Corso di *Tecniche Avanzate per la Grafica*

Algoritmi di Rendering di Base

Docente:
Massimiliano Corsini

Laurea Specialistica in Informatica

Facoltà di Scienze MM. FF. NN.
Università di Ferrara

Overview

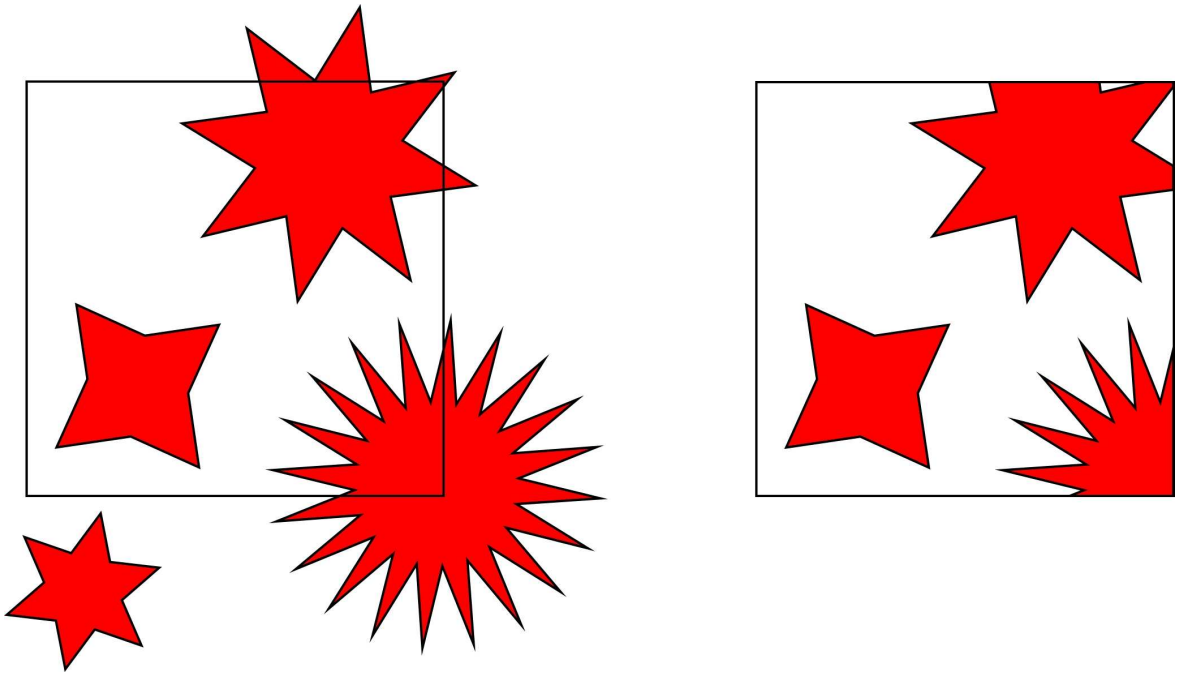
- Clipping
  - Clipping di punti e segmenti
    - Algoritmo di Cohen-Sutherland
    - Algoritmo di Liang-Barsky
  - Clipping di poligoni (generalità)
- Rimozione delle superfici nascoste
  - Approccio object-space e image-space
  - Back-face culling
  - L’algoritmo *z-buffer*
  - L’algoritmo del pittore *(depth-sort)*

Overview

- Rasterizzazione;
  - Rasterizzazione di segmenti;
  - L’algoritmo di Bresenham per i segmenti;
  - Filling di poligoni (cenni)

Clipping

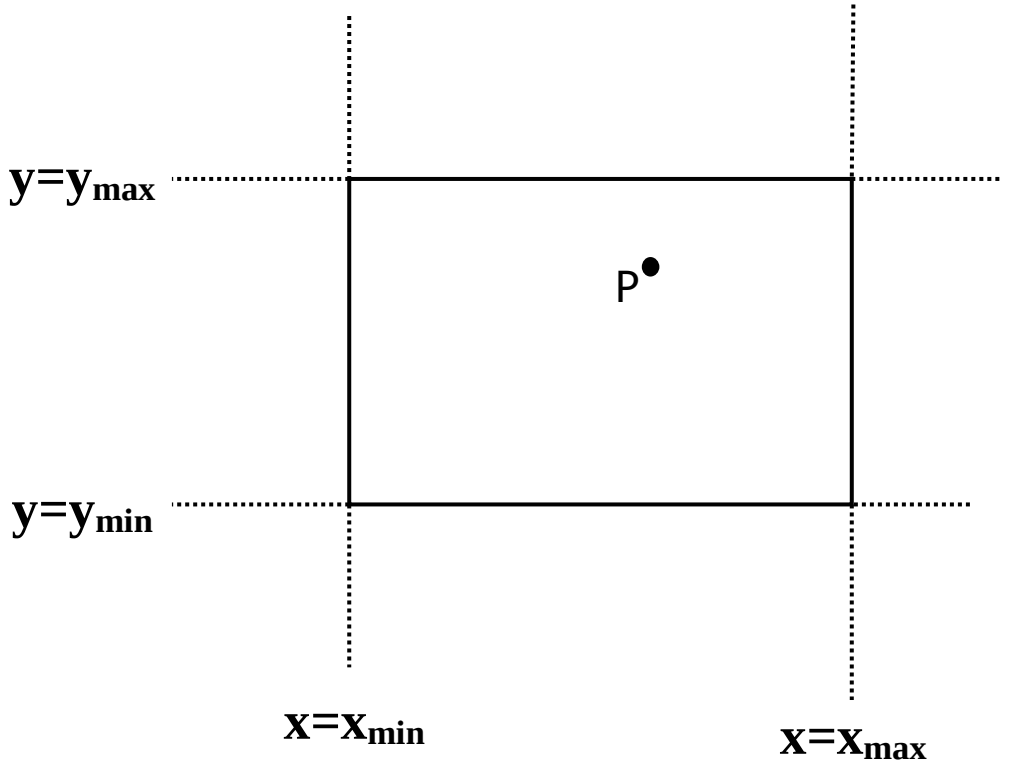
- L'operazione di *clipping* consiste nell'individuare (e rimuovere) le primitive grafiche (o parti di esse) esterne ad una finestra rettangolare o esaedrale oppure, più in generale, esterne ad un poligono o poliedro convesso.
- Solitamente si è interessati al clipping rispetto a rettangoli o esaedri



Clipping di un punto

- Clipping di un punto: un punto è all'interno del rettangolo di clipping se e solo se sono soddisfatte le 4 disuguaglianze:

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

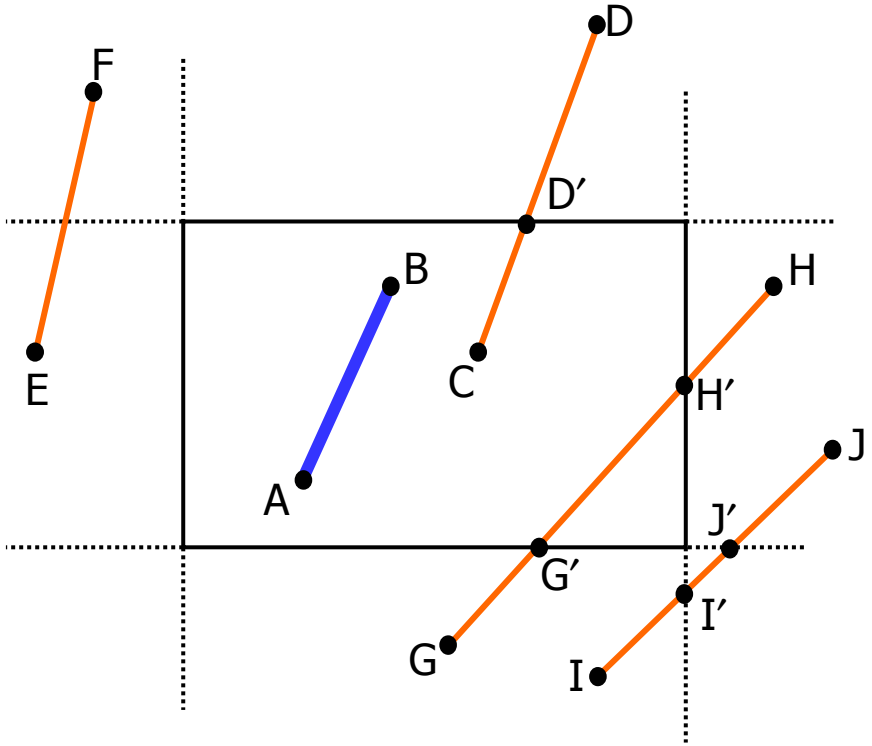


Clipping di un segmento

- Clipping di un segmento: necessario analizzare le posizioni dei suoi punti estremi.
  - Se gli estremi sono entrambi interni al rettangolo di clipping, il segmento è interno;
  - Se un estremo è interno e l'altro esterno, allora il segmento interseca il rettangolo di clipping ed è necessario determinare l'intersezione;
  - Se entrambi gli estremi sono esterni al rettangolo, il segmento può intersecare o meno il rettangolo di clipping e si rende necessaria una analisi più accurata per individuare le eventuali parti interne del segmento.

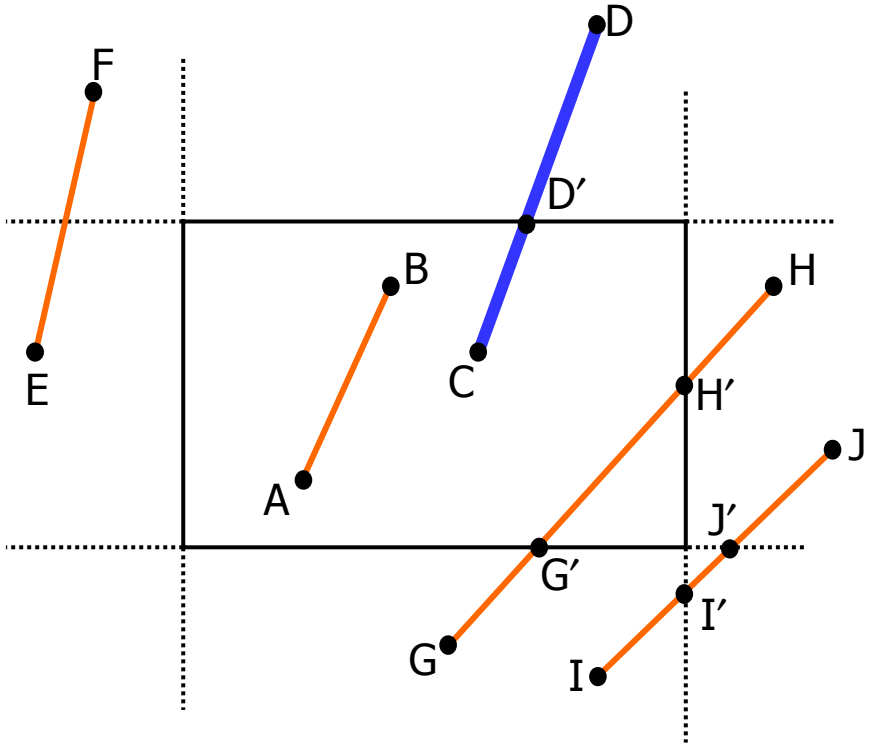
Clipping di un segmento

- Se gli estremi sono entrambi interni al rettangolo di clipping, il segmento (AB) è interno.



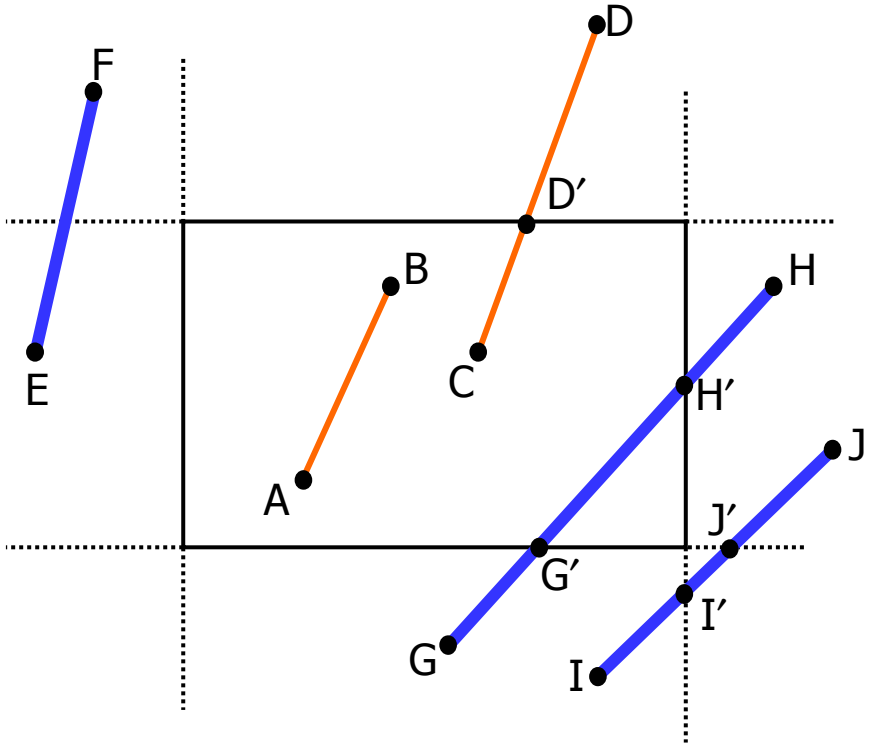
Clipping di un segmento

- Se un estremo è interno e l'altro esterno, allora il segmento interseca il rettangolo di clipping ed è necessario determinare l'intersezione (CD)



Clipping di un segmento

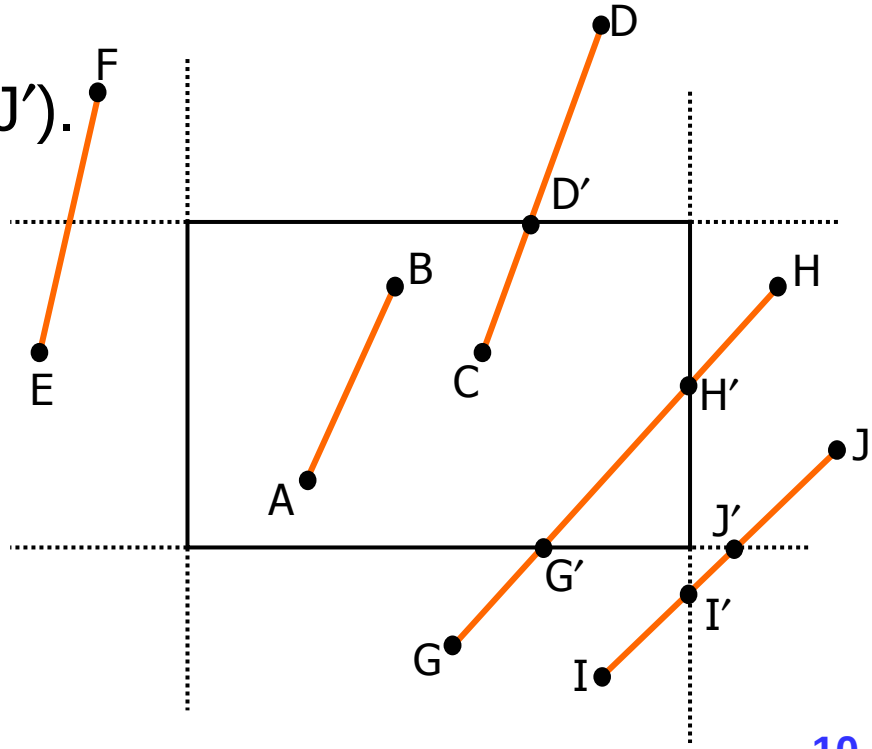
- Se entrambi gli estremi sono esterni al rettangolo, il segmento può intersecare o meno il rettangolo di clipping e si rende necessaria una analisi più accurata per individuare le eventuali parti interne del segmento (EF, GH, IJ).



Clipping di un segmento
Approccio diretto

- L'approccio diretto alla soluzione del problema è quello di determinare le intersezioni tra la retta su cui giace il segmento e le 4 rette su cui giacciono i lati del rettangolo di clipping;
- Individuati i punti di intersezione occorre verificare l'effettiva appartenenza al rettangolo di clipping (G′ e H′) o meno (I′ e J′).
- Le intersezioni si determinano mediante l'eq. parametrica dei segmenti relativi.

$$x = x_a + t(x_b - x_a)$$
$$y = y_a + t(y_b - y_a)$$
$$t \in [0,1]$$



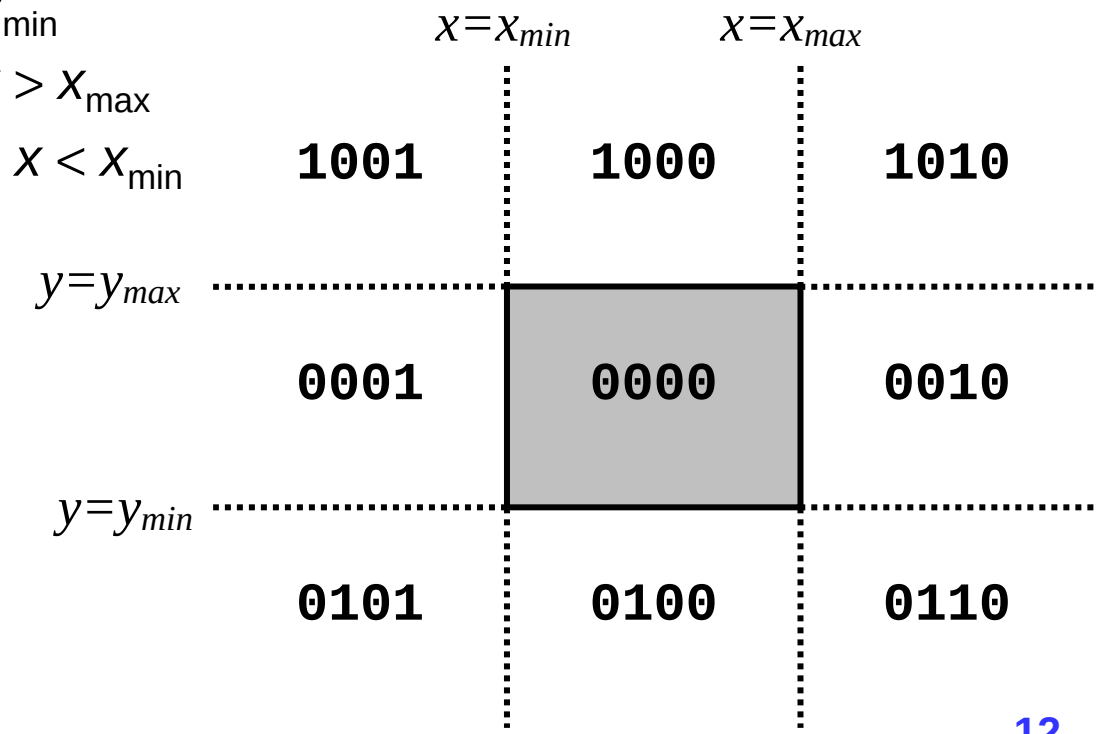
Clipping di un segmento
Approccio diretto

- Per ogni coppia segmento-lato di rettangolo si risolve il sistema di equazioni parametriche che definiscono il segmento in funzione di t_{segm} ed il lato in funzione di t_{lato};
- Se t_{segm} e t_{lato} assumono valori nell'intervallo [0, 1] allora l'intersezione appartiene al segmento ed al rettangolo di clipping;
- E' necessario verificare in anticipo il parallelismo tra le linee prima di determinare l'intersezione;

- Algoritmo costoso e quindi inefficiente.

