Grafica Computazionale

Rimozione delle superfici nascoste

Fabio Ganovelli
fabio.ganovelli@gmail.com
a.a. 2006-2007

Dalle diapositive a corredo del libro: "Fondamenti di Grafica Tridimensionale Interattiva"
R. Scateni, P. Cignoni, C. Montani e R. Scopigno - McGrawHill Italia

Argomenti trattati

- Rimozione delle superfici nascoste;
  - Approccio object-space e image-space;
  - Rimozione efficiente delle superfici nascoste;
  - L'algoritmo *z-buffer;*
  - L'algoritmo scan-line;
  - L'algoritmo del pittore *(depth-sort)*
  - L'algoritmo di Warnock.

Rimozione superfici nascoste (HSR)

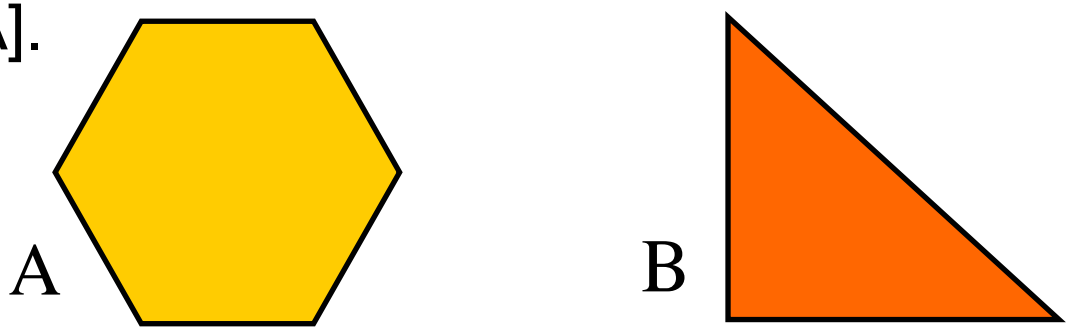
- Gli oggetti della scena sono generalmente opachi;
- Gli oggetti più vicini all'osservatore possono nascondere (occludere) la vista (totale o parziale) di oggetti più lontani;
- Il problema della rimozione delle superfici nascoste (HSR, Hidden surface removal) consiste nel determinare le parti della 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.

HSR

- Gli algoritmi per la rimozione delle superfici nascoste si possono classificare in:
  - 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 del piano immagine**), la primitiva geometrica visibile “attraverso” quel punto. Gli algoritmi operano nello spazio immagine della scena proiettata.

