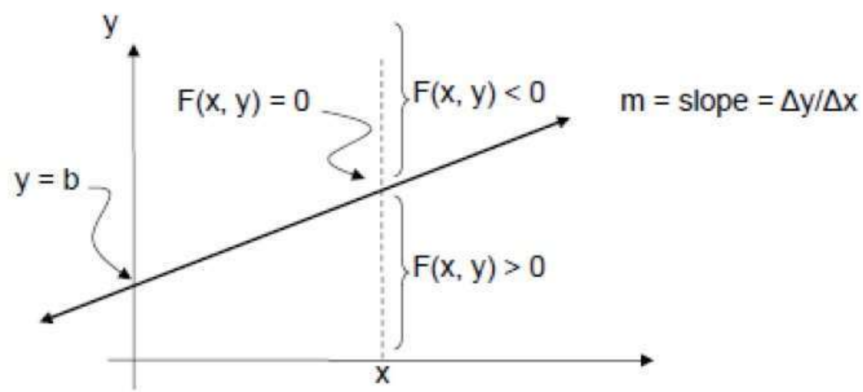
Algoritmo del punto medio

Es necesario evaluar con base a la ecuación implícita de la recta:

$$f(x, y) = mx + b - y$$

Para $m > 0$, dado cualquier x :

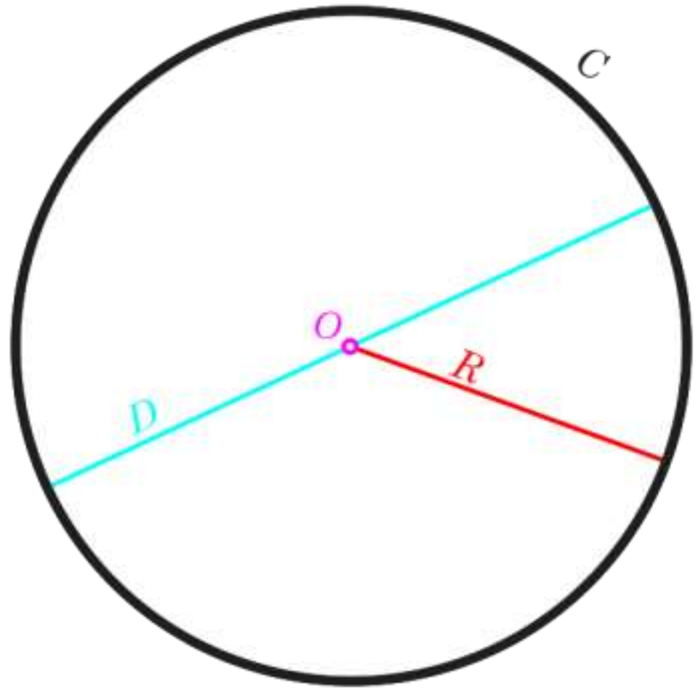
- Si y está sobre la línea, entonces $f(x, y) = 0$.
- Si y está por encima de la línea, entonces $f(x, y) < 0$.
- Si y está por debajo de la línea, entonces $f(x, y) > 0$.



Tomado de: https://www.tutorialspoint.com/computer_graphics/line_generation_algorithm.htm

Algoritmo del punto medio

Círculos



Tomado de: https://en.wikipedia.org/wiki/Circle#/media/File:Circle-withsegments.svg

Un círculo consiste en un conjunto de puntos que se encuentran a una distancia r de una posición central (x_c, y_c):

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

Si x aumenta con pasos unitarios,

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

Círculo

También es posible representar un círculo mediante la ecuación:

$$f_{cir}(x, y) = (x - x_c)^2 + (y - y_c)^2 - r^2$$

Donde, dado un par de coordenadas x, y y un radio:

- $f_{cir}(x, y) < 0$ entonces el punto (x, y) está dentro del círculo.
- $f_{cir}(x, y) = 0$ entonces el punto (x, y) está en la circunferencia.
- $f_{cir}(x, y) > 0$ entonces el punto (x, y) está fuera del círculo.