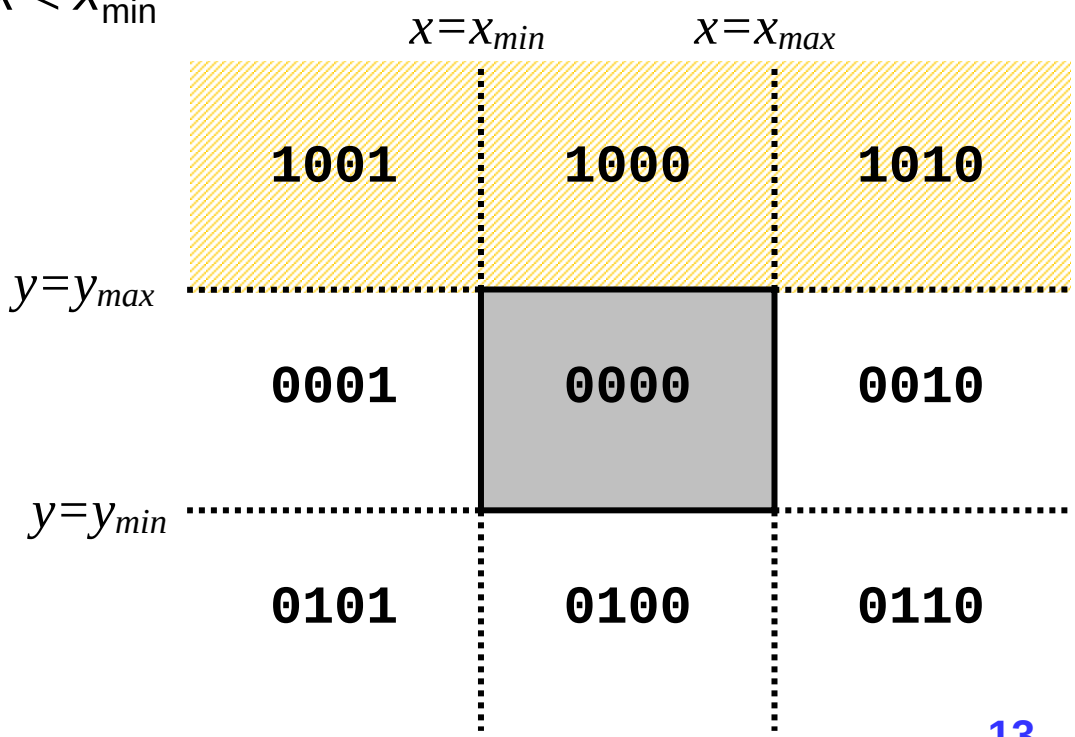
Clipping di un segmento
Algoritmo Cohen-Sutherland

- Idea di base: le rette che delimitano il rettangolo di clipping suddividono il piano in nove regioni;
- Ad ogni regione viene associato un codice numerico di quattro cifre binarie:
- bit 1: sopra edge alto $y > y_{max}$
- bit 2: sotto edge basso $y < y_{min}$
- bit 3: a destra edge destro $x > x_{max}$
- bit 4: a sinistra edge sinistro $x < x_{min}$



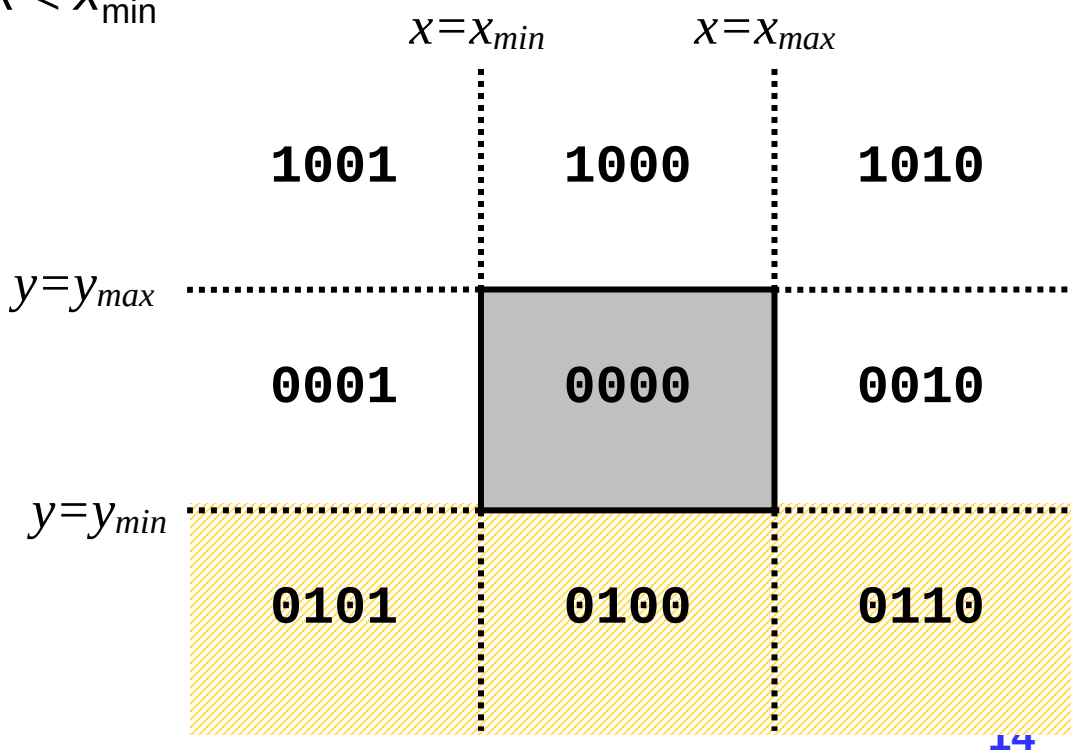
Clipping di un segmento
Algoritmo Cohen-Sutherland

- Ad ogni regione viene associato un codice numerico di quattro cifre binarie:
- bit 1: sopra edge alto $y > y_{max}$
- bit 2: sotto edge basso $y < y_{min}$
- bit 3: a destra edge destro $x > x_{max}$
- bit 4: a sinistra edge sinistro $x < x_{min}$



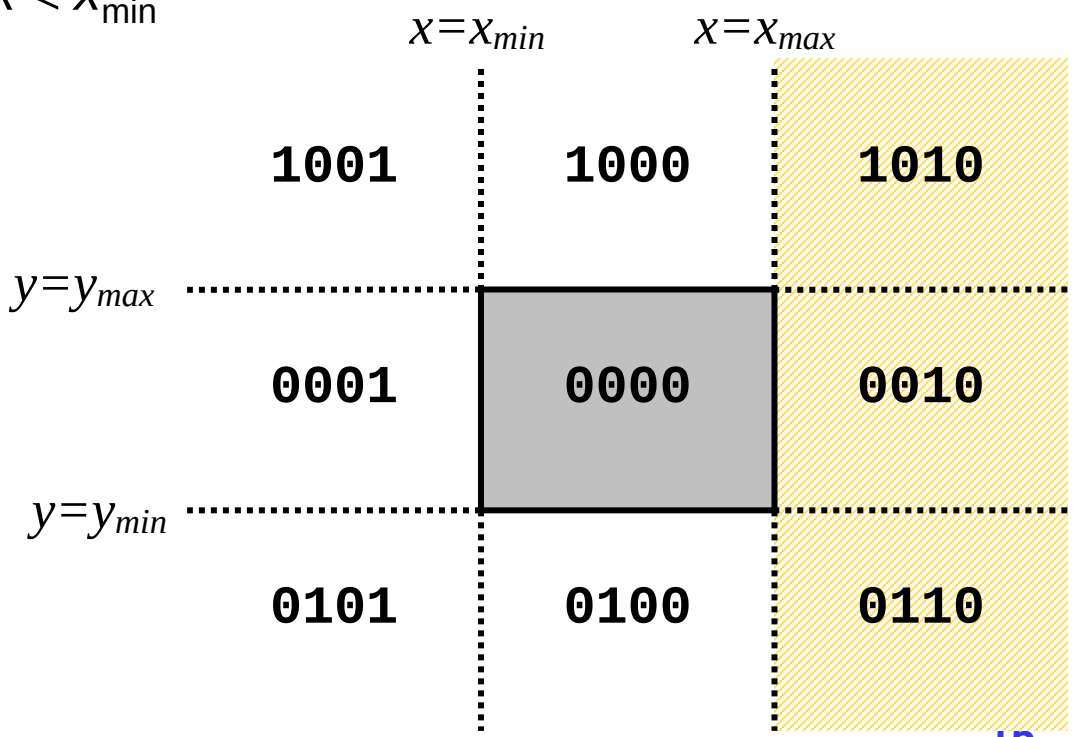
Clipping di un segmento
Algoritmo Cohen-Sutherland

- Ad ogni regione viene associato un codice numerico di quattro cifre binarie:
- bit 1: sopra edge alto $y > y_{max}$
- bit 2: sotto edge basso $y < y_{min}$
- bit 3: a destra edge destro $x > x_{max}$
- bit 4: a sinistra edge sinistro $x < x_{min}$



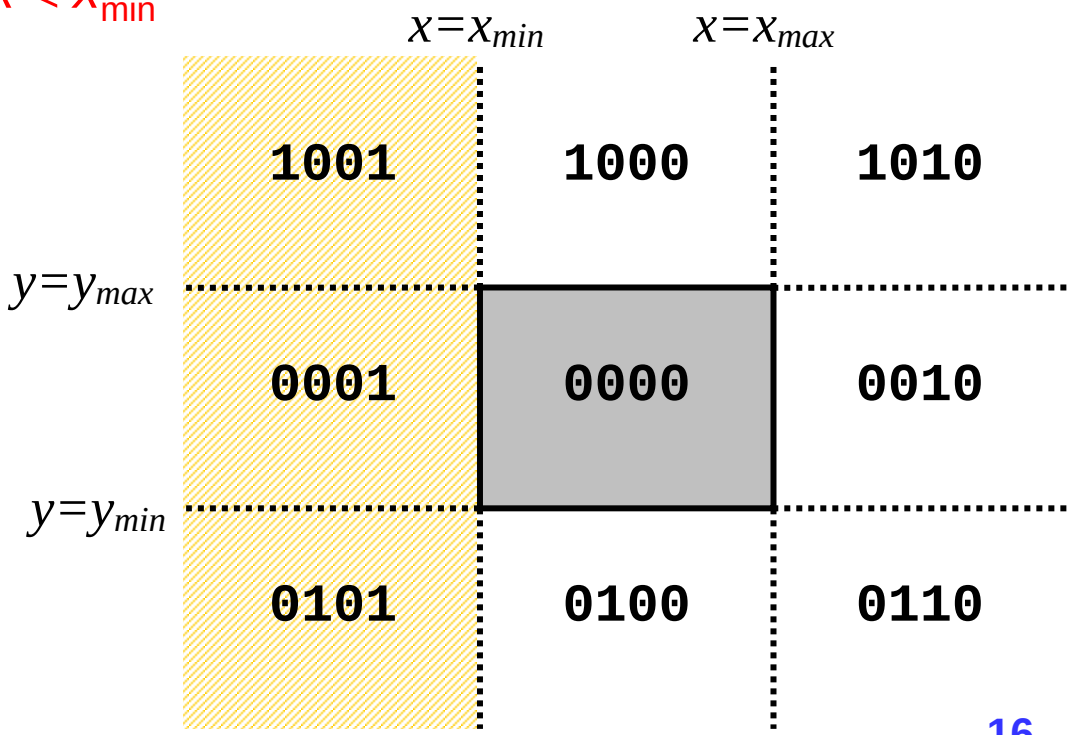
Clipping di un segmento
Algoritmo Cohen-Sutherland

- Ad ogni regione viene associato un codice numerico di quattro cifre binarie:
- bit 1: sopra edge alto $y > y_{max}$
- bit 2: sotto edge basso $y < y_{min}$
- bit 3: a destra edge destro $x > x_{max}$
- bit 4: a sinistra edge sinistro $x < x_{min}$



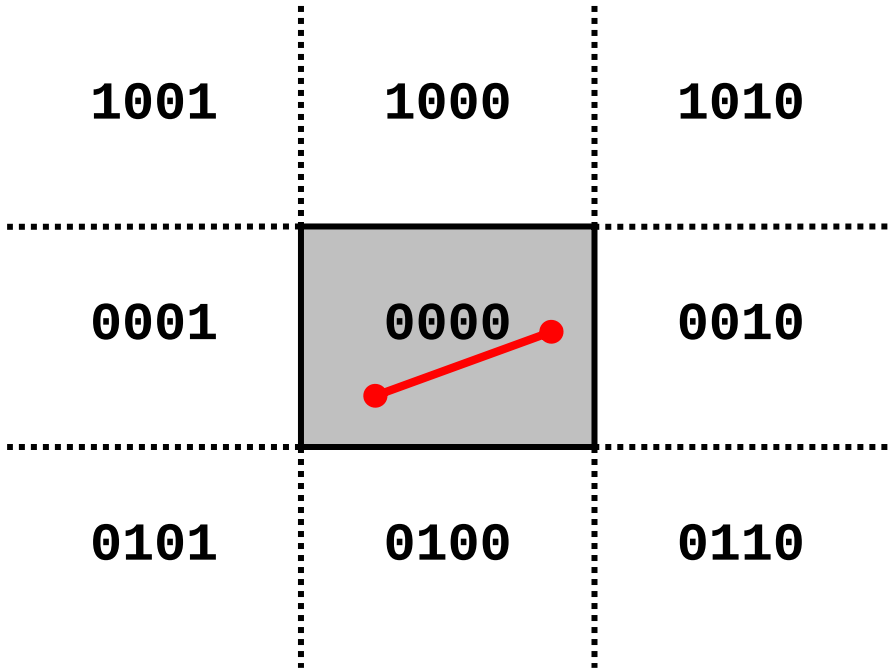
Clipping di un segmento
Algoritmo Cohen-Sutherland

- Ad ogni regione viene associato un codice numerico di quattro cifre binarie:
- bit 1: sopra edge alto $y > y_{max}$
- bit 2: sotto edge basso $y < y_{min}$
- bit 3: a destra edge destro $x > x_{max}$
- bit 4: a sinistra edge sinistro $x < x_{min}$



Clipping di un segmento
Algoritmo Cohen-Sutherland

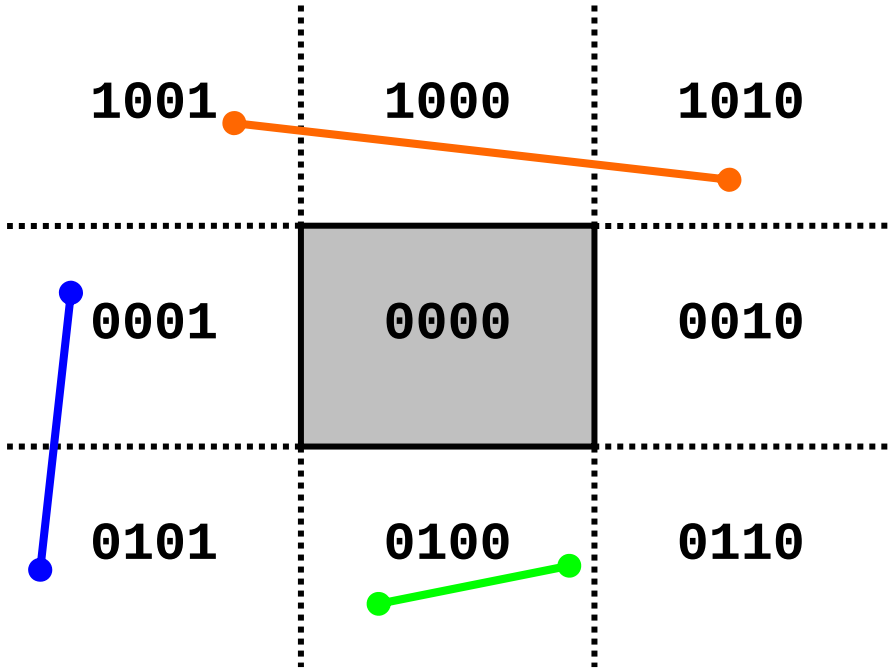
- Il clipping di un segmento prevede la codifica (e confronto) dei suoi estremi sulla base delle regioni di appartenenza;
- Se il codice di entrambi gli estremi è 0000 (OR logico tra i codici ritorna un risultato nullo), allora si può banalmente decidere che il segmento è interamente interno al rettangolo di clipping.



$0000 \vee 0000 = 0000$

Clipping di un segmento
Algoritmo Cohen-Sutherland

- Se l'operazione di AND logico tra i codici degli estremi restituisce un risultato non nullo allora il segmento è esterno al rettangolo di clipping.
- In questo caso, infatti, gli estremi giacciono in uno stesso semipiano (quello identificato dal bit a 1 del risultato) e quindi il segmento non interseca il rettangolo di clipping.

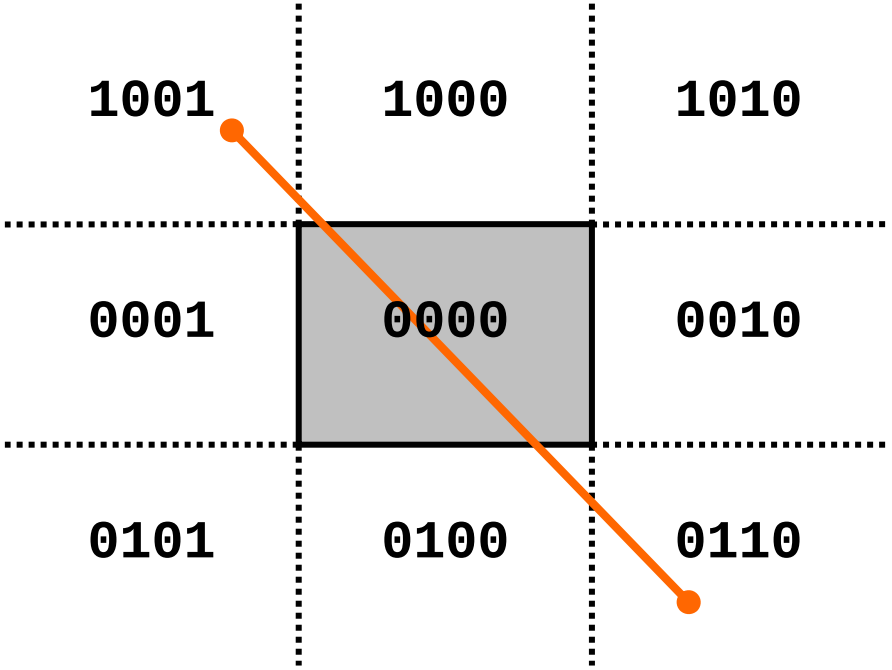


$1001 \wedge 1010 = 1000$

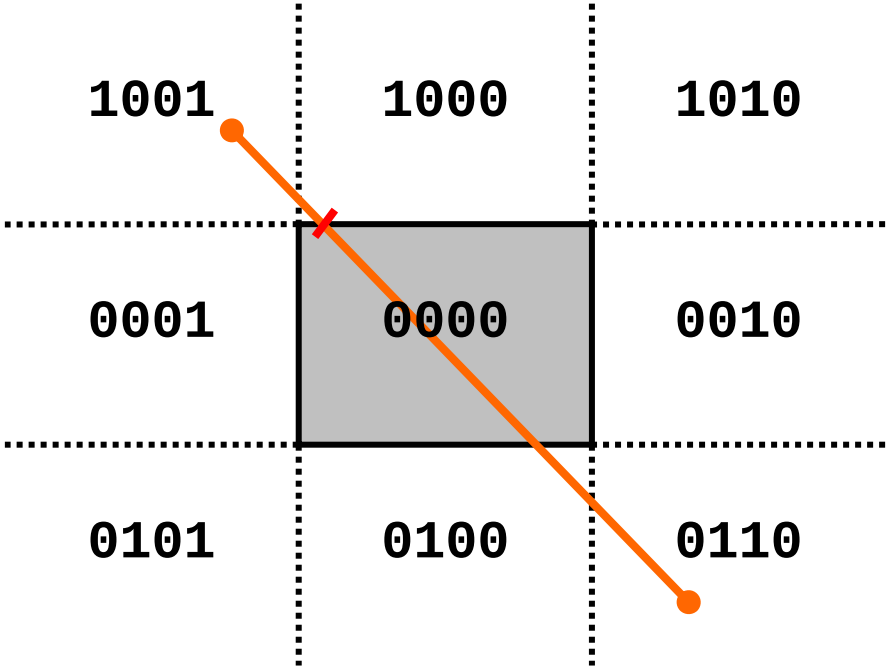
$0100 \wedge 0100 = 0100$

$0001 \wedge 0101 = 0001$

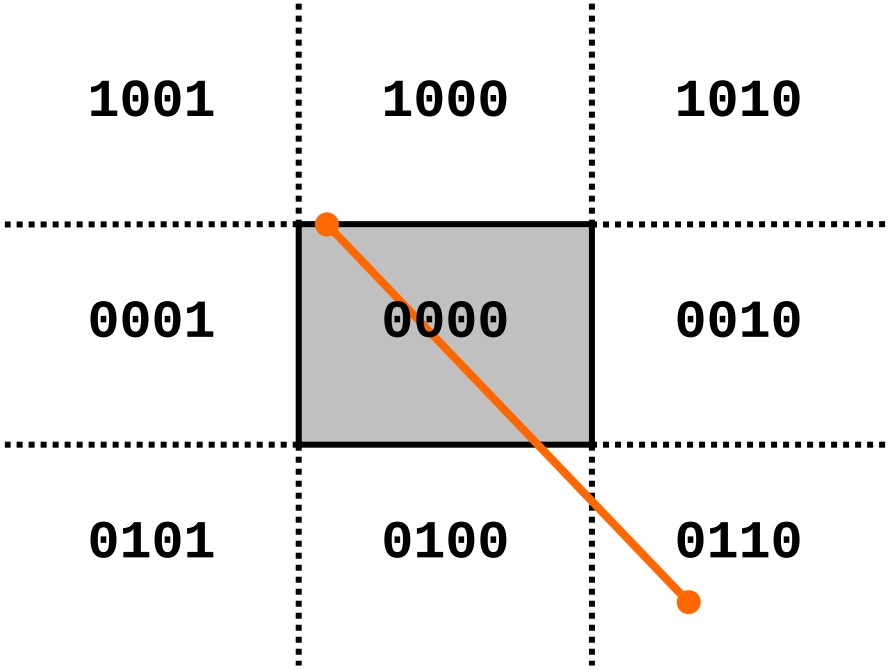
- Se il risultato dell'AND è nullo (ed almeno uno dei codici associati ai vertici è diverso da 0000):



- Se il risultato dell'AND è nullo (ed almeno uno dei codici...):
  - Si individua l'intersezione tra il segmento ed il lato relativo al primo bit *discordante tra i codici* (bit 1, $y=y_{max}$ in fig.);

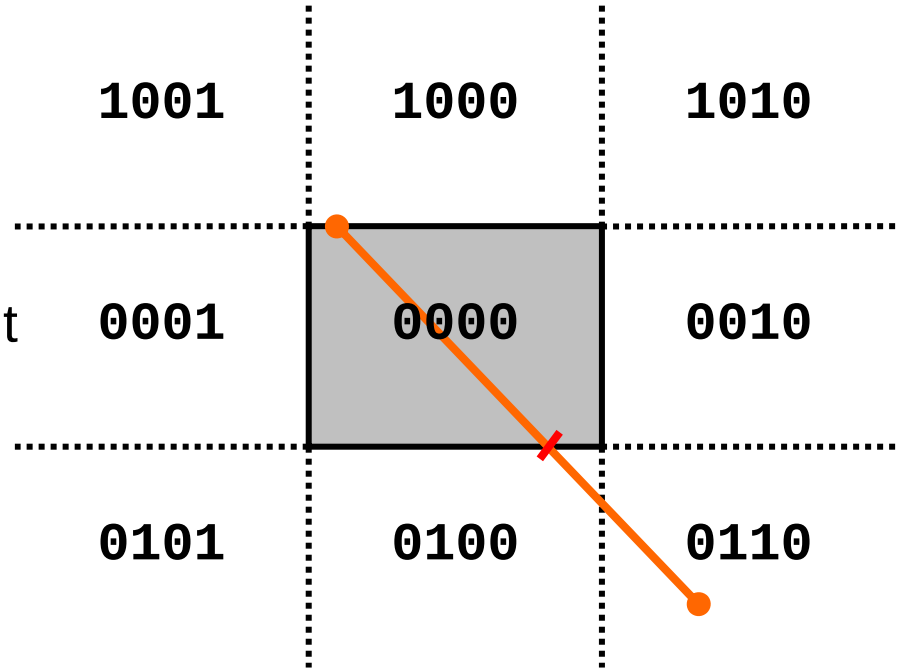


- Se il risultato dell'AND è nullo (ed almeno uno dei codici...):
  - Si individua l'intersezione tra il segmento ed il lato relativo al primo bit *discordante tra i codici* (bit 1, $y=y_{max}$ in fig.);
  - L'estremo con bit a 1 (in prima posizione nell'esempio) viene sostituito dal nuovo vertice;