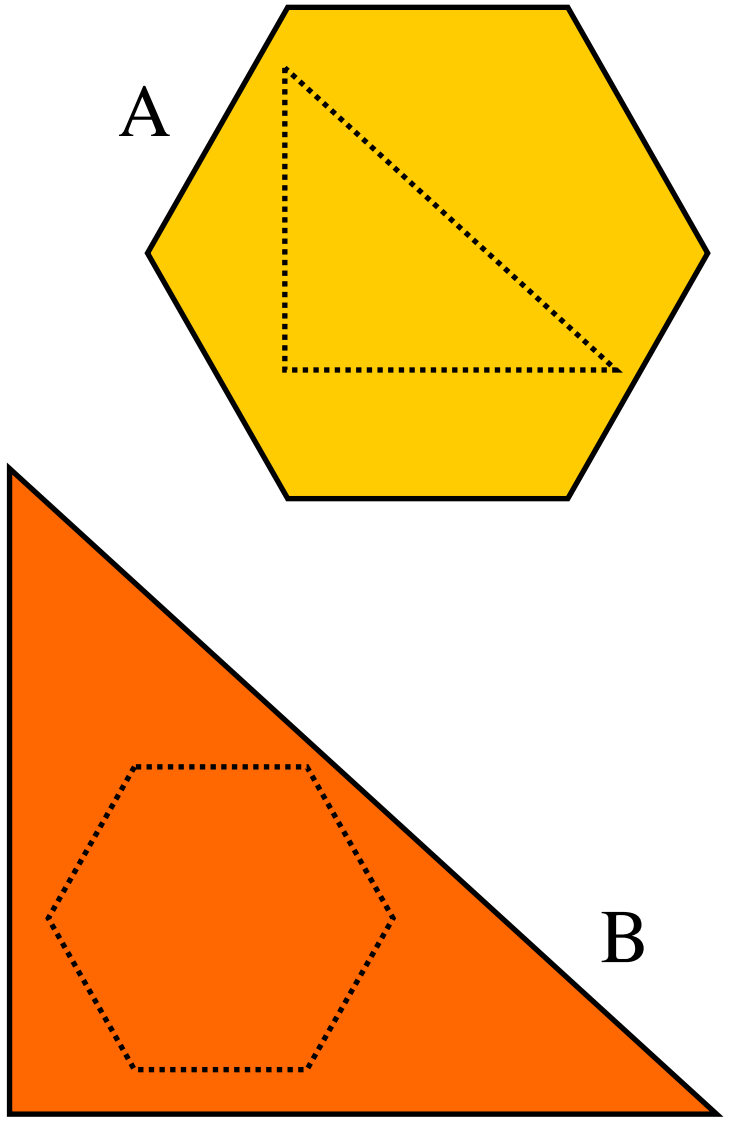
HSR: Object-space

- 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:
  - A [B] oscura B [A]; solo A [B] è visualizzata;
  - A e B sono completamente visibili; entrambe le primitive sono visualizzate;
  - A [B] occlude parzialmente B [A]: è necessario individuare le parti visibili di B [A].



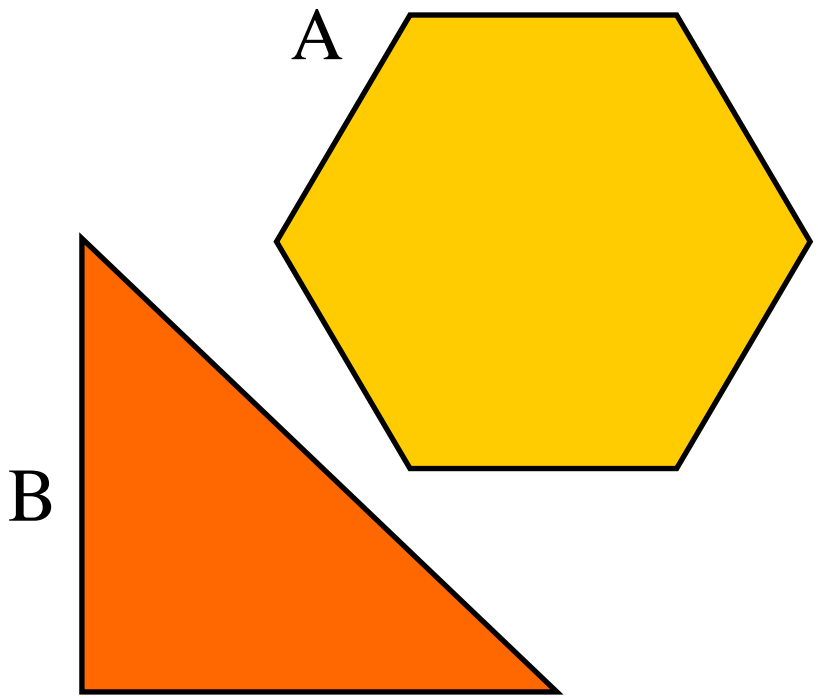
HSR: Object-space

- A oscura B: tutti i punti di A sono più vicini all'osservatore di tutti i punti di B e la proiezione di B sul piano di vista ricade all'interno della proiezione di A. Visualizza A;
- B oscura A: tutti i punti di B sono più vicini all'osservatore di tutti i punti di A e la proiezione di A sul piano di vista ricade all'interno della proiezione di B. Visualizza B.