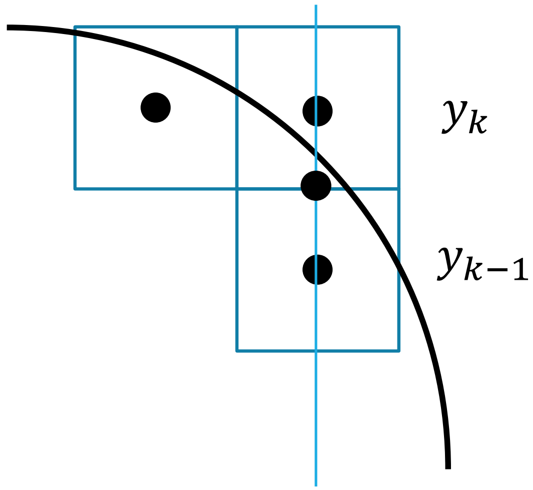
Círculo

Dado un parámetro de decisión $\boldsymbol{p_k}$, evaluado en el punto medio entre y_k y y_{k+1}:

$$\boldsymbol{p_k} = f_{cir}\left(x_{k+1}, y_k - \frac{1}{2}\right) = (x_{k+1})^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2$$

$p_k < 0 \rightarrow$ Se toma el pixel superior (y_k)

$p_k \geq 0 \rightarrow$ Se toma el pixel inferior $(y_k - 1)$



Algoritmo del punto medio

Se calcula para $\boldsymbol{p_{k+1}}$:

$$\mathrm{p_{k+1}} = \mathrm{f_{cir}}\left(\mathrm{x_{k+1}} + 1, \mathrm{y_{k+1}} - \frac{1}{2}\right) = [(\mathrm{x_{k+1}}) + 1]^2 + \left(\mathrm{y_{k+1}} - \frac{1}{2}\right)^2 - \mathrm{r}^2$$

$$p_{k+1} = p_k + 2(x_k + 1) + ({y_{k+1}}^2 - {y_k}^2) + (y_{k+1} - y_k) + 1$$

Cálculos incrementales:

$p_k < 0 \rightarrow p_{k+1} = p_k + 2(x_{k+1}) + 1$

$p_k \geq 0 \rightarrow p_{k+1} = p_k + 2(x_{k+1}) + 1 - 2 \cdot y_{k+1}$

Algoritmo del punto medio

Parámetro de decisión inicial se evalúa en $(x_0, y_0) = (0, r)$:

$$p_0 = f_{cir}\left(1, r - \frac{1}{2}\right) = 1 + \left(r - \frac{1}{2}\right)^2 - \mathrm{r}^2$$

$$p_0 = \frac{5}{4} - \mathrm{r}$$

Cálculos incrementales:

$p_k < 0 \rightarrow p_{k+1} = p_k + 2(x_{k+1}) + 1$

$p_k \geq 0 \rightarrow p_{k+1} = p_k + 2(x_{k+1}) + 1 - 2 \cdot y_{k+1}$

Algoritmo del punto medio

Dados el centro del círculo (x_c, y_c) y r:

Paso 1: Establecer las coordenadas del primer punto, llevado al origen.

$$(x - x_c, y - y_c) = (0, r)$$

Paso 2: Calcular el parámetro de decisión inicial:

$$p_0 = \frac{5}{4} - r$$

Paso 3: Para x_k, comenzando en k=0

- Si $p_k < 0$ dibujar(x_{k+1}, y_k) y calcular:
$$\boldsymbol{p_{k+1} = p_k + 2(x_{k+1}) + 1}$$
- Si $p_k \geq 0$ dibujar(x_{k+1}, y_{k-1}) y calcular:
$$\boldsymbol{p_{k+1} = p_k + 2(x_{k+1}) + 1 - 2 \cdot y_{k+1}}$$

Paso 4: Repetir el paso 3, mientras que $x \geq y$

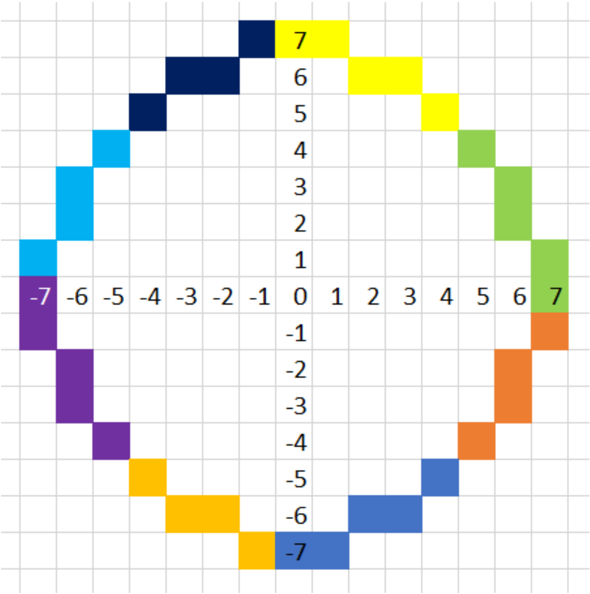
Algoritmo del punto medio

Paso 5: Determinar, por simetría, los puntos en los 7 octantes restantes.