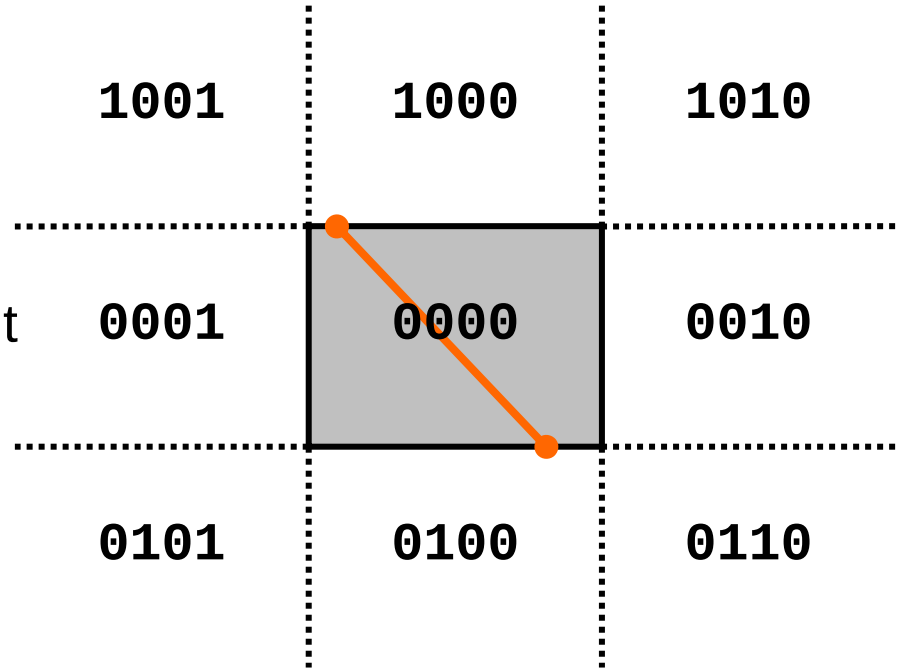
Clipping di un segmento
Algoritmo Cohen-Sutherland

- Se il risultato dell'AND è nullo (ed almeno uno dei codici...):
  - Si individua l'intersezione tra il segmento ed il lato relativo al primo bit discordante tra i codici (bit 1, $y=y_{max}$ in fig.);
  - L'estremo con bit a 1 (in prima posizione nell'esempio) viene sostituito dal nuovo vertice;
  - Si itera il procedimento (in fig., bit 2 discordante, intersezione del segmento con $y=y_{min}$);



Clipping di un segmento
Algoritmo Cohen-Sutherland

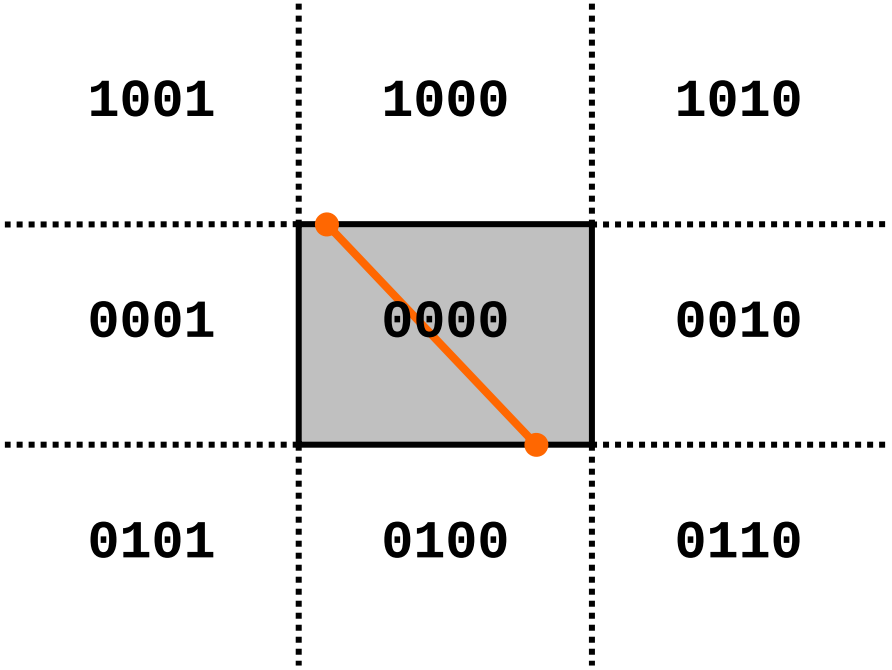
- Se il risultato dell'AND è nullo (ed almeno uno dei codici...):
  - Si individua l'intersezione tra il segmento ed il lato relativo al primo bit discordante tra i codici (bit 1, $y=y_{max}$ in fig.);
  - L'estremo con bit a 1 (in prima posizione nell'esempio) viene sostituito dal nuovo vertice;
  - Si itera il procedimento (in fig., bit 2 discordante, intersezione del segmento con $y=y_{min}$);
  - L'estremo con bit a 1 (il bit 2 in fig.) viene sostituito dal nuovo estremo.



Clipping di un segmento
Algoritmo Cohen-Sutherland

- Ad ogni iterazione si controlla l'eventuale terminazione del processo (OR logico nullo);

- L'algoritmo rimuove progressivamente le parti esterne; risulta efficiente quando molti dei segmenti da clippare sono completamente esterni al rettangolo di clipping.



Clipping di un segmento
Algoritmo Liang-Barsky

- Il generico segmento AB di estremi (x_a,y_a) e (x_b,y_b) può essere espresso in forma parametrica come:

$$x = x_a + (x_b - x_a)t = x_a + \Delta x t,$$
$$y = y_a + (y_b - y_a)t = y_a + \Delta y t$$

- Per ogni punto (x,y) del segmento interno al rettangolo di clipping valgono le relazioni:

$$x_{min} \leq x_a + \Delta x t \leq x_{max},$$
$$y_{min} \leq y_a + \Delta y t \leq y_{max}.$$

- Che possono essere riscritte nelle 4 disuguaglianze:

$$p_k t \leq q_k \quad k = 1, 2, 3, 4$$

- Che possono essere riscritte nelle 4 disuguaglianze:

$$p_k t \leq q_k \quad k = 1, 2, 3, 4$$

- Dove p e q valgono le quantità:

$$\begin{aligned} p_1 &= -\Delta x, \quad & q_1 &= x_a - x_{min}, \\ p_2 &= \Delta x, \quad & q_2 &= x_{max} - x_a, \\ p_3 &= -\Delta y, \quad & q_3 &= y_a - y_{min}, \\ p_4 &= \Delta y, \quad & q_4 &= y_{max} - y_a, \end{aligned}$$

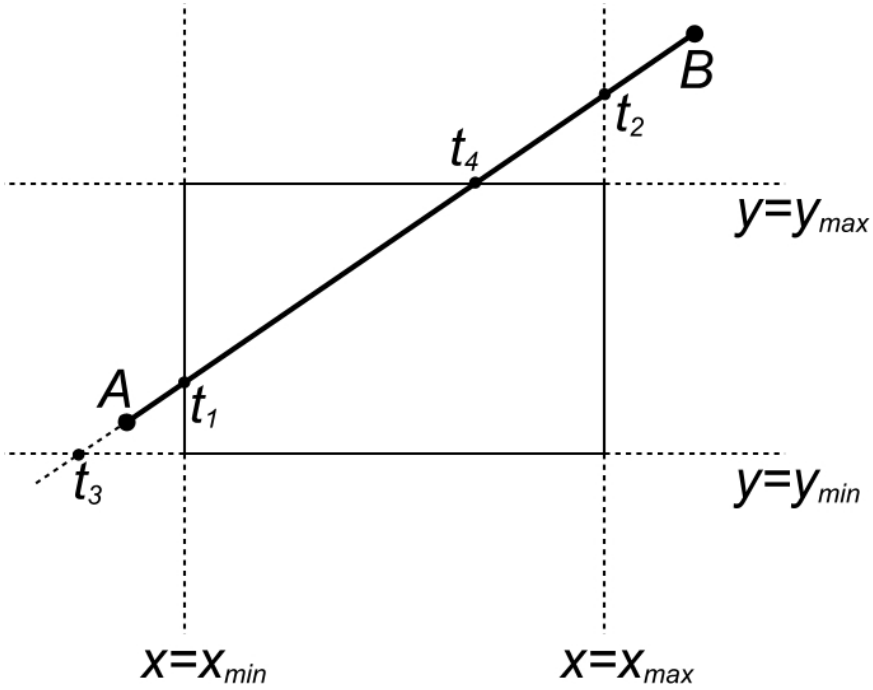
- Relative alle relazioni per il vincolo $x=x_{min}$ (1), $x=x_{max}$ (2), $y=y_{min}$ (3) e $y=y_{max}$ (4).

Clipping di un segmento
Algoritmo Liang-Barsky

- A(14,19), B(33,32)
- x_{min}=16,x_{max}=30,y_{min}=18,y_{max}=27

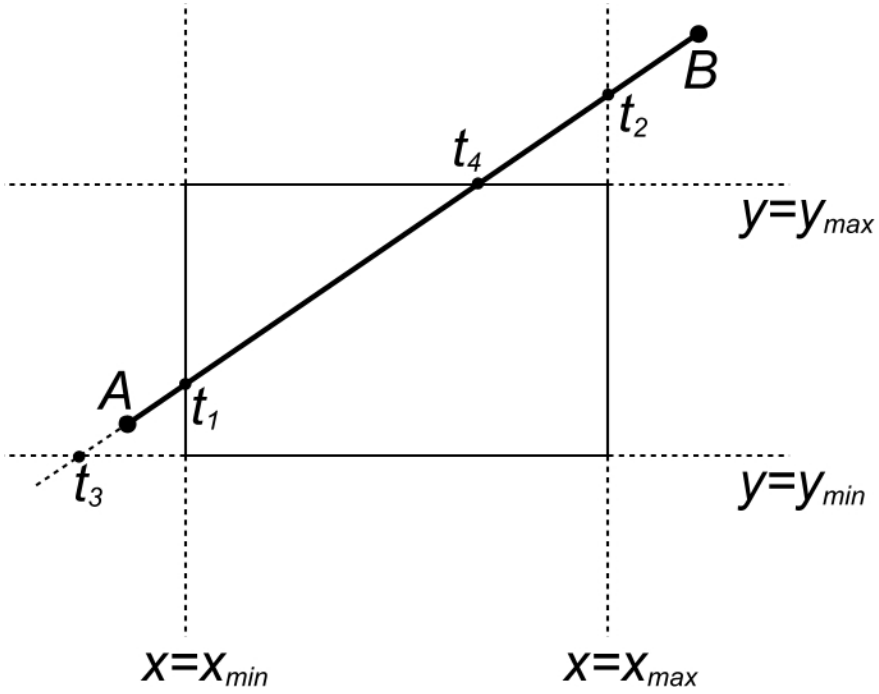
$$p_1 = -19, \quad q_1 = -2,$$
$$p_2 = 19, \quad q_2 = 16,$$
$$p_3 = -13, \quad q_3 = 1,$$
$$p_4 = 13, \quad q_4 = 8.$$

- Se p_k<0 allora nel muoversi nel verso da A a B si passa da fuori a dentro rispetto al vincolo k;
- Se p_k>0 allora nel muoversi nel verso da A a B si passa da dentro a fuori rispetto al vincolo k;
- t_1=0.105, t_2=0.842, t_3=-0.077, t_4=0.615



Clipping di un segmento
Algoritmo Liang-Barsky

- t_1=0.105, t_2=0.842, t_3=-0.077, t_4=0.615
- La parte di AB interna al rettangolo di clipping è individuata da t_e (entrata) e t_u (uscita) dove:
- t_e è il massimo tra 0 (valore minimo di t) ed i valori t_k per cui si entra nella regione di clipping (p_k<0)
- t_e = max(0,t_1,t_3) = 0.105
- t_u è il minimo tra 1 (valore massimo di t) ed i valori t_k per cui si esce dalla regione di clipping (p_k>0)
- t_u = min(1,t_2,t_4) = 0.615



Clipping di un segmento
Algoritmo Liang-Barsky

$$A(13,25), \quad B(31,43)$$

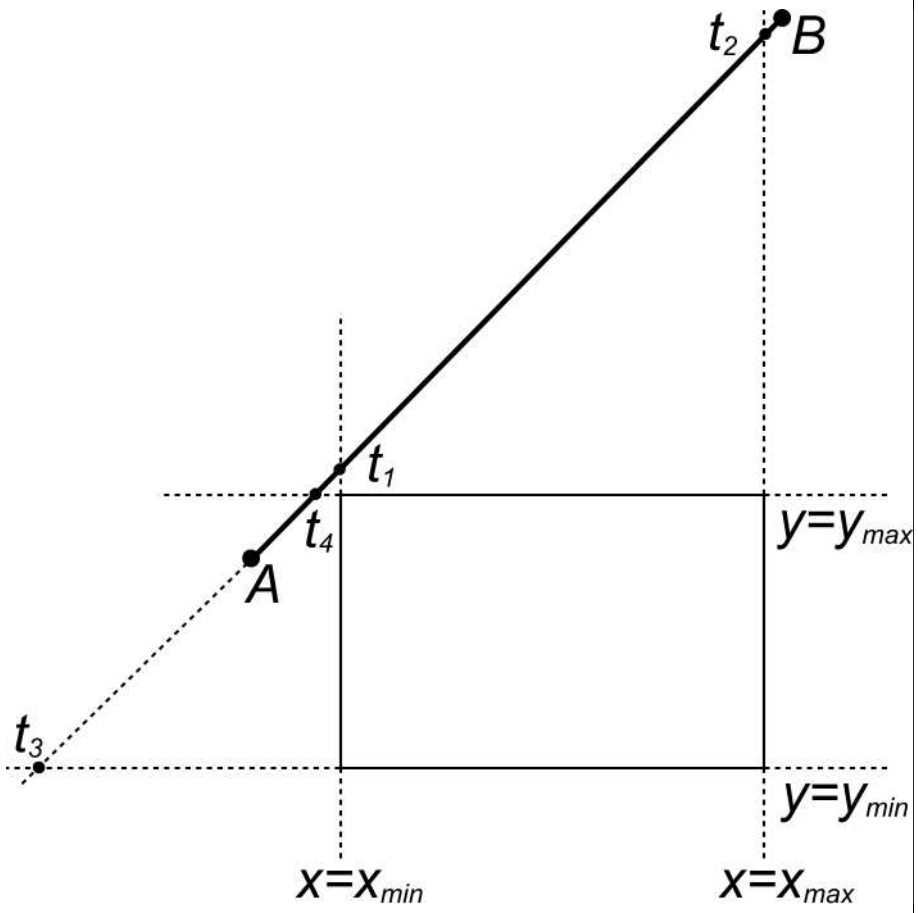
$$x_{min} = 16, \quad x_{max} = 30,$$

$$y_{min} = 18, \quad y_{max} = 27,$$

$$p_1 = -18, \quad q_1 = -3,$$
$$p_2 = 18, \quad q_2 = 17,$$
$$p_3 = -18, \quad q_3 = 72,$$
$$p_4 = 18, \quad q_4 = 2,$$

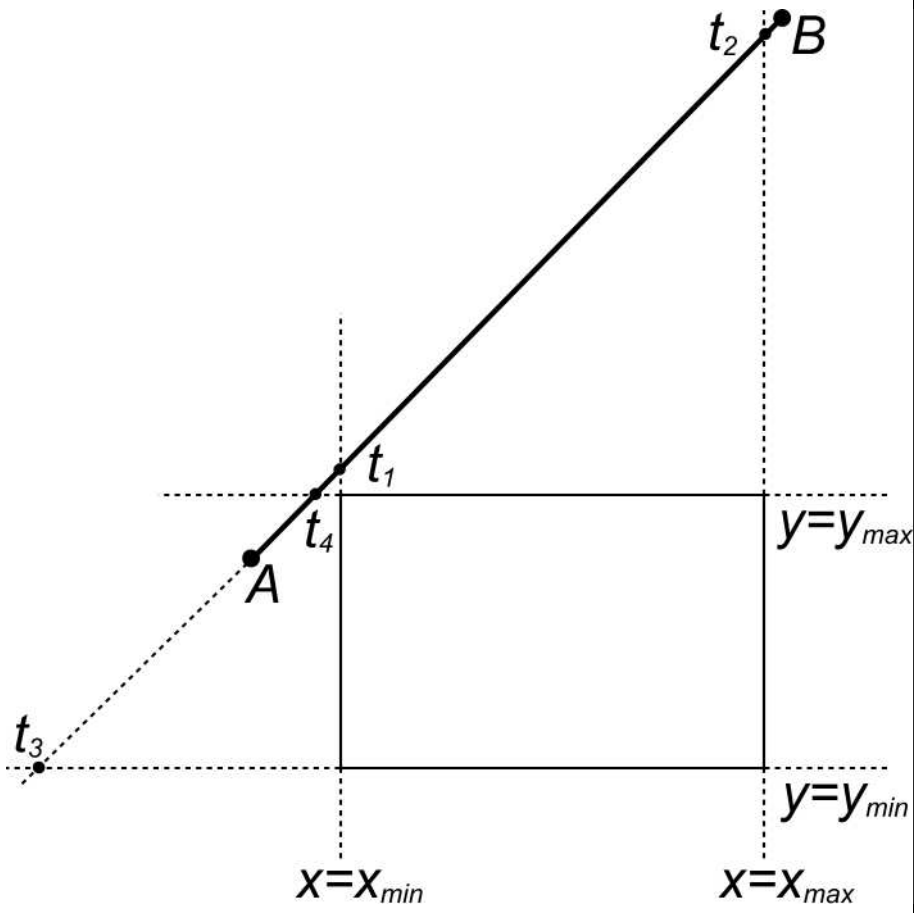
$$t_1 =\sim 0.167, \quad t_2 =\sim 0.944,$$
$$t_3 =\sim (-0.389), \quad t_4 =\sim 0.111$$
$$t_e = \max(0, t_1, t_3)$$
$$t_u = \min(1, t_2, t_4)$$

$$(t_e > t_u)$$



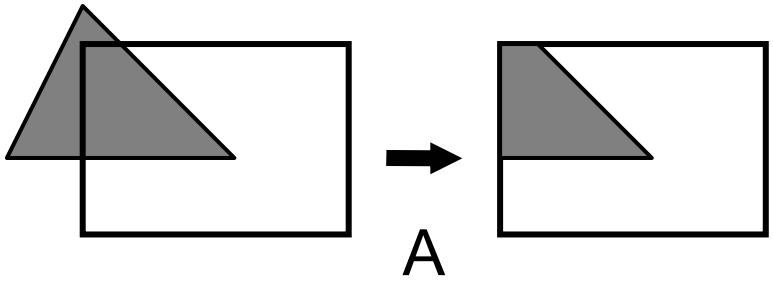
Clipping di un segmento
Algoritmo Liang-Barsky

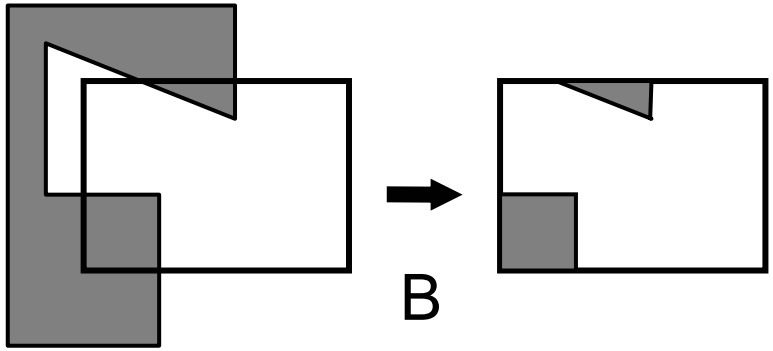
- Se $t_e > t_u$ allora il segmento è esterno al rettangolo di clipping.
- L'algoritmo di Liang-Barsky evita la determinazione di intersezioni (inutili);
- Gli algoritmi di Cohen-Sutherland e Liang-Barsky sono estendibili al clipping 3D.