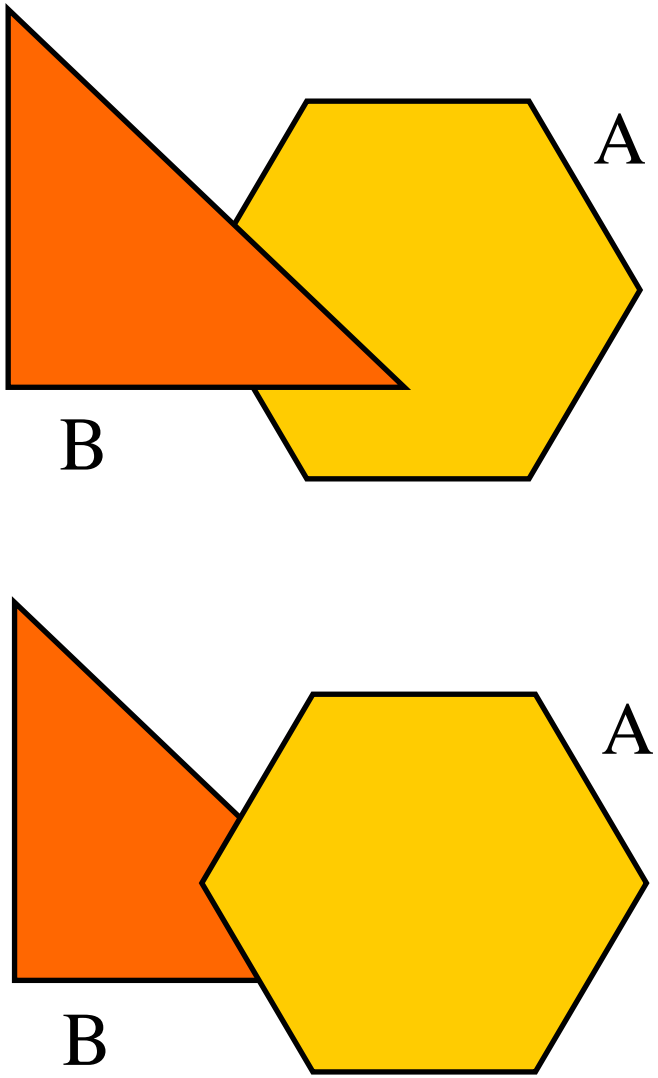
HSR: Object-space

- A e B non si occludono: le due proiezioni di A e B sul piano di vista sono disgiunte;
- Visualizza A e B.



HSR: Object-space

- A e B si oscurano parzialmente: l'intersezione tra le due proiezioni di A e B sul piano di vista è non nulla e diversa sia dalla proiezione di A che da quella di B.
- Individua le parti visibili di ciascuna primitiva.

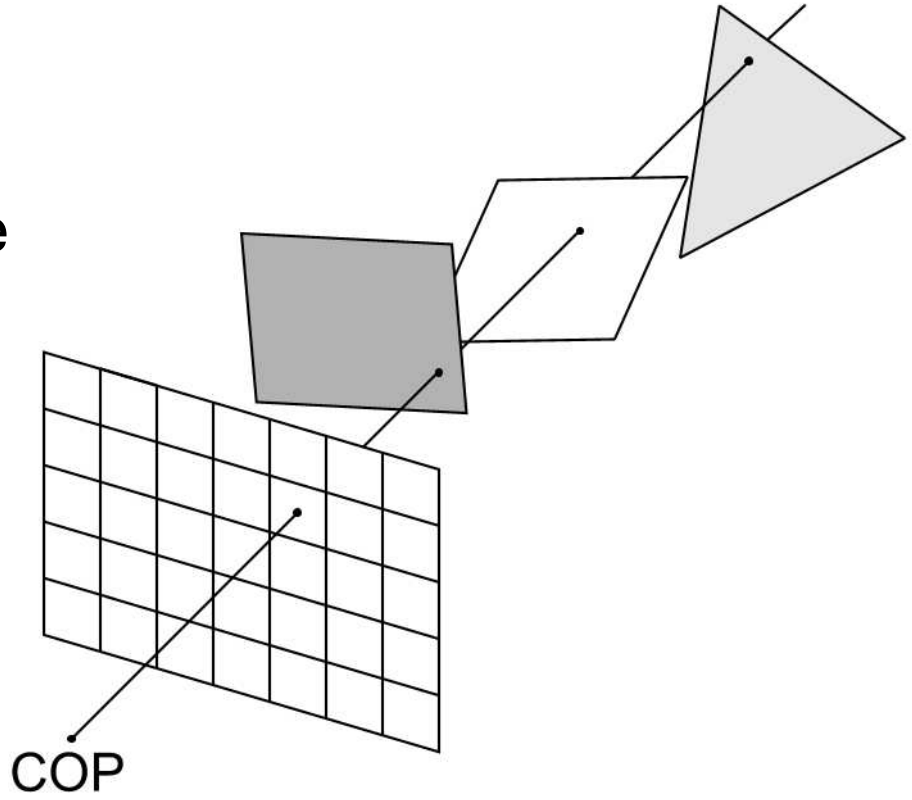


HSR: Object-space

- Un possibile algoritmo risolutivo:
  - 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 di ordine $O(k^2)$
- L'approccio object-space è consigliabile solo quando le primitive nella scena sono relativamente poche.