Paso 6: Trasladar cada (x, y) al círculo centrado en (x_c, y_c):

$$x = x + x_c$$
$$y = y + y_c$$

k	Pk	Pk+1	Xk	Yk
0	-5,75	-2,75	0	7
1	-2,75	2,25	1	7
2	2,25	-7,75	2	6
3	-7,75	1,25	3	6
4	1,25	-2,75	4	5
5	-2,75	10,3	5	5

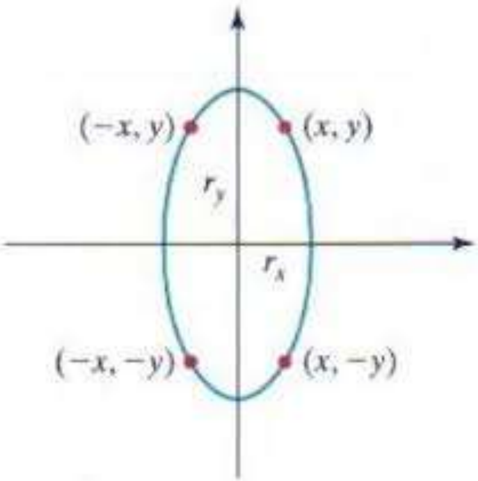


Algoritmo del punto medio

Una elipse consiste en un conjunto de puntos que se encuentran a una distancia rx del centro, en el semieje horizontal, y una distancia ry del centro, en el semieje vertical:

$$\left(\frac{x - x_c}{r_x}\right)^2 + \left(\frac{y - y_c}{r_y}\right)^2 = 1$$

Para graficar una elipse, el algoritmo es similar al explicado previamente, teniendo en cuenta que la simetría no es por octantes, sino por cuadrantes.



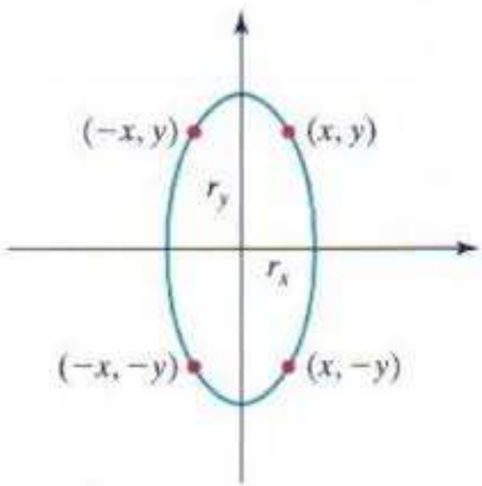
Elipses

Una elipse consiste en un conjunto de puntos que se encuentran a una distancia rx del centro, en el semieje horizontal, y una distancia ry del centro, en el semieje vertical:

$$\left(\frac{x - x_c}{r_x}\right)^2 + \left(\frac{y - y_c}{r_y}\right)^2 = 1$$

Para graficar una elipse, el algoritmo es similar al explicado previamente, teniendo en cuenta que la simetría no es por octantes, sino por cuadrantes.

https://www.cpp.edu/~raheja/CS445/MEA.pdf



Elipses

Hacer una aplicación tipo “Paint” que permita dibujar:

- Puntos.
- Líneas.
- Círculos.
- Elipses.
- Rectángulos.
- Polígonos.

Teniendo en cuenta: color de borde y grosor.

La entrega se hará en grupos de máximo 4 personas.

La fecha de entrega será el **5 de octubre** (Usando funciones de Processing).

La fecha de entrega será el **16 de octubre** (Usando los algoritmos de clase).

TAREA

[1] Rueda, A. (2014). ***Primitivas 2D – Líneas y Curvas***. Pontificia Universidad Javeriana.

[2] Beat, S. (1989). ***Algorithms for drawing thick lines and curves on raster devices***. Tomado de: https://doi.org/10.3929/ethz-a-000505295

[3] Flanagan, C. The Bresenham Line-Drawing Algorithm. Tomado de: https://www.cs.helsinki.fi/group/goa/mallinnus/lines/bresenh.html

[4]Tutorials point. Line generation algorithm. https://www.tutorialspoint.com/computer_graphics/line_generation_algorithm.htm

Bibliografía

[5]Tutorials point. Circle generation algorithm. https://www.tutorialspoint.com/computer_graphics/circle_generation_algorithm.htm

Bibliografía