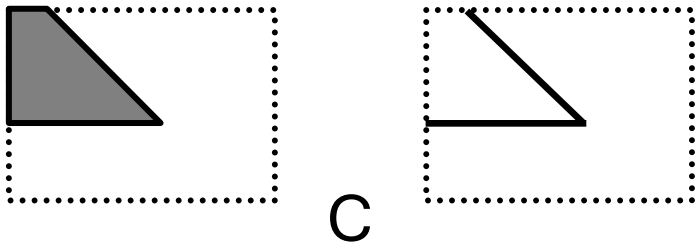
Clipping di un poligono

- Il clipping di un poligono è un'operazione più complessa rispetto al clipping di un segmento per diversi aspetti:
- Dal semplice poligono convesso (A);
- Al poligono concavo che origina più componenti connesse (B);



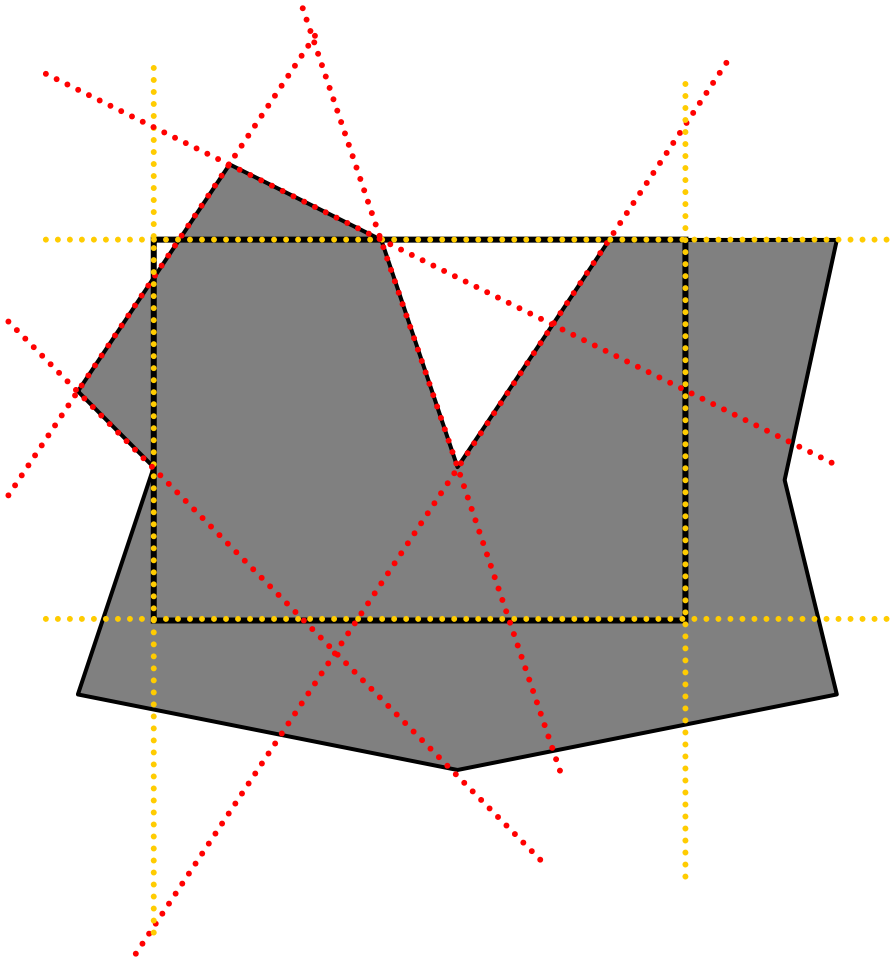


- In ogni caso il risultato consta di uno o più poligoni e non solo segmenti sconnessi (C).



Clipping di un poligono

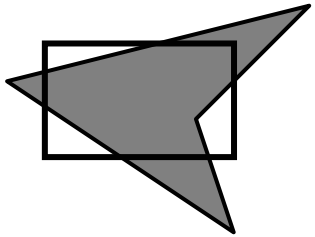
- L’approccio diretto consiste nel confrontare ogni lato del poligono con le 4 rette che delimitano il rettangolo di clipping;
- Questo approccio implica l’esecuzione di operazioni costose (la determinazione di intersezioni) e spesso inutili.



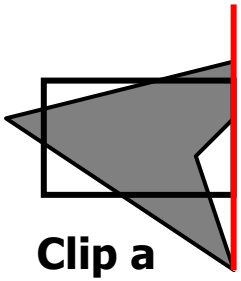
Clipping di un poligono
Algoritmo Sutherland-Hogdman

- Approccio *divide et impera;*

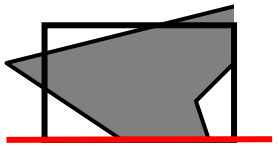
- Il problema è ricondotto al clipping di un poligono generico rispetto ad una retta;



Poligono originale

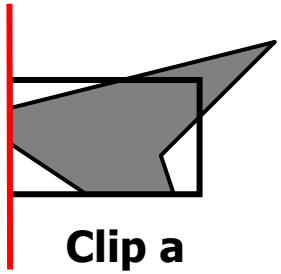


Clip a destra

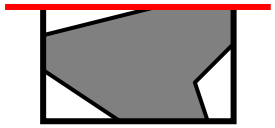


Clip in basso

- La procedura è applicata sequenzialmente alle 4 rette che definiscono il rettangolo di clipping.



Clip a sinistra



Clip in alto

Rimozione Superfici Nascoste

- Gli oggetti della scena sono generalmente opachi;
- Gli oggetti più vicini all'osservatore possono nascondere (occludere) la vista (totale o parziale) di oggetti più lontani;
- Il problema della rimozione delle superfici nascoste (Hidden surface Removal) consiste nel determinare le parti della scena non visibili dall'osservatore;
- La rimozione delle superfici nascoste non è solo dipendente dalla disposizione degli oggetti nella scena ma anche dalla relazione esistente tra oggetti e posizione dell'osservatore.

Object Space ed Image Space

- Gli algoritmi per la rimozione delle superfici nascoste si possono classificare object-space ed image-space:
  - gli algoritmi che operano in ***object-space*** determinano, **per ogni primitiva geometrica** della scena, le parti della primitiva che non risultano oscurate da altre primitive nella scena. Gli algoritmi operano nello spazio di definizione delle primitive;
  - gli algoritmi che operano in ***image-space*** determinano, per ogni punto “significativo” del piano di proiezione **(ogni pixel del piano immagine)**, la primitiva geometrica visibile “attraverso” quel punto. Gli algoritmi operano nello spazio immagine della scena proiettata.

HSR: Object-Space

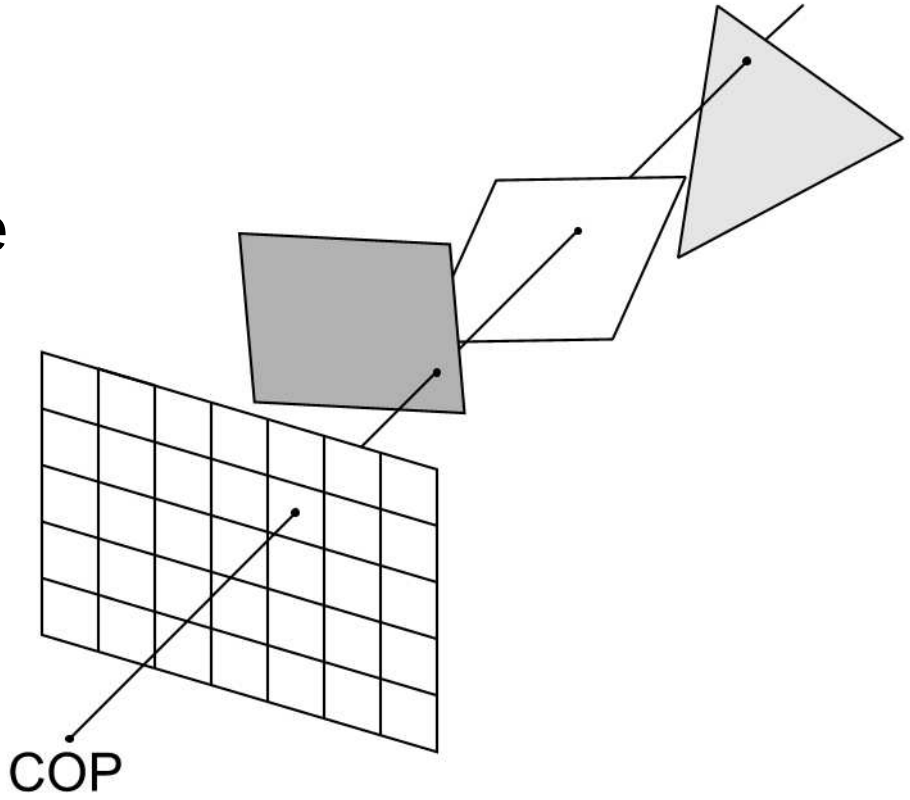
- Nell'ipotesi di una scena 3D composta da *k* primitive geometriche planari ed opache, si può derivare un generico algoritmo di tipo object-space analizzando gli oggetti a coppie;
- Fissato un punto di vista, le relazioni spaziali di due primitive geometriche A e B possono essere:
  1. A oscura B; solo A deve essere visualizzata;
  2. A e B sono completamente visibili; entrambe le primitive sono visualizzate;
  3. A occlude parzialmente B: *è necessario individuare le parti visibili di B*.

HSR: Object-Space

- Algoritmo:
  - Proiettare le *k* primitive geometriche;
  - Al generico passo analizzare la *i-esima* primitiva (i=1, …, k-1) con le rimanenti *k – i* in modo da individuare le parti visibili.
- La complessità dell'approccio object-space risulta quindi di ordine **$O(k^2)$**

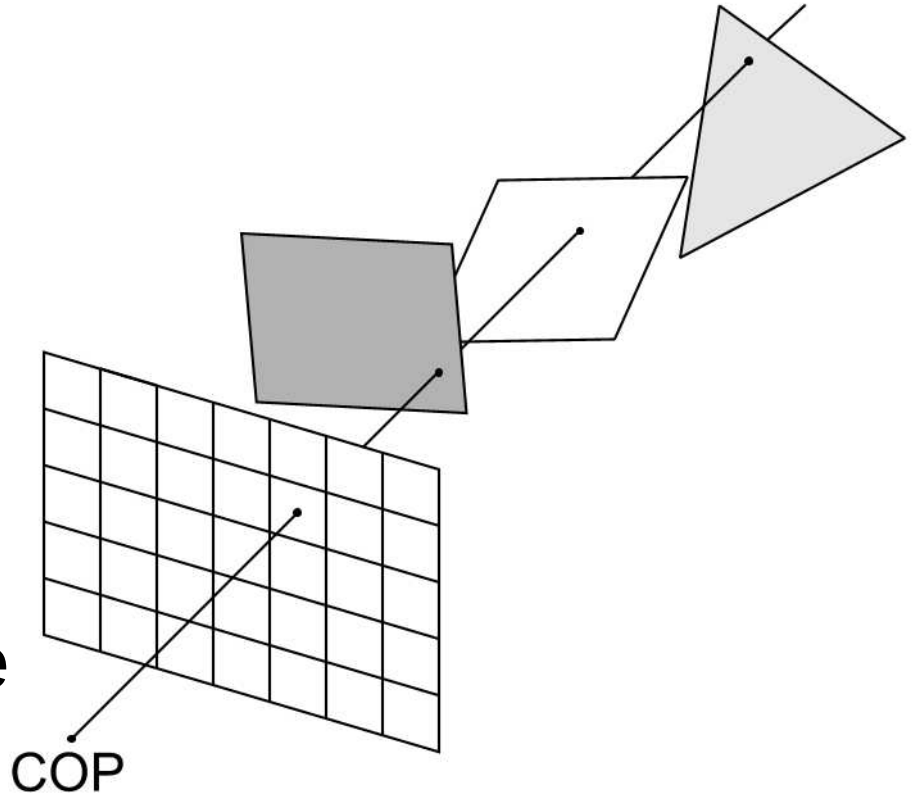
HSR: Image-Space

- Per ogni pixel del piano immagine si considera una semiretta avente origine nel centro di proiezione e passante per il pixel in esame. La semiretta attraversa la scena fino a colpire una primitiva o a perdersi sul fondo.



HSR: Image-Space

- Per ogni primitiva si calcola l'intersezione della semiretta con il piano di appartenenza e si memorizzano le intersezioni
- Tra le intersezioni accumulate si sceglie quella con distanza minore dal centro di proiezione e si attribuisce al pixel in esame il colore della primitiva intersecata.



HSR: Image-Space

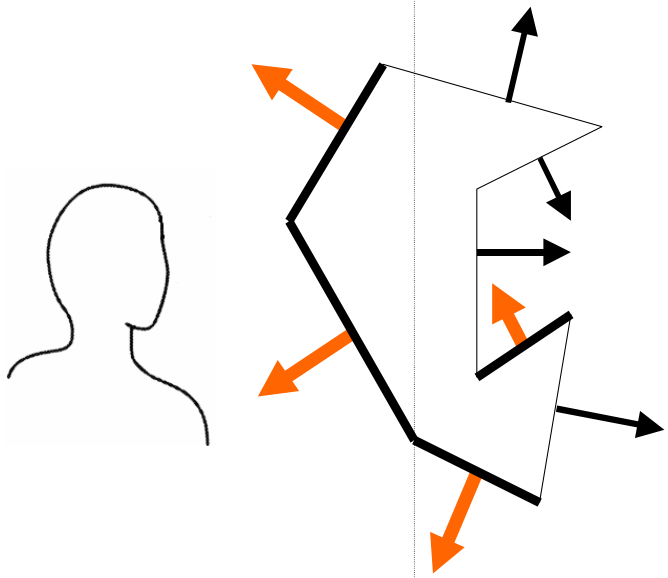
- L'operazione fondamentale dell'approccio ***image-space*** è il calcolo delle intersezioni tra semiretta e primitive (per ogni semiretta massimo si hanno *k* intersezioni);
- Per un display *n x m*, questa operazione deve essere eseguita *n x m x k* volte, la complessità risulta comunque di ordine ***O(k).***

- Sia nell'approccio object-space che in quello image-space la rimozione delle superfici nascoste è riconducibile ad un problema di **ordinamento (in profondità)**.

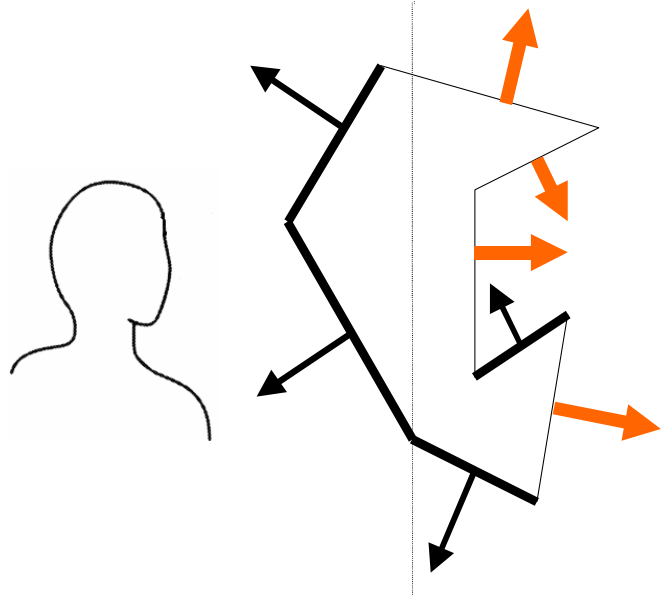
- Back-face culling
  - Significa ELIMINAZIONE DELLE FACCE POSTERIORI
  - Se gli oggetti della scena sono rappresentati da solidi ***chiusi***;
  - Se ogni *faccia* è stata modellata in modo tale che la normale ad essa sia diretta verso l'esterno dell'oggetto;
  - Allora...

Aumentare l'efficienza nell'HSR: Back-face Culling

- Le facce la cui normale forma angoli superiori a ±90° con la direzione di vista **certamente non sono visibili**;

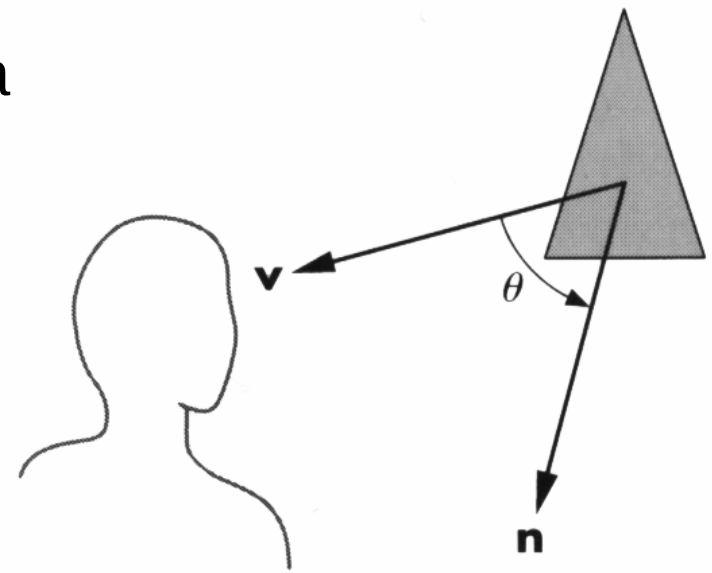


- Le facce la cui normale forma angoli inferiori a ±90° con la direzione di vista **possono essere visibili**;



Aumentare l'efficienza nell'HSR: Back-face Culling

- Per ridurre il carico di lavoro richiesto per la rimozione delle superfici nascoste può essere quindi opportuno eliminare inizialmente tutti le primitive geometriche la cui normale è orientata verso il semispazio opposto all'osservatore, non visibile all'osservatore;
- Indicato con θ l'angolo tra la normale e l'osservatore, la primitiva in esame è visibile se $-90° \leq \theta \leq 90°$, cioè se $\cos\theta \geq 0$.
- Invece di calcolare la quantità $\cos\theta$ possiamo valutare il prodotto scalare $n \cdot v \geq 0$



Aumentare l'efficienza nell'HSR: Back-face Culling

- Se l'operazione è eseguita in coordinate normalizzate di vista (dopo aver eseguito la proiezione) la determinazione delle facce *back-facing* si riduce ad un controllo del segno della coordinata *z* delle normali: ad un segno positivo corrispondono facce *front-facing*, ad un segno negativo facce *back-facing;*
- Questo procedimento (detto *back-face culling*) consente, in media, di dimezzare il tempo necessario per il rendering di oggetti solidi dato che, tipicamente, circa metà delle facce di un oggetto solido sono *back-facing*.

Algoritmo Z-Buffer

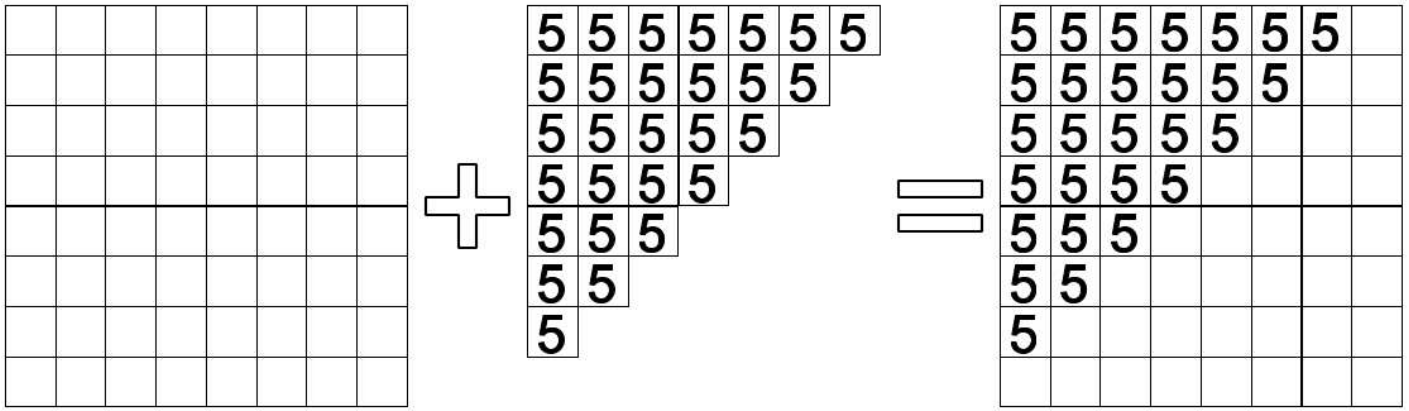
- ***L'algoritmo z-buffer*** è un algoritmo di tipo *image-space*, basato su una logica molto semplice e facile da realizzare;
- Lavora durante la fase di rasterizzazione (*è implementato in hardware*) ed ha bisogno, come struttura dati di supporto, di un'area di memoria, detta *depth buffer*, delle stesse dimensioni del *frame buffer* .
- Per ogni posizione (x,y) della vista che si genera, il *frame buffer* contiene il colore assegnato a quel pixel, il *depth buffer* la profondità del punto corrispondente sulla primitiva visibile.

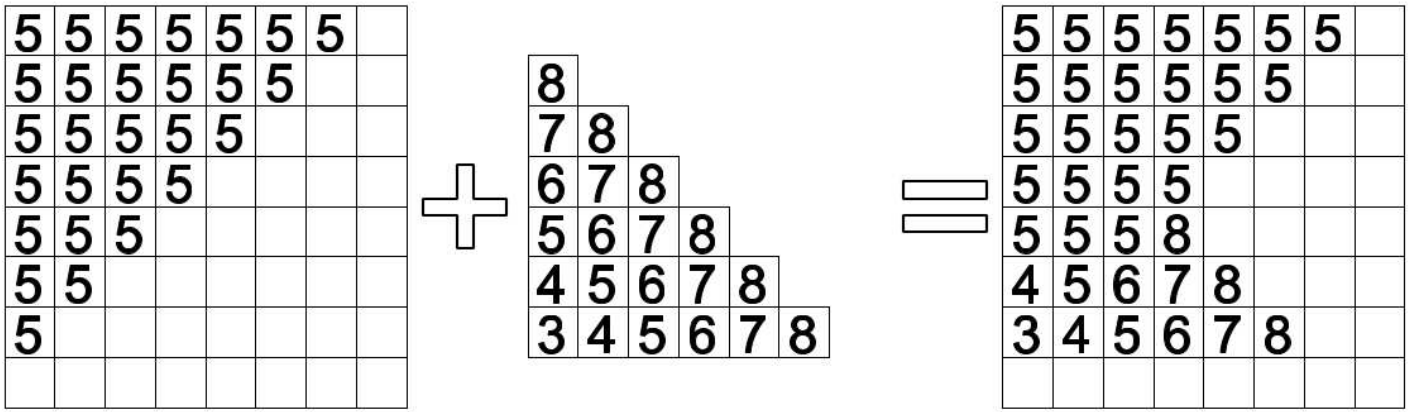
Algoritmo Z-Buffer

- Durante la fase di rasterizzazione delle primitive si determina, per ogni pixel (x,y) su cui la primitiva viene mappata, la profondità della primitiva in quel punto.
- Se la profondità *z* in (x,y) è inferiore alla profondità corrente memorizzata nello z-buffer allora lo Z-buffer assume *z* come profondità corrente in (x,y) ed il pixel (x,y) nel frame buffer assume il valore colore della primitiva in esame.