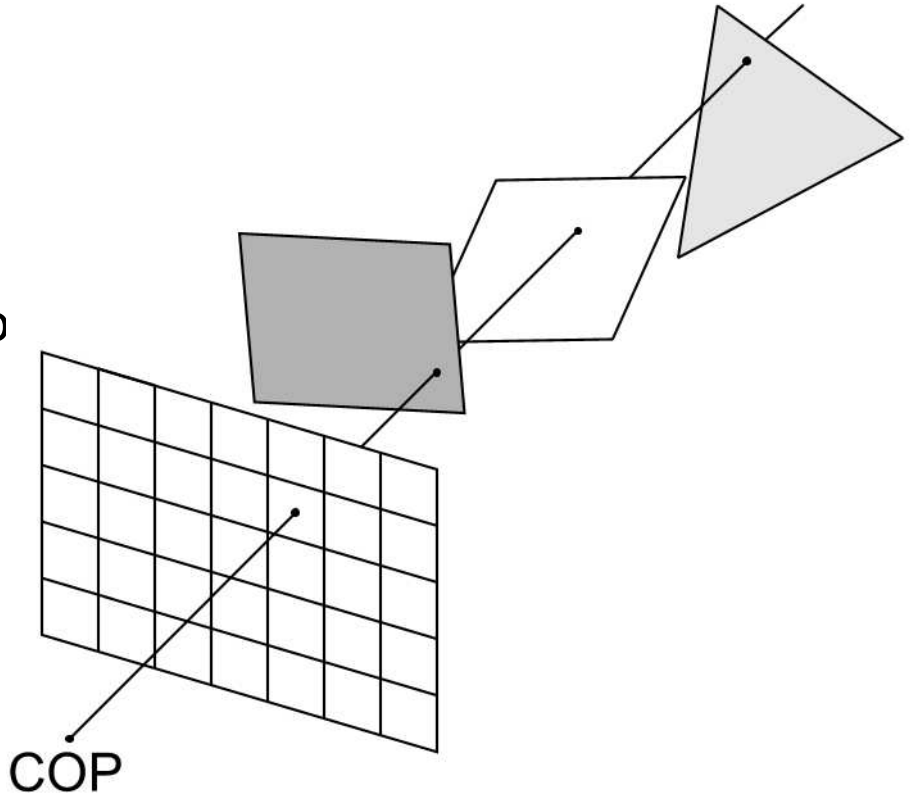
HSR: Image-space

- 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:
  - Se l'intersezione è interna alla primitiva si memorizza la distanza del punto di intersezione dal piano di vista;
  - Se l'intersezione è esterna viene immediatamente scartata.
- Tra le distanze accumulate si sceglie la minore (l'intersezione più vicina al 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 semirette e piani di appartenenza delle primitive (per ogni semiretta al più *k* intersezioni);
- Anche se 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).