Algoritmo Z-Buffer

- Esempio



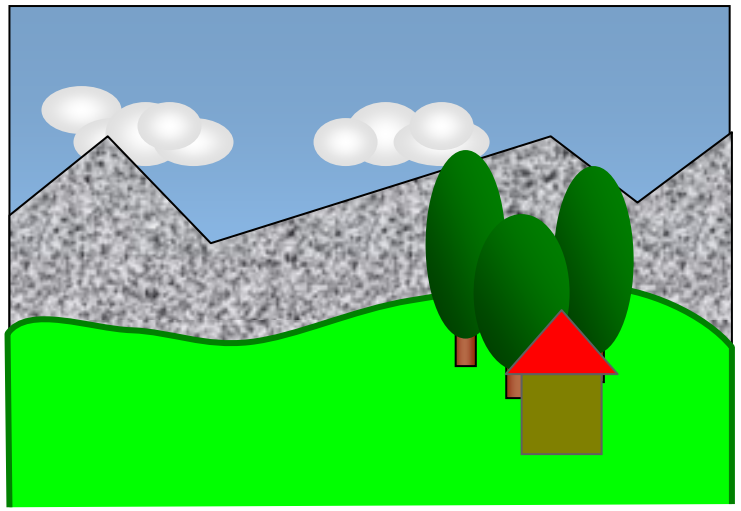


Algoritmo Z-Buffer

- Lo z-buffer ha la stessa risoluzione del frame buffer;
- Ogni elemento dello z-buffer è inizializzato al valore della distanza massima dal centro di proiezione;
- Non è richiesto alcun ordine preventivo tra le primitive geometriche;
- Implementato sull'hardware grafico;
- Complessità pressoché costante (ad un aumento delle primitive corrisponde in genere una diminuzione della superficie coperta).

Algoritmo del Pittore (depth-sort)

- Gli oggetti della scena 3D siano rappresentati mediante primitive geometriche (poligoni) planari;
- I poligoni planari siano ordinati sulla base della loro distanza dall'osservatore;
- L'idea di base è quella di seguire un approccio analogo a quello usato da un pittore: dipingere prima il poligono più lontano dall'osservatore e quindi dipingere via via i poligoni rimanenti seguendo l'ordine definito in precedenza.
- Gli elementi più lontani sono progressivamente oscurati da quelli più vicini all'osservatore.

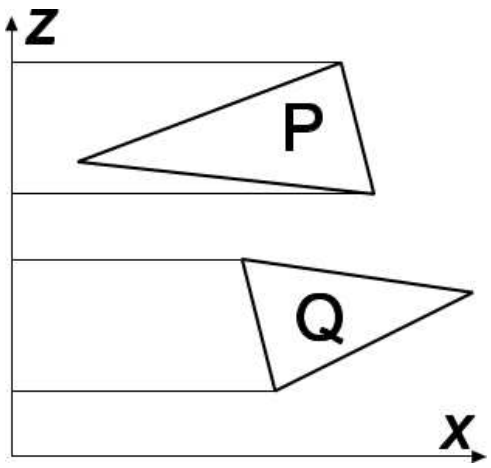


Algoritmo del Pittore

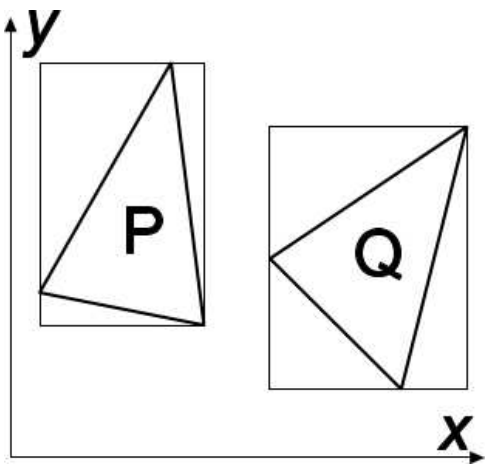
- L'algoritmo:
  - Individuare un ordinamento in profondità (lungo la direzione di vista) delle primitive della scena (*depth sort,* ordinamento in profondità). Ordinamento effettuato in object-space.
  - Visualizzare le primitive della scena in modalità *back-to-front.* Rasterizzazione delle primitive effettuata in image-space, nello spazio di coordinate del dispositivo;
- Occorre una strategia che permetta di risolvere i problemi legati alle eventuali sovrapposizioni in profondità delle primitive geometriche.

Algoritmo del Pittore

- Il primo passo consiste nell'ordinamento delle primitive lungo la direzione di vista (z) in accordo al vertice con profondità massima;
- Se gli ***extent*** della primitiva a profondità maggiore (P) non si sovrappongono agli extent della primitiva che segue (Q), allora P precede Q;

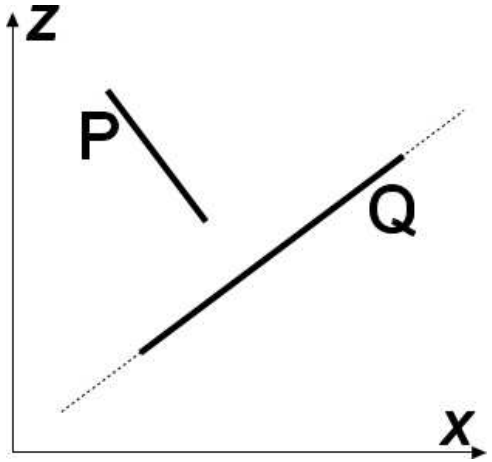


- Altrimenti, se i ***bounding box*** di P e Q sul piano xy non interferiscono, allora P precede Q;

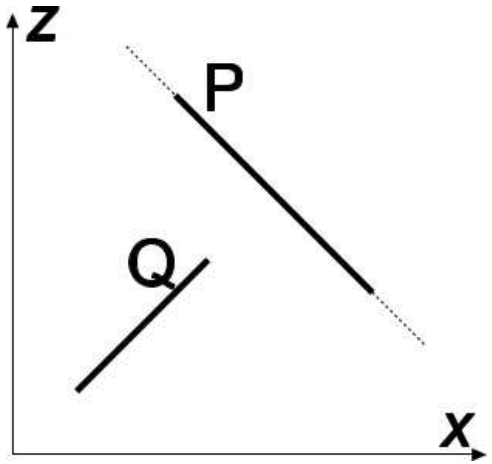


Algoritmo del Pittore

- Altrimenti, se tutti i vertici di P si trovano dalla parte opposta dell'osservatore rispetto al piano individuato da Q, allora P precede Q;

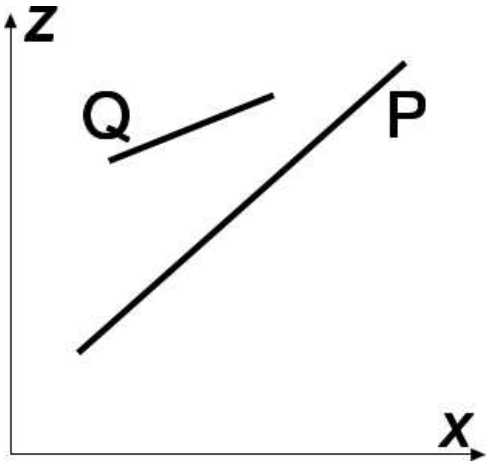


- Altrimenti, se tutti i vertici di Q si trovano dalla stessa parte dell'osservatore rispetto al piano individuato da P, allora P precede Q;
- Altrimenti, se le proiezioni di P e Q sul piano immagine non interferiscono, allora P precede Q.

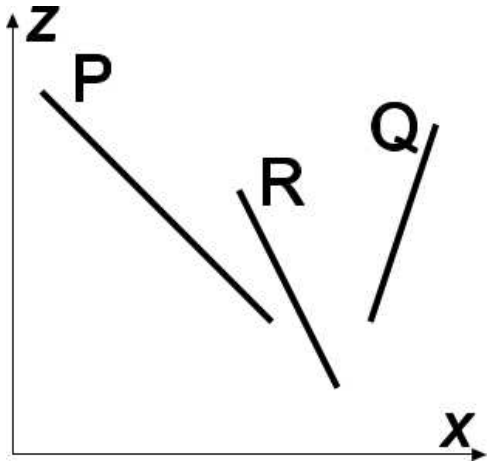


Algoritmo del Pittore

- Se tutti i test precedenti forniscono esito negativo allora si procede allo scambio di P con Q nell'ordinamento e, nuovamente, all'esecuzione dei test;

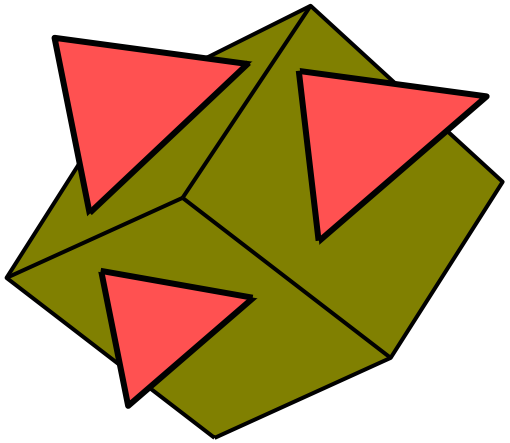


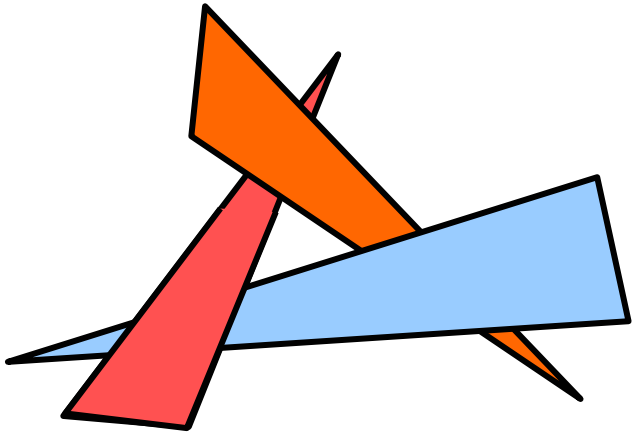
- Esempio (vedi figura): l'ordinamento iniziale P, Q, R è sovvertito dallo scambio di P con R e quindi dallo scambio di R con Q.



Algoritmo del Pittore

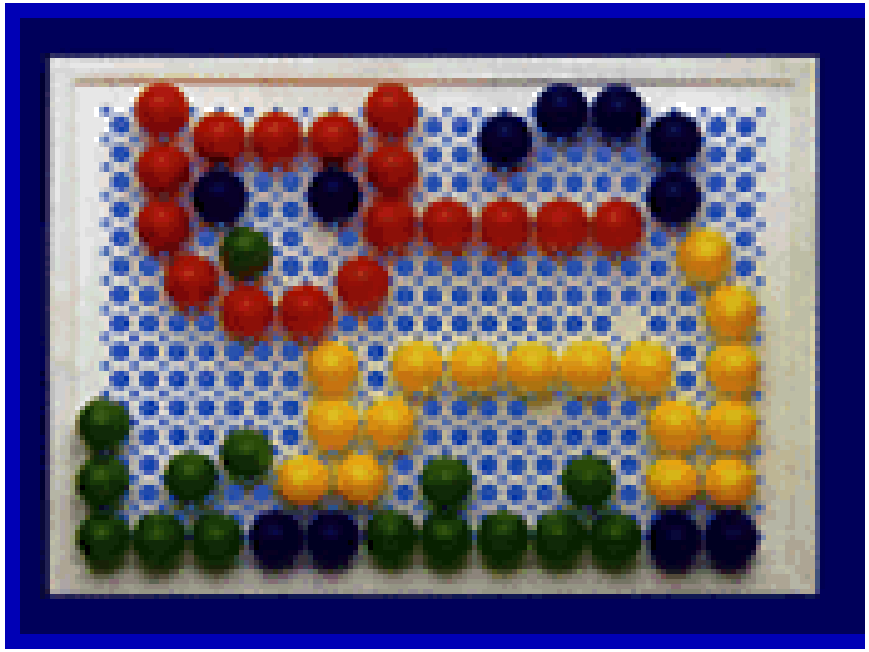
- La presenza di oggetti nella scena che si oscurano in modo ciclico può dar luogo a non terminazione dell'algoritmo;
- In questi casi è necessario procedere alla suddivisione delle primitive geometriche che danno luogo a cicli.
- La tecnica depth buffer presenta prestazioni efficienti soprattutto per scene poco complesse (con bassa probabilità di interferenza in z delle primitive).





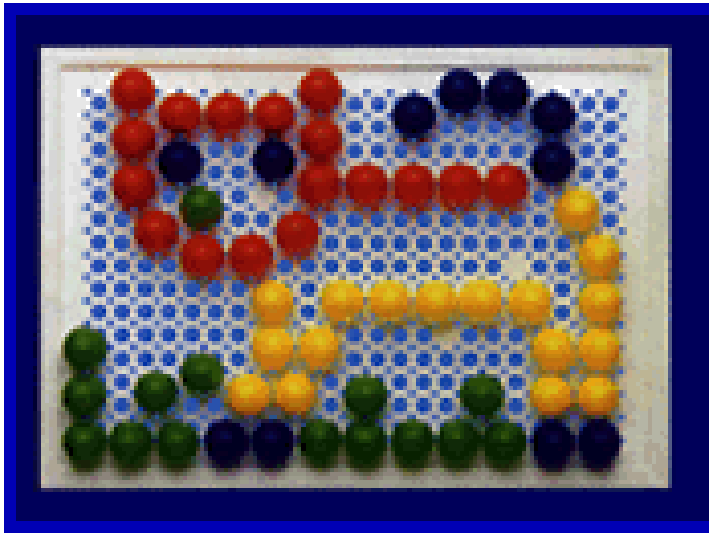
Rasterizzazione

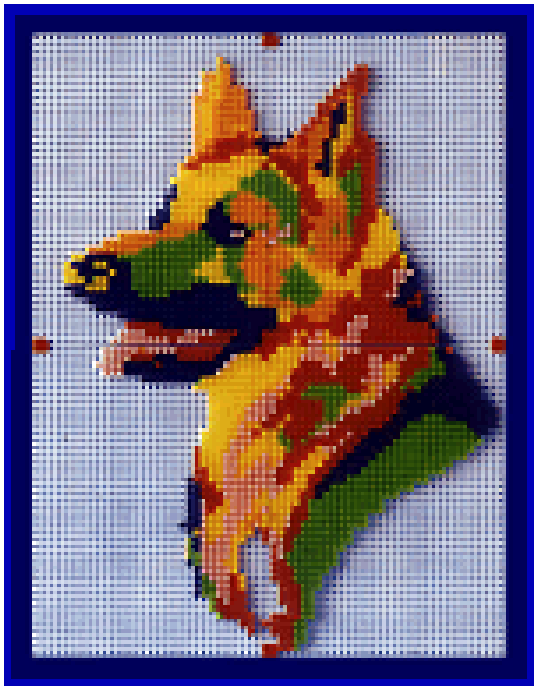
- Con il termine *rasterizzazione* si intende il processo di discretizzazione che consente di trasformare una primitiva geometrica definita in uno spazio continuo 2D nella sua rappresentazione discreta, composta da un insieme di pixel di un dispositivo di output



Rasterizzazione

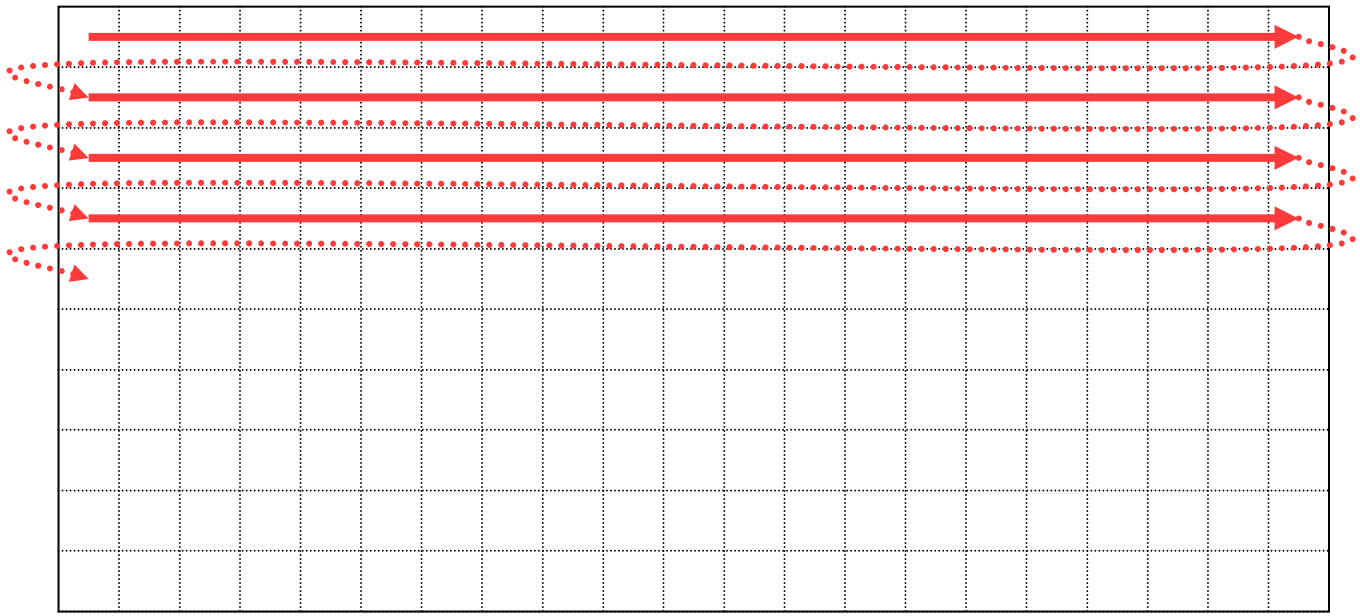
- Concetto molto simile al gioco che prevede un piano rettangolare con tanti fori disposti in maniera regolare e chiodini colorati;
- Se il piano è sufficientemente grande e si hanno abbastanza tonalità di colore a disposizione, l'immagine finale che ne risulta non è troppo diversa dall'originale.



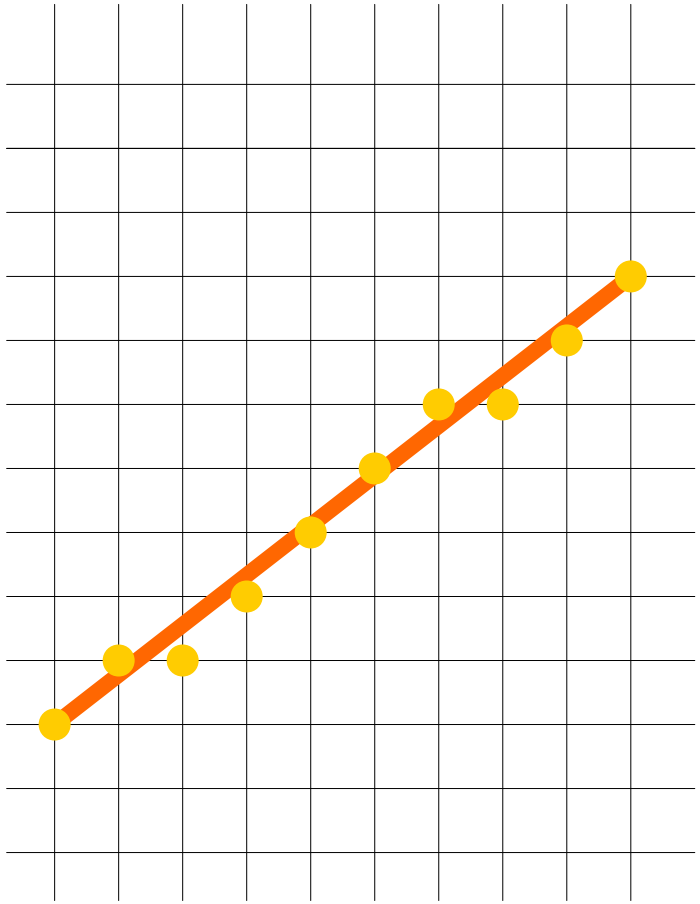


Rasterizzazione

- Gli algoritmi di rasterizzazione si dicono anche di *scan-conversion* dal nome delle linee (*scan-line*) di pixel che compongono l'immagine raster sul dispositivo di output.