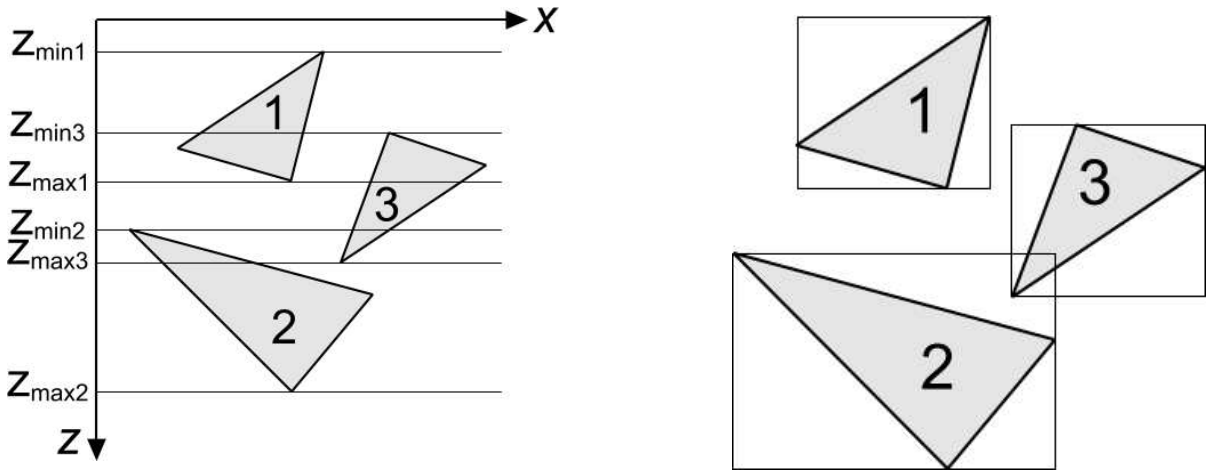
- Sia nell'approccio object-space che in quello image-space la rimozione delle superfici nascoste è riconducibile ad un problema di ordinamento (in profondità).

HSR: Aumentare l'efficienza

- Coerenza
  - Coerenza a livello di oggetto;
  - Coerenza in profondità (parti adiacenti di una stessa primitiva sono "vicine" in profondità).
- Trasformazioni prospettiche e parallele
  - Il punto $P_1(x_1,y_1,z_1)$ oscura $P_2=(x_2,y_2,z_2)$ se P_1 e P_2 sono sullo stesso proiettore;
  - Nella proiezione parallela la verifica è a $x_1=x_2$ e $y_1=y_2$;
  - Per una proiezione prospettica occorre verificare che $x_1/z_1=x_2/z_2$ e $y_1/z_1=y_2/z_2$;
  - Può convenire applicare una trasformazione geometrica globale in modo tale che la proiezione parallela della scena trasformata sia uguale alla proiezione prospettica non trasformata.

HSR: Aumentare l'efficienza

- Extent, bounding box e bounding volume
  - Extent 1D, bounding box 2D e bounding volume 3D si rivelano fondamentali per l'ordinamento. Se non c'è sovrapposizione tra i bounding box allora non occorre confrontare le relative primitive (condizione necessaria ma non sufficiente!).