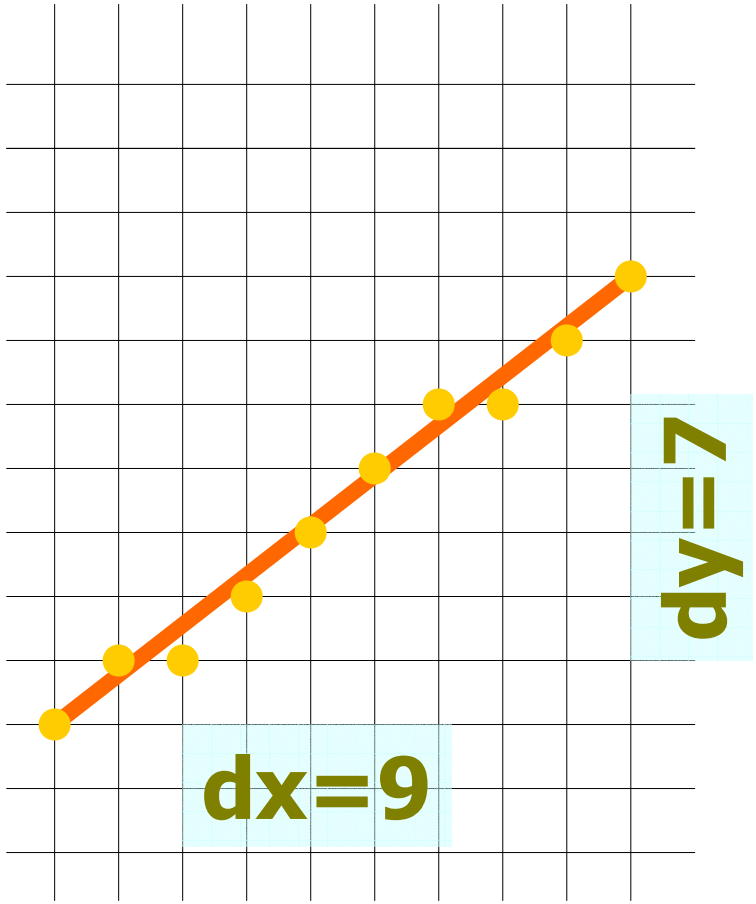


Rasterizzazione di Segmenti



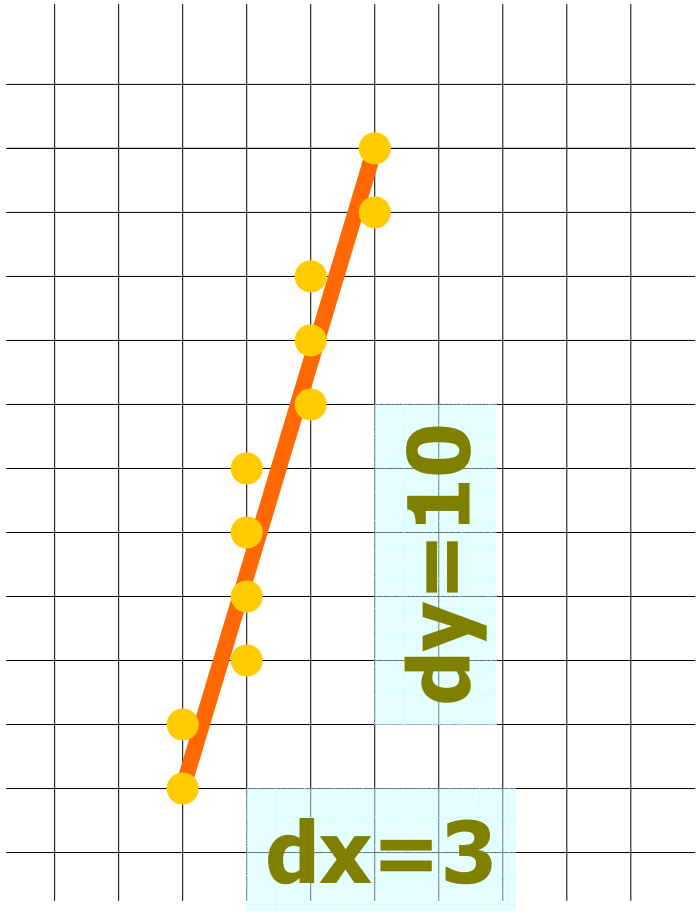
- L'algoritmo di rasterizzazione di un segmento di retta deve individuare le coordinate dei pixel che giacciono sulla linea ideale o che sono il più vicino possibile ad essa
- la sequenza di pixel deve approssimare al meglio il segmento.

Rasterizzazione di Segmenti



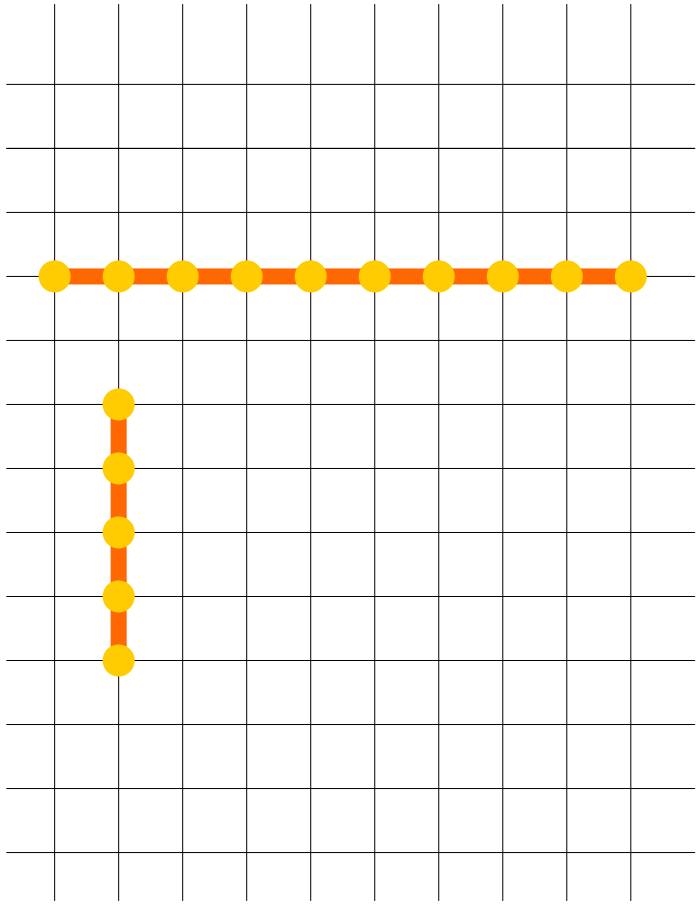
- Lo spessore minimo del segmento rasterizzato (idealmente nullo) risulterà di un pixel;
- Per coefficienti angolari $|m| \leq 1$ la rasterizzazione presenta un pixel per ogni colonna.

Rasterizzazione di Segmenti



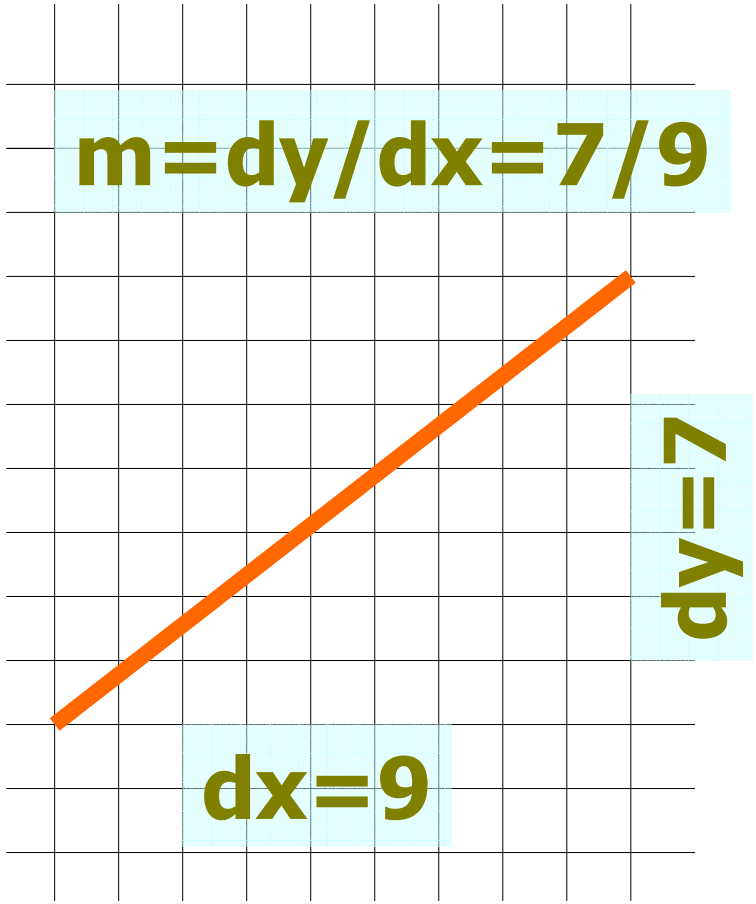
- Per coefficienti angolari $|m|>1$ la rasterizzazione presenta un pixel per ogni riga.

Rasterizzazione di Segmenti



- Banale la rasterizzazione di segmenti orizzontali o verticali (sequenze di pixel su una riga o una colonna).

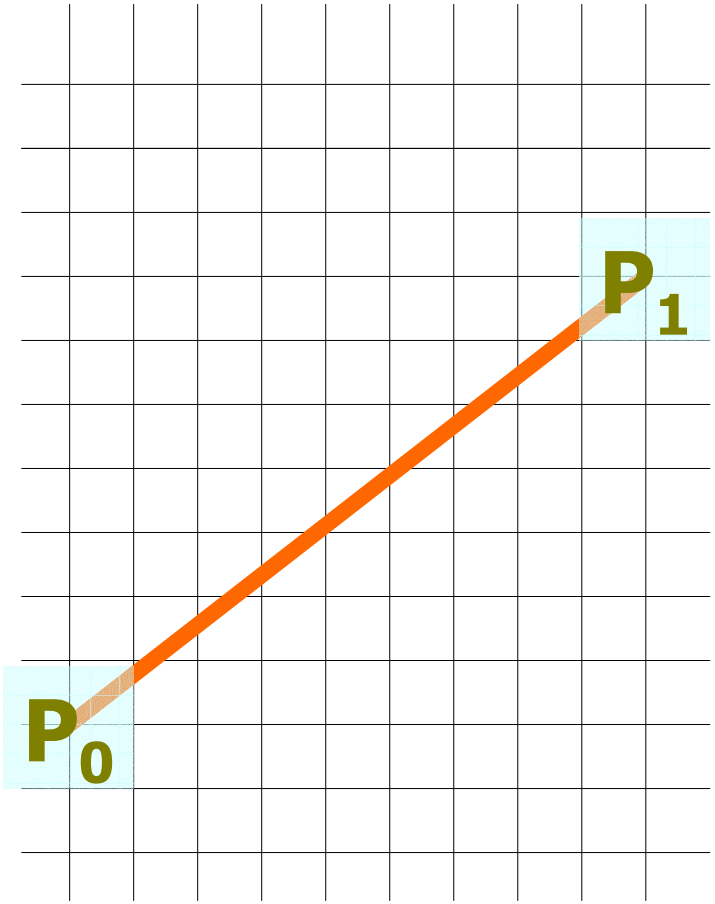
Rasterizzazione di Segmenti
Soluzione Analitica



- Senza perdere di generalità ci limiteremo nel seguito al caso di segmenti con coefficienti angolari $m \leq 1$;
- L'espressione analitica della retta su cui giace il segmento è:

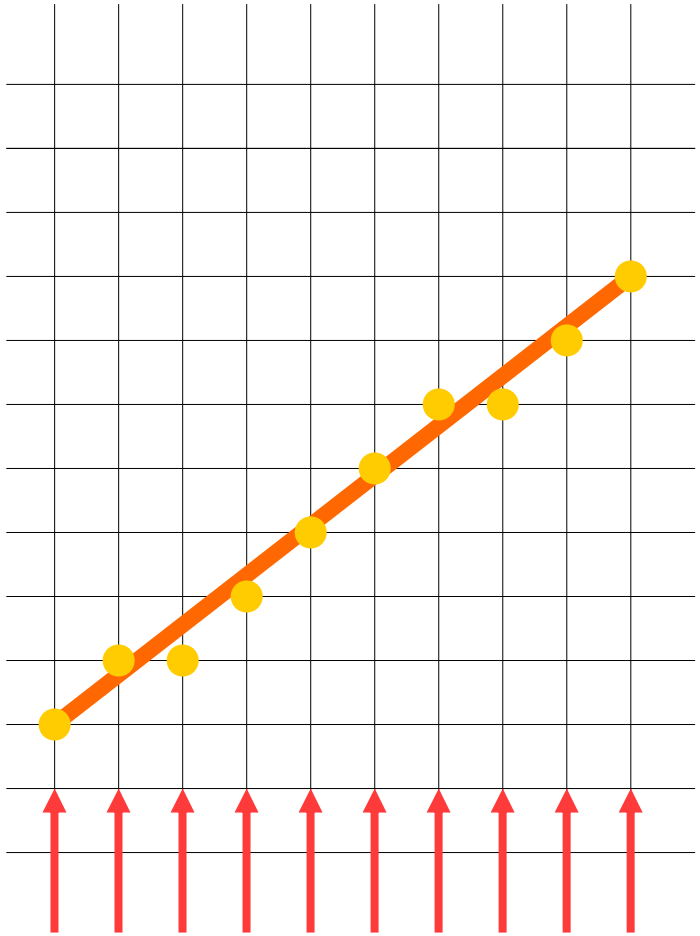
$$y = \mathrm{m}x + \mathrm{B}$$

Rasterizzazione di Segmenti
Soluzione Analitica



- Si intende rasterizzare il segmento di estremi $P_0=(x_0,y_0)$ e $P_1=(x_1,y_1)$;
- Entrambi gli estremi presentano coordinate intere.

Rasterizzazione di Segmenti
Soluzione Analitica



- 1. A partire dal pixel con coordinata *x* minima x_0:
  - 2.1 Incrementare *x* con passo costante uguale a 1;
  - 2.2 ∀ valore assunto dall'ascissa (x_i), calcolare y_i come $y_i = mx_i + B$;
  - 2.3 Arrotondare y_i all'ordinata intera più vicina.

Rasterizzazione di Segmenti
Soluzione Analitica

- L'algoritmo analitico seleziona il pixel più vicino alla linea ideale, il pixel cioè che ha distanza minima dalla linea;
- L'individuazione di un pixel implica 3 operazioni: una moltiplicazione (mx_i), un'addizione (mx_i+B) , ed un arrotondamento (y_i).

- La moltiplicazione può essere eliminata utilizzando una tecnica incrementale: il punto sulla retta può essere individuato sulla base del punto precedente
- L'algoritmo che ne deriva prende il nome di *algoritmo DDA* (*digital differential analyzer*)

Rasterizzazione di Segmenti
Algoritmo DDA

- Notando che:

$$y_{i+1} = \mathrm{m}x_{i+1} + \mathrm{B}$$

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

$$y_{i+1} = \mathrm{m}x_i + \mathrm{B} + \mathrm{m}\Delta x$$

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

- e che $\Delta x = 1$

- si ha $y_{i+1} = y_i + \mathrm{m}$

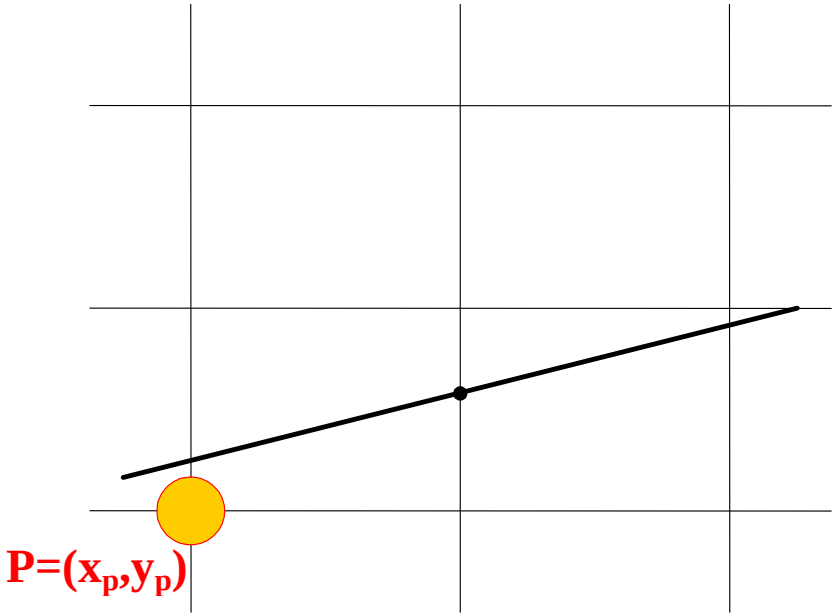
- Quindi, per ogni punto della linea, abbiamo:

$$x_{i+1} = x_i + 1 \quad \Rightarrow \quad y_{i+1} = y_i + \mathrm{m}$$

- Ad ogni passo è necessaria una operazione di arrotondamento con variabili (e l'aritmetica) in virgola mobile;
- L'impiego di aritmetica floating point implica introduzione e propagazione di errore.

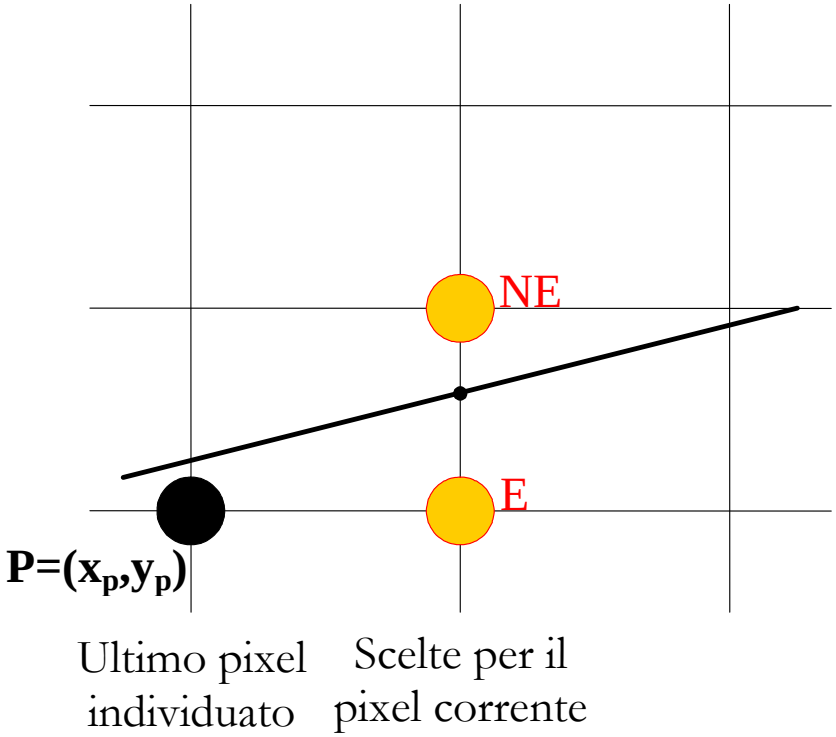
- L'algoritmo di Bresenham (detto anche *algoritmo del punto di mezzo*) risolve il problema dell'errore introdotto dall'uso di aritmetica floating point nell'algoritmo DDA;
- L'algoritmo di Bresenham fa uso solo di operazioni in aritmetica intera;
- E' ancora un algoritmo di tipo differenziale; fa uso delle informazioni calcolate per individuare il pixel al passo *i* per individuare il pixel al passo *i+1.*

- Nel seguito, ancora l'ipotesi non restrittiva m<1.

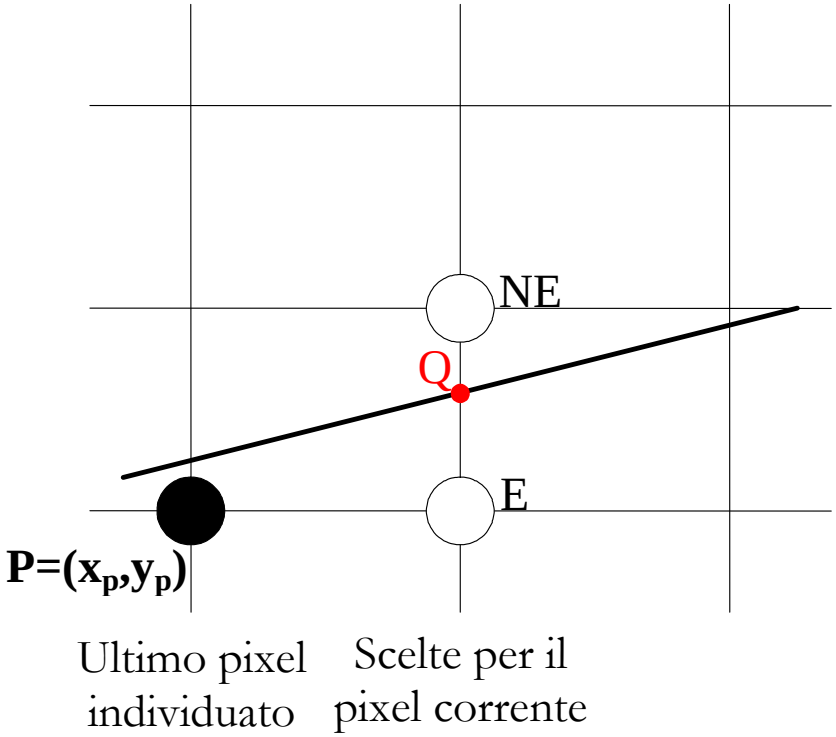


Ultimo pixel individuato

- Supponiamo che l'ultimo pixel individuato dal processo di rasterizzazione sia il pixel *P* di coordinate $P=(x_p, y_p)$

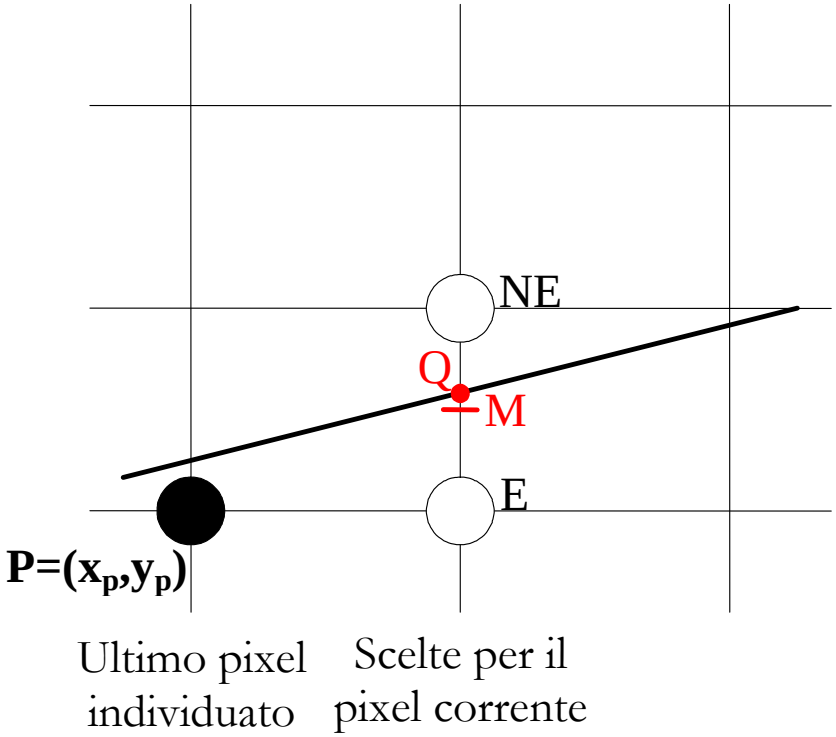


- Il prossimo pixel della rasterizzazione sarà il pixel immediatamente a destra di *P* (*E*, per *east pixel*) oppure quello in alto a destra (*NE*, per *north-east pixel*).
- La scelta del prossimo pixel è limitata a due sole possibilità



- Indichiamo con Q il punto in cui il segmento interseca la retta $x = x_p+1$. Il prossimo pixel è quello, tra E e NE, con distanza minima da Q.
- La scelta di un pixel è ricondotta alla misura di una distanza.