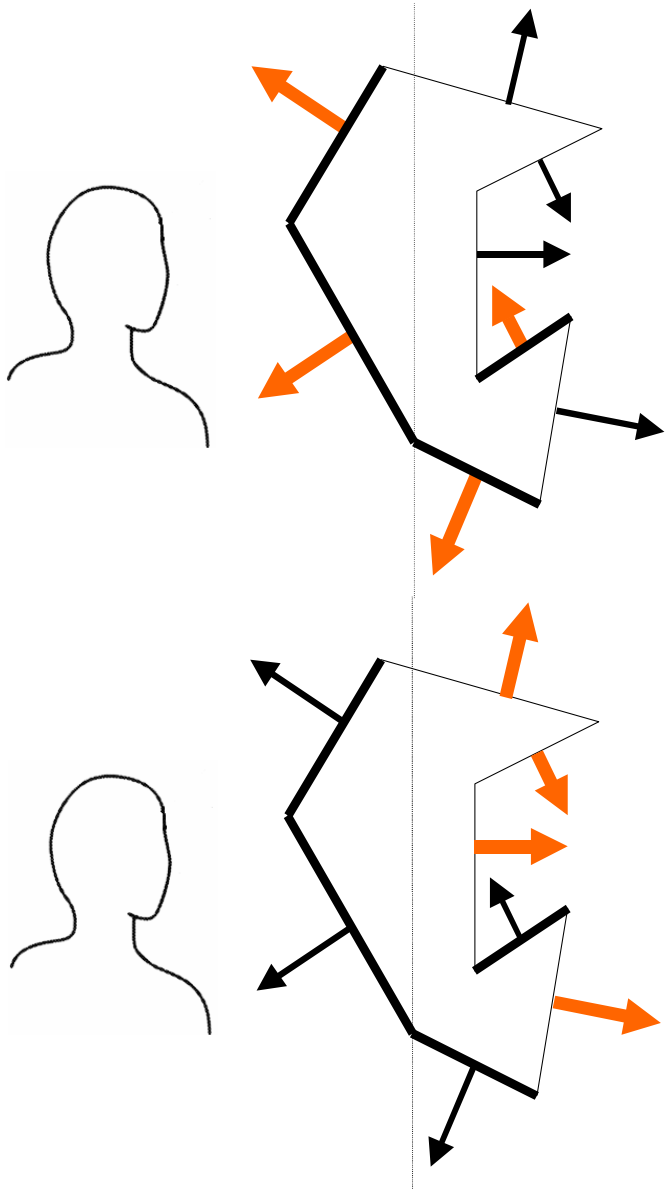


HSR: Aumentare l'efficienza (Back-face culling)

- Back-face culling (Eliminazione delle facce posteriori)
  - Se gli oggetti della scena sono rappresentati da poliedri solidi chiusi (cioè le *facce poligonali* dei poliedri delimitano completamente i volumi dei solidi);
  - Se ogni *faccia poligonale* è stata modellata in modo tale che la normale ad essa sia diretta verso l'esterno del poliedro di appartenenza;
  - Se nessuna parte del poliedro viene tagliata dal *front clipping plane* ...

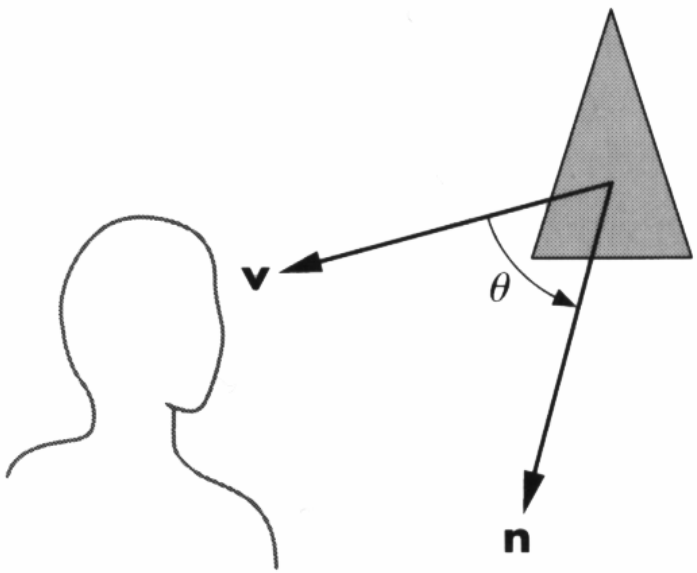
HSR: Aumentare l'efficienza (Back-face culling)

- Nelle ipotesi precedenti…
- Le facce la cui normale forma angoli superiori a ±90° con la direzione di vista possono essere visibili;
- Le facce la cui normale forma angoli inferiori a ±90° con la direzione di vista certamente non sono visibili;



HSR: Aumentare l'efficienza (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 deve essere rimossa se $-90° \leq \theta \leq 90°$, cioè se $\cos\theta \geq 0$.
- Invece di calcolare la quantità $\cos\theta$ possiamo valutare il prodotto scalare $n \cdot v \geq 0$



HSR: Aumentare l'efficienza (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, sempre in media, circa metà delle facce di un poliedro sono *back-facing* e quindi la loro visualizzazione sarebbe comunque inutile.

HSR: Aumentare l'efficienza

- Suddivisione spaziale
  - Riorganizzare gli oggetti della scena (o le loro proiezioni sul piano immagine) in gruppi coerenti dal punto di vista spaziale;
  - Si effettua di solito mediante griglie 2D o 3D che servono a limitare i confronti in profondità tra le primitive della scena.