Rasterizzazione di Segmenti
Algoritmo di Bresenham



- Detto M il punto di mezzo del segmento E-NE, si deve scegliere il punto che sta dalla stessa parte di Q rispetto ad M;
- Dobbiamo quindi definire da che parte è Q rispetto ad M.
- La scelta di un pixel è ricondotta all'analisi della relazione geometrica tra due punti.

- Il problema è quindi definire da che parte si trova Q (intersezione del segmento con la retta $x = x_p+1$) rispetto a M (punto medio tra i centri dei pixel E ed NE);
- Conviene utilizzare la forma implicita dell'equazione della retta:

$$F(x, y) = ax + by + c = 0$$

- Poiché m = dy/dx; $dx = x_1 - x_0$; $dy = y_1 - y_0$ la forma esplicita può essere riscritta come:

$$y = mx + \mathrm{B}$$

$$y = \frac{dy}{dx}x + \mathrm{B}$$

$$y = \frac{y_1 - y_0}{x_1 - x_0}x + \mathrm{B}$$

- Quindi

$$y = \frac{dy}{dx} x + \mathrm{B}$$

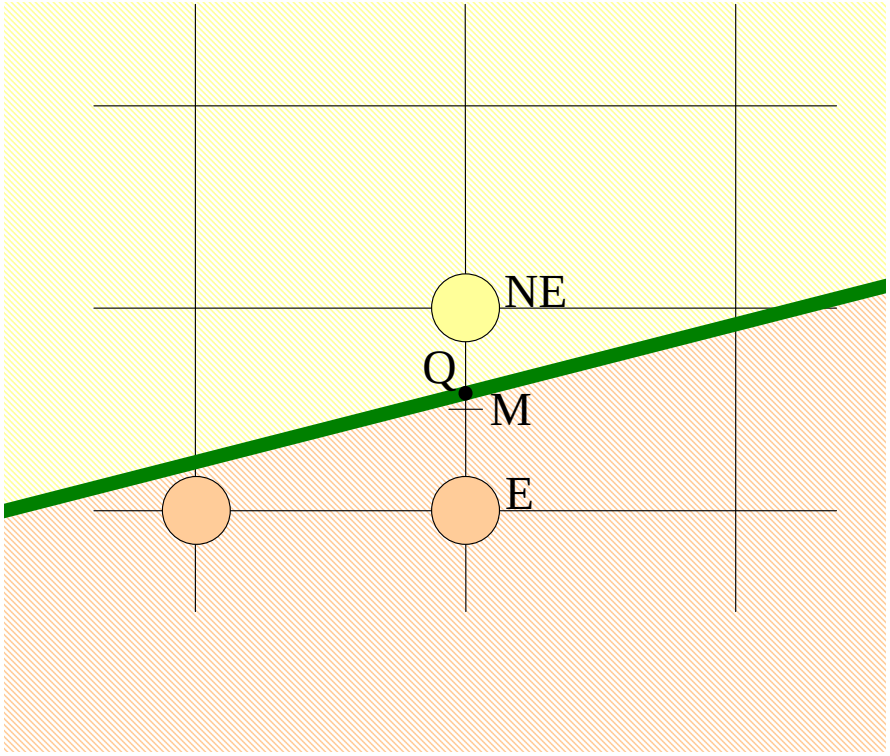
$$dx \cdot y = dy \cdot x + \mathrm{B} \cdot dx$$

$$dy \cdot x - dx \cdot y + \mathrm{B} \cdot dx = 0$$

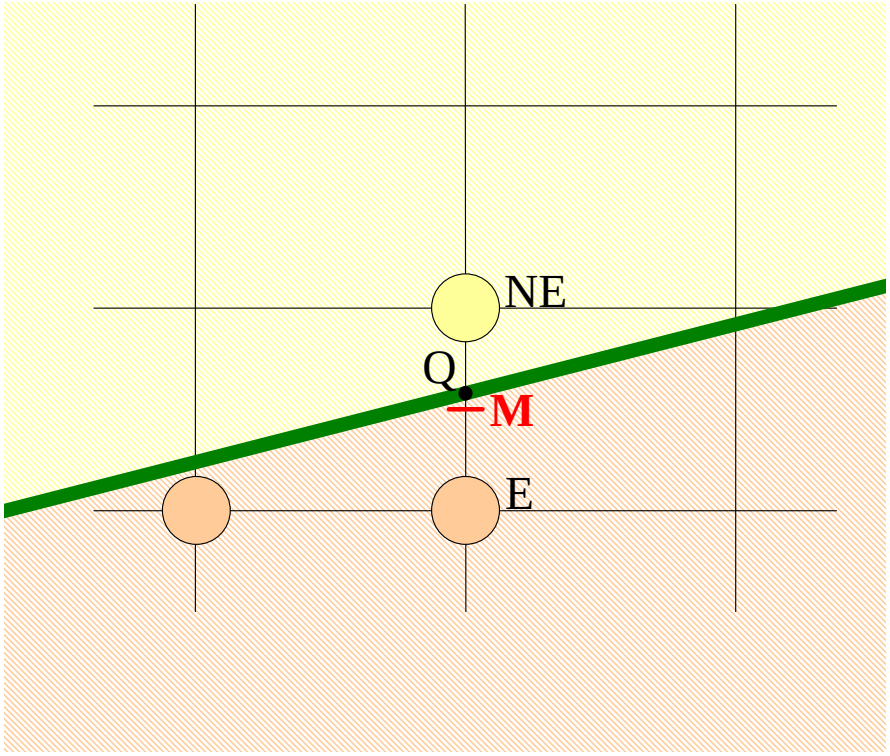
$$\mathrm{F}(x, y) = dy \cdot x - dx \cdot y + \mathrm{B} \cdot dx = 0$$

- con

$$\mathrm{a} = dy; \quad \mathrm{b} = -dx; \quad \mathrm{c} = \mathrm{B} \cdot dx$$



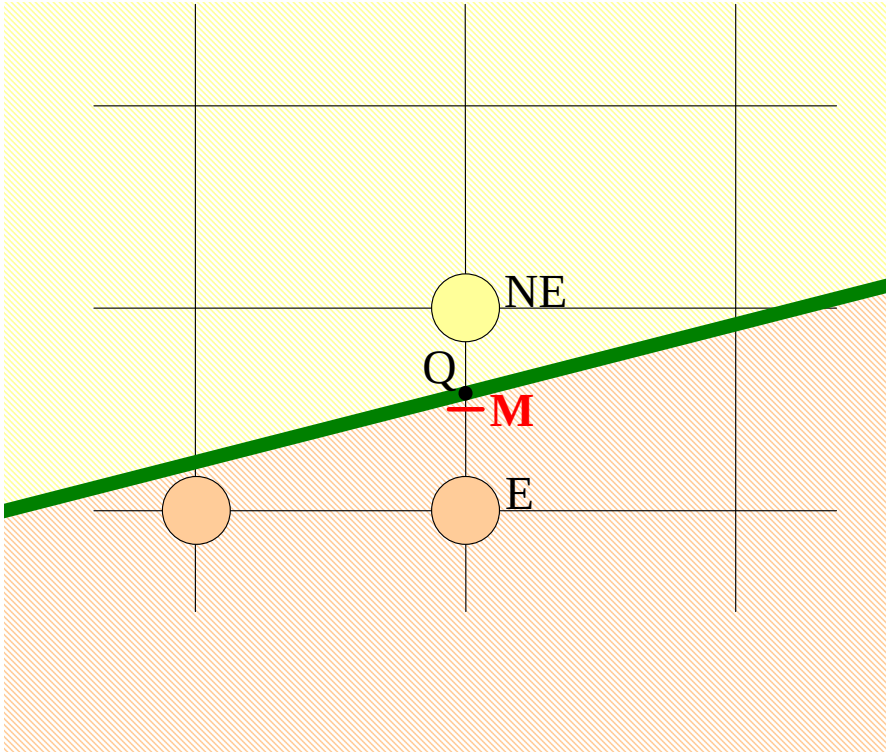
- La funzione F:
  - vale 0 per tutti i punti della retta;
  - assume valori positivi sotto la retta;
  - assume valori negativi sopra la retta.
- F(Q)=0



- La scelta tra E e NE si riduce alla valutazione del segno della funzione F nel punto M.

$$\mathrm{F(M)} = \mathrm{F}(x_p + 1, y_p + \tfrac{1}{2})$$

- L'analisi della relazione geometrica tra due punti si riduce quindi a valutare il segno di una funzione.

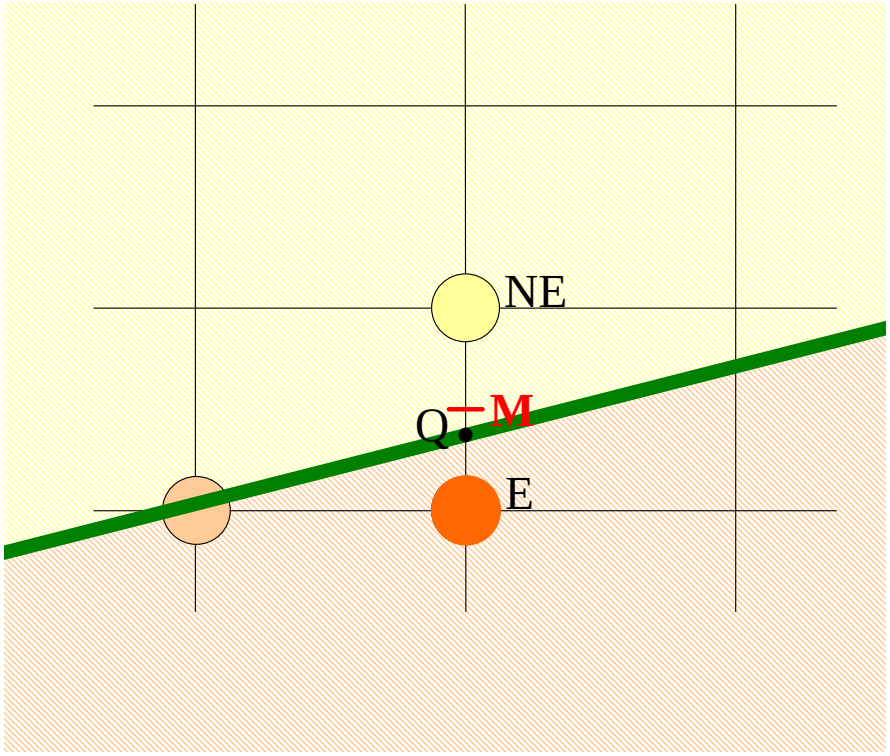


- La decisione si basa sul segno di F(M); indichiamo F(M) come *variabile di decisione d*.

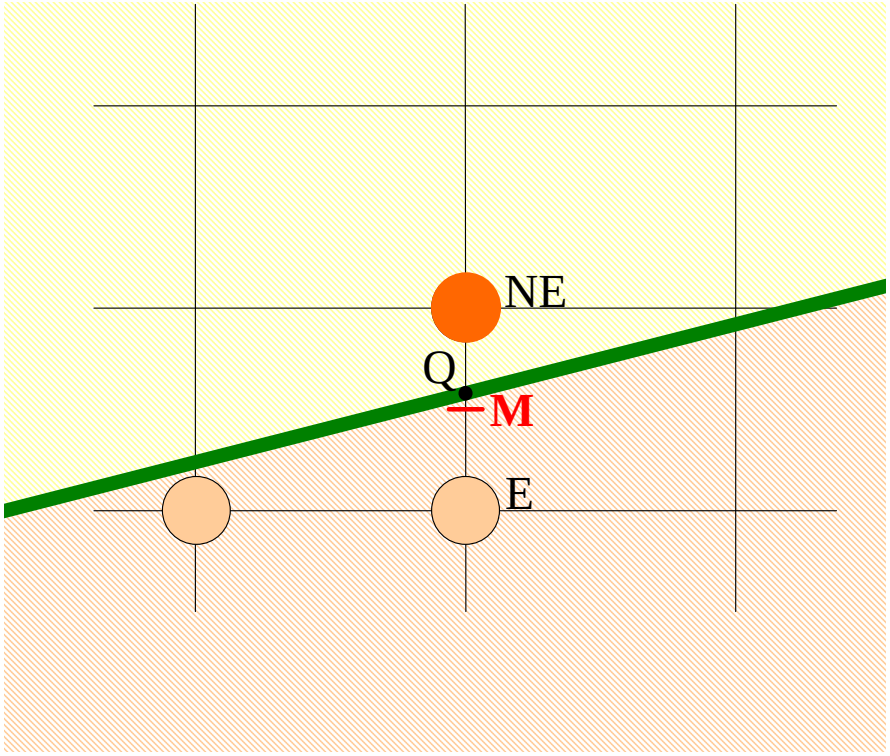
$$d = F(M)$$

$$d = F(x_p + 1, y_p + \tfrac{1}{2})$$

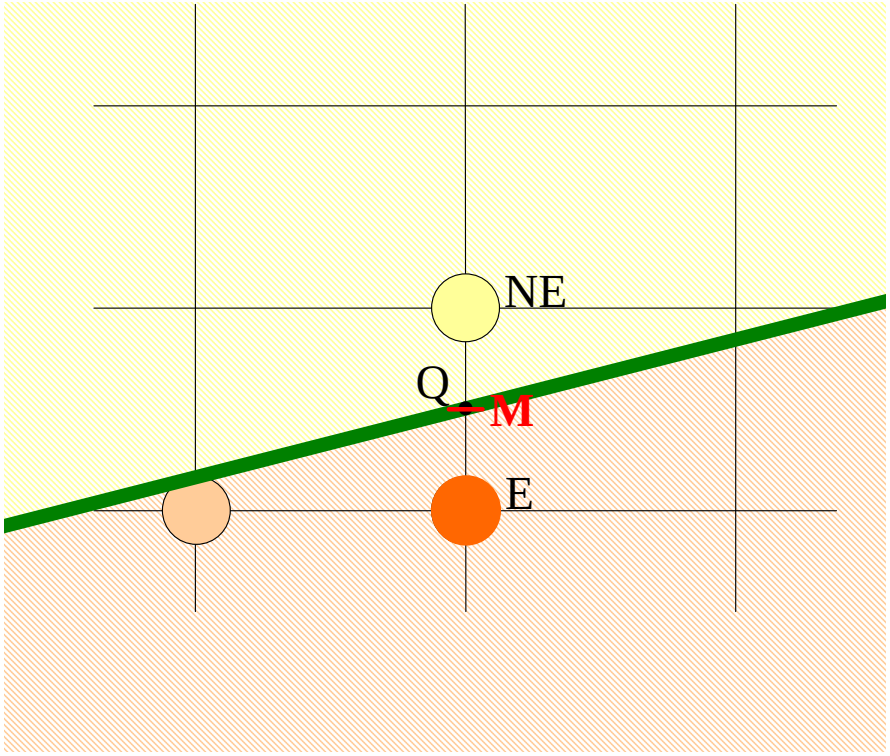
$$d = a(x_p + 1) + b(y_p + \tfrac{1}{2}) + c$$



- Se $d < 0$
  - M giace sopra la retta;
  - Scegliamo E come prossimo pixel della rasterizzazione.

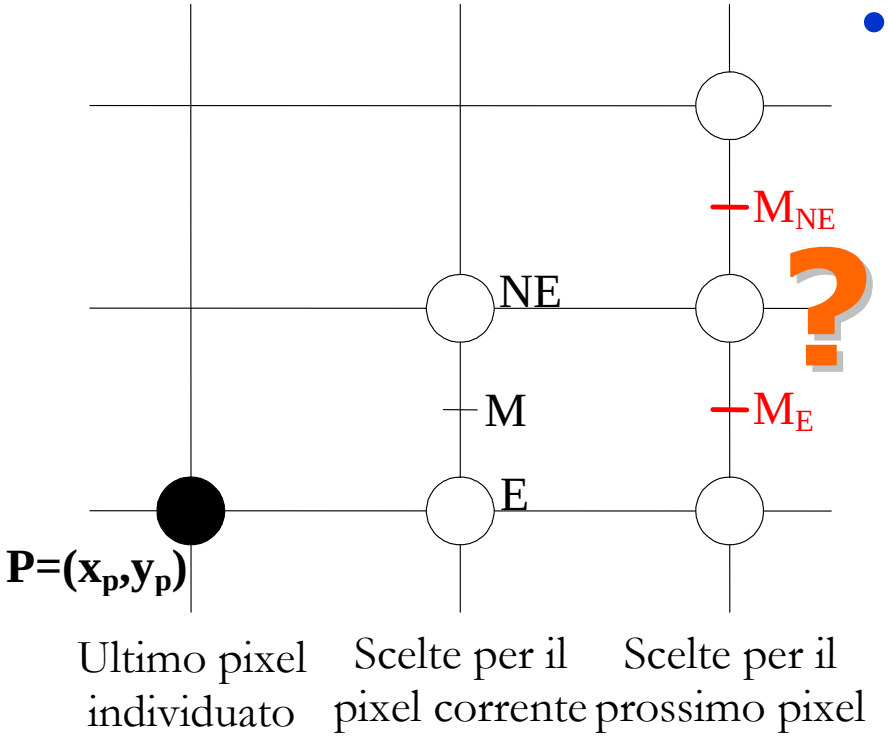


- Se $d > 0$
  - M giace sotto la retta;
  - Scegliamo NE come prossimo pixel della rasterizzazione.



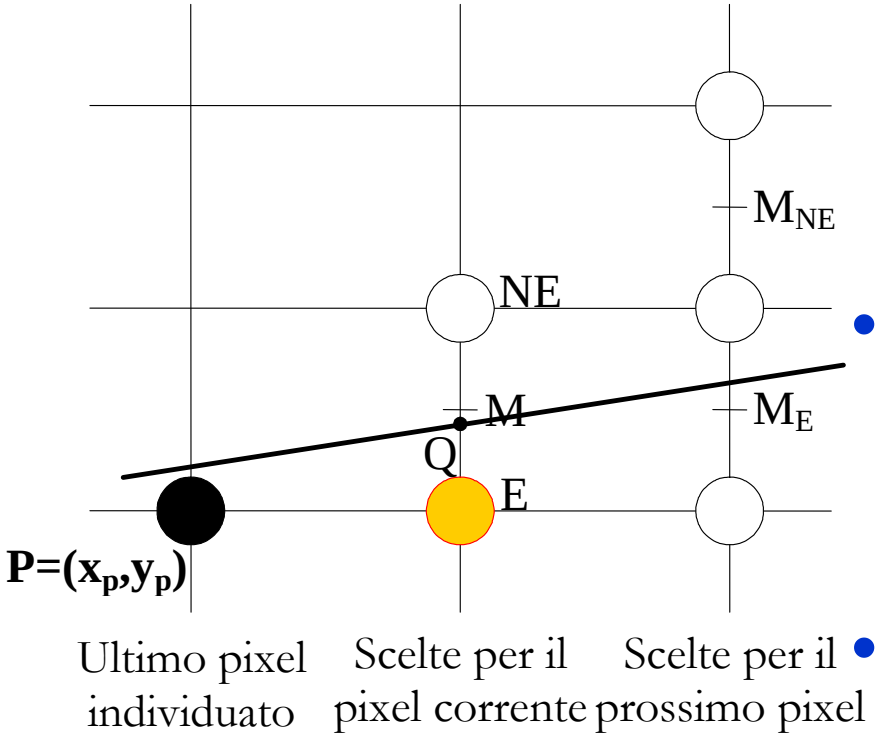
- Se $d = 0$
  - M appartiene alla retta (Q≡M);
  - Scegliamo come prossimo pixel della rasterizzazione uno qualsiasi tra E ed NE
  - Supponiamo che E sia scelto.

Rasterizzazione di Segmenti
Algoritmo di Bresenham



- L'algoritmo di Bresenham costruisce anche *d* in modo incrementale.
- A tal fine è necessario individuare il punto M al prossimo passo ($x = x_p+2$) sulla base della scelta fatta al passo corrente.