HSR: L'algoritmo *z-buffer (o depth-buffer)*

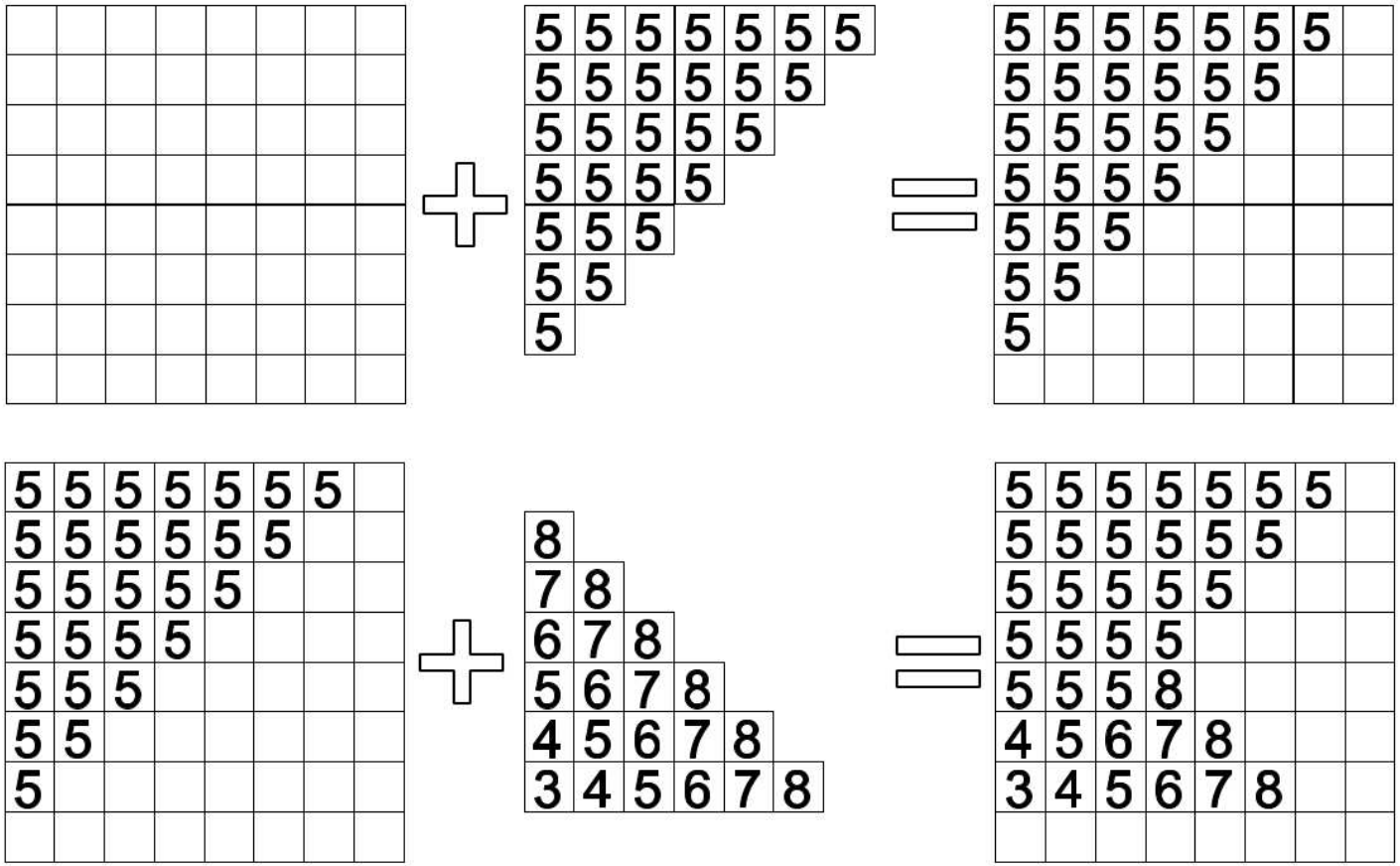
- L'algoritmo **z-buffer** è un algoritmo di tipo image-space, basato su una logica molto semplice e facile da realizzare;
- Lavora in accoppiamento con l'algoritmo di *scan conversion* (rasterizzazione, disegno, delle primitive geometriche sulla memoria di quadro) ed ha bisogno, come struttura dati di supporto, di un'area di memoria, il *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.

HSR: L'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. La rasterizzazione avviene dopo la proiezione sul piano di vista, nello spazio 3D NDC;
- Se la profondità z in (x,y) è inferiore alla profondità corrente memorizzata nello z-buffer allora (x,y) assume z come profondità corrente ed il pixel (x,y) nel frame buffer assume il valore colore della primitiva in esame.

HSR: L'algoritmo *z-buffer*

- Esempio