Rasterizzazione di Segmenti
Algoritmo di Bresenham



- Se l'ultimo pixel selezionato è stato E

$$d_{new} = F(x_p + 2, y_p + \tfrac{1}{2})$$

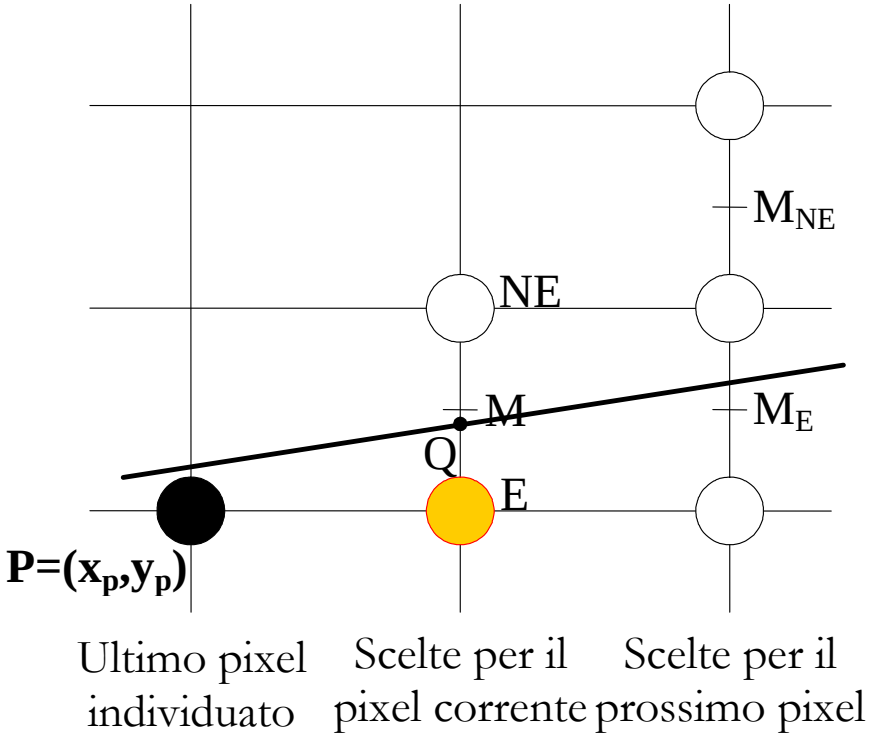
$$d_{new} = a(x_p + 2) + b(y_p + \tfrac{1}{2}) + c$$

- poiché

$$d = a(x_p + 1) + b(y_p + \tfrac{1}{2}) + c$$

- sottraendo si ha

$$d_{new} = d_{old} + a$$

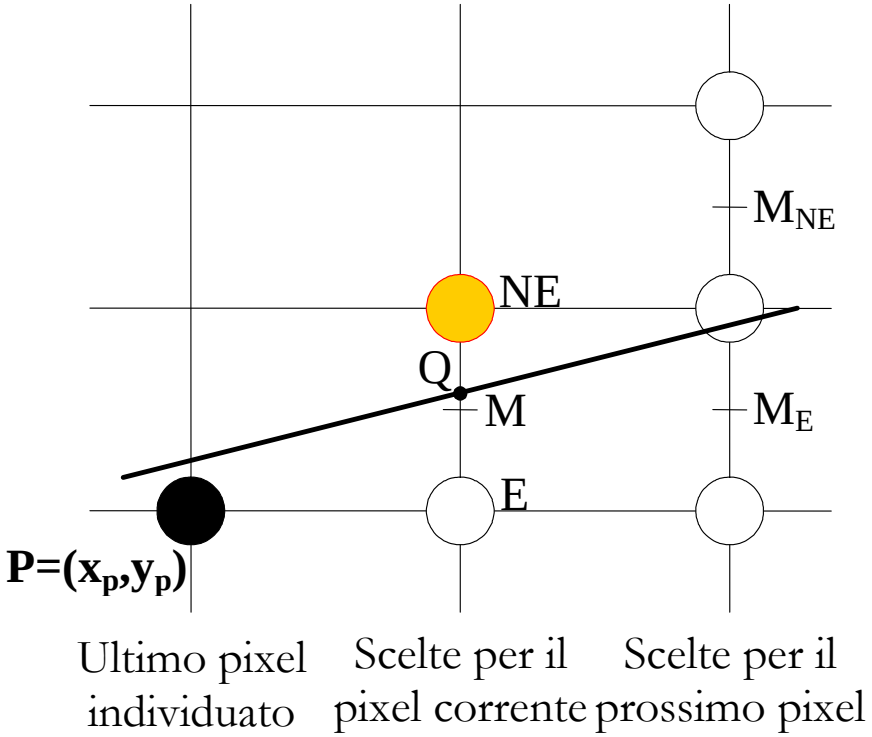


- L'incremento da aggiungere a *d* dopo aver scelto E è quindi:

$$\Delta_E = a = dy$$

- Questo risultato vale ad ogni passo della rasterizzazione.

Rasterizzazione di Segmenti
Algoritmo di Bresenham



- Se invece l'ultimo pixel selezionato è stato NE:

$$d_{new} = F(x_p + 2, y_p + \tfrac{3}{2})$$

$$d_{new} = a(x_p + 2) + b(y_p + \tfrac{3}{2}) + c$$

- quindi

$$d_{new} = d_{old} + a + b$$

- da cui

$$\Delta_{NE} = a + b = dy - dx$$

- Ad ogni passo l'algoritmo sceglie il prossimo pixel tra due possibili candidati basandosi sul valore corrente di una variabile *d* di decisione;
- ricalcola il valore della variabile di decisione incrementalmente aggiungendo al suo valore corrente una quantità fissa predefinita (Δ_E o Δ_{NE});
- Il valore iniziale per *d* risulta:

$$F(x_0+1, y_0+\tfrac{1}{2}) = a(x_0+1)+b(y_0+\tfrac{1}{2})+c$$

$$F(x_0+1, y_0+\tfrac{1}{2}) = ax_0+by_0+c+a+\frac{b}{2}$$

$$F(x_0+1, y_0+\tfrac{1}{2}) = F(x_0, y_0)+a+\frac{b}{2}$$

- Poiché (x_0, y_0) appartiene al segmento $\mathrm{F}(x_0, y_0) = 0$ quindi:

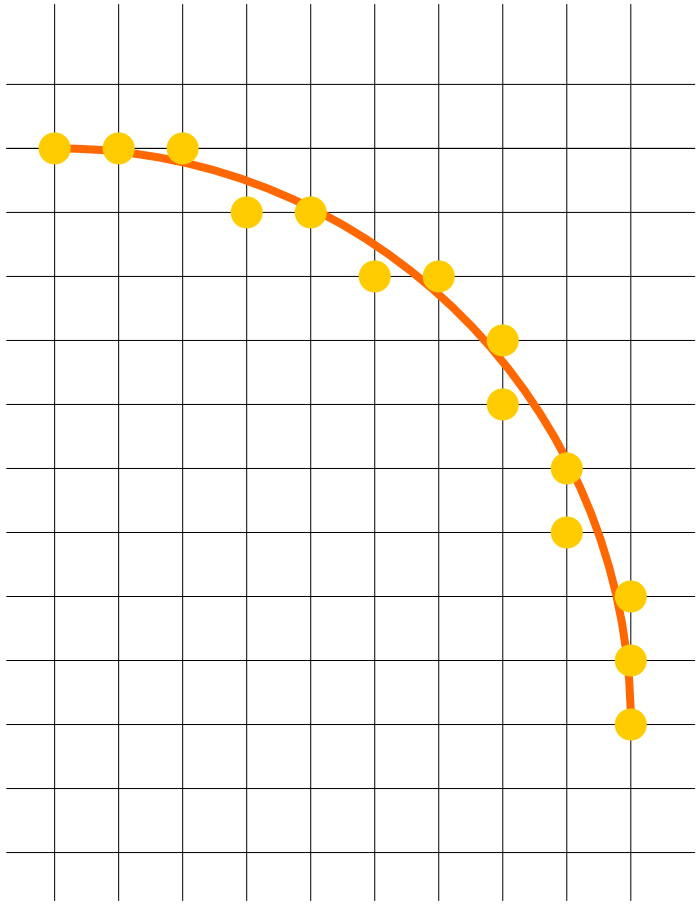
$$\mathrm{F}(x_0 + 1, y_0 + \tfrac{1}{2}) = \mathrm{F}(x_0, y_0) + \mathrm{a} + \frac{\mathrm{b}}{2}$$

$$\mathrm{d}_{\mathrm{start}} = \mathrm{a} + \frac{\mathrm{b}}{2} = dy - \frac{dx}{2}$$

- La frazione può essere eliminata utilizzando come funzione decisionale la funzione $2\mathrm{F}$.

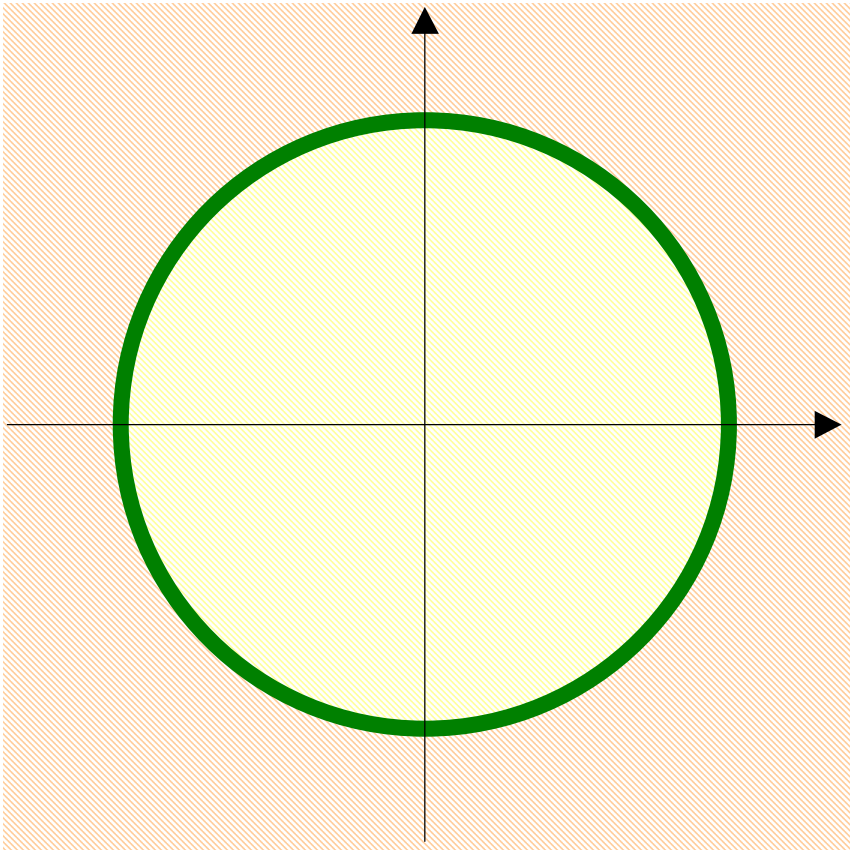
- L’algoritmo di Bresenham si dice **algoritmo differenziale del primo ordine**;
- Il primo ordine si riferisce ai passi “in avanti” (in effetti sono derivazioni della funzione) utilizzati per calcolare la differenza tra i valori della funzione;
- Il suo parente matematico più stretto è un’equazione differenziale del primo ordine.

Rasterizzazione di Circonferenze



- Così come per i segmenti l'algoritmo di rasterizzazione di una circonferenza o di un arco deve calcolare le coordinate dei pixel che giacciono il più possibile vicino ad essa

Rasterizzazione di Circonferenze
Algoritmo di Bresenham

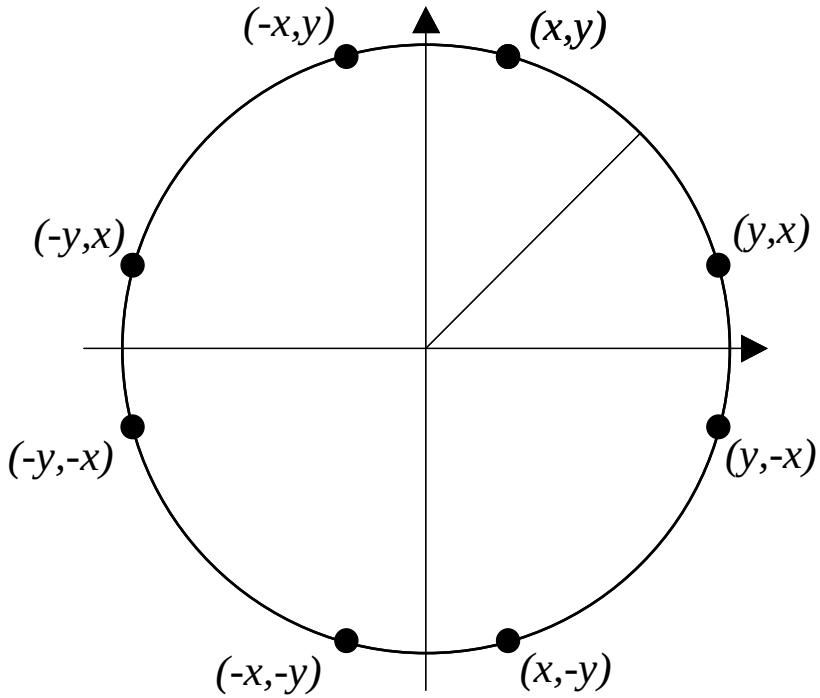


- Molto simile all'algoritmo di Bresenham per segmenti;
- Si considera la funzione implicita:

$$F(x, y) = x^2 + y^2 - R^2$$

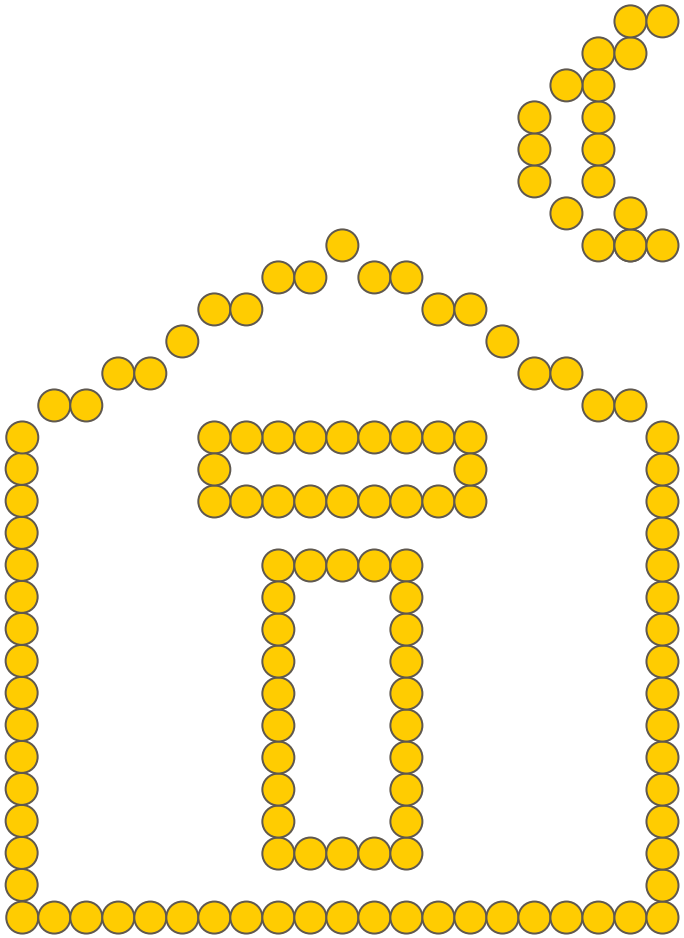
- che vale:
  - F=0 sulla circonferenza;
  - F<0 al suo interno;
  - F>0 al suo esterno.

Rasterizzazione di Circonferenze (Ottimizzazione)



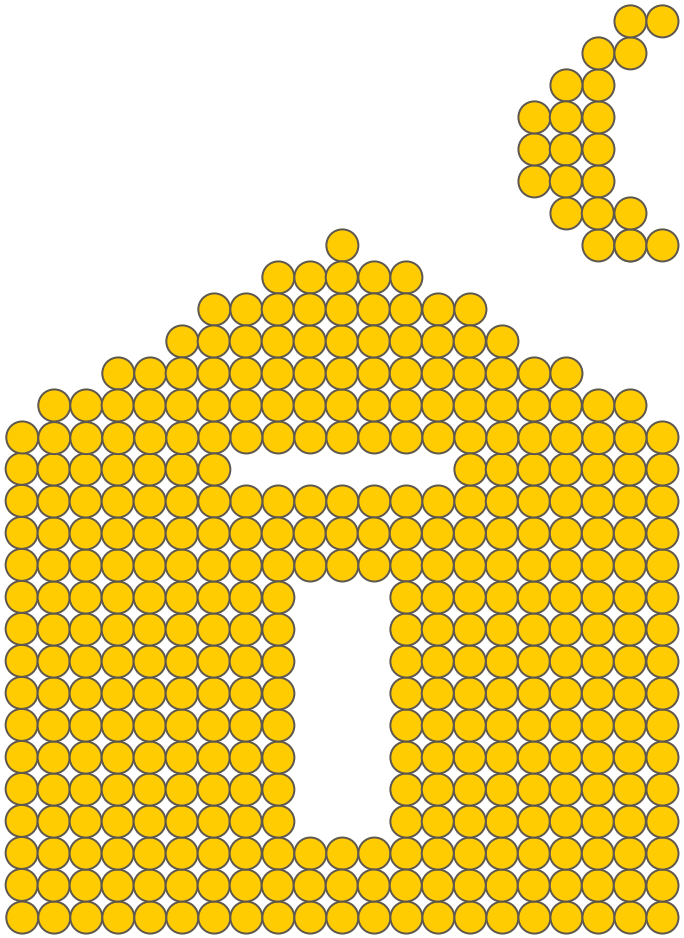
- La rasterizzazione di una circonferenza può essere effettuata per ***ottanti***;
- Se un pixel (*x*, *y*) rappresenta la circonferenza, anche gli altri 7 pixel da esso derivati per simmetria radiale cambiando di segno e scambiando tra loro *x* e *y* la rappresentano.
- Questa ottimizzazione vale per tutti gli algoritmi di rasterizzazione di circonferenze, non solo per Bresenham

Rasterizzazione di Poligoni



- La tecnologia raster permette di “disegnare” primitive geometriche rappresentate dal solo contorno e primitive “piene”.

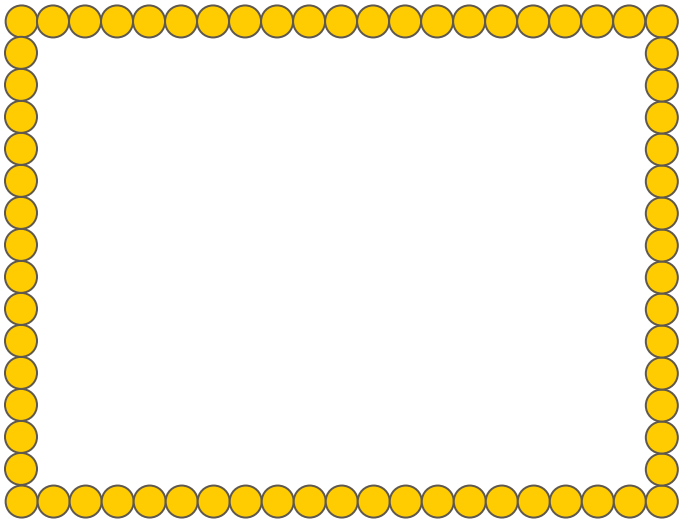
Rasterizzazione di Poligoni



- La tecnologia raster permette di “disegnare” primitive geometriche rappresentate dal solo contorno e primitive “piene”.

Rasterizzazione di Rettangoli (banale)

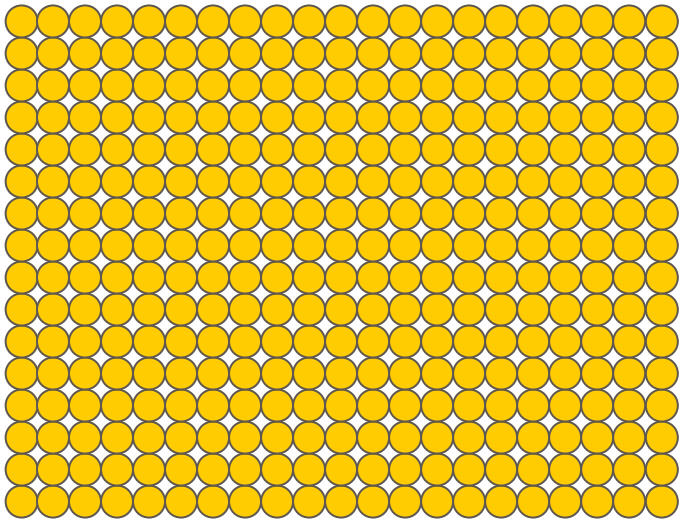
- Disegnare un rettangolo *vuoto* consiste nell'applicare 4 volte l'algoritmo di rasterizzazione dei segmenti che ne rappresentano i lati;



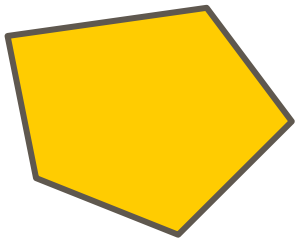
Rasterizzazione di Rettangoli (banale)

- Per disegnare un rettangolo *pieno,* con lati paralleli agli assi cartesiani, è sufficiente innescare un doppio ciclo di "accensione" dei pixel interni al rettangolo

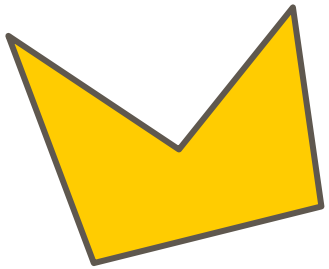
```
for y from y_min to y_max of the rectangle
    for x from x_min to x_max
        WritePixel(x, y)
```



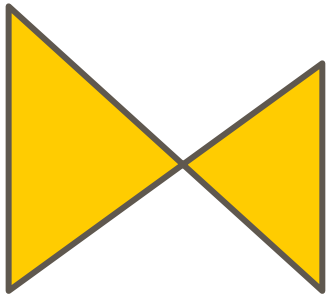
Rasterizzazione di Poligoni



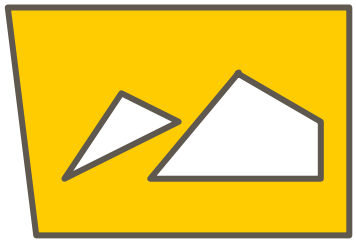
Convesso



Concavo



Intrecciato



Contorni multipli

- L'algoritmo di rasterizzazione deve operare correttamente sulle diverse tipologie di primitive geometriche:
  - Poligono Convesso
  - Poligono Concavo
  - Poligono Intrecciato
  - Poligono A Contorni multipli

Rasterizzazione di Poligoni

- Non ci interessa entrare nel dettaglio della rasterizzazione di poligoni generici
- Ci interessa sapere che l'hardware grafico implementa rasterizzazione efficiente di triangoli, punti e linee
- Tali algoritmi funzionano male per triangoli stretti e lunghi

Domande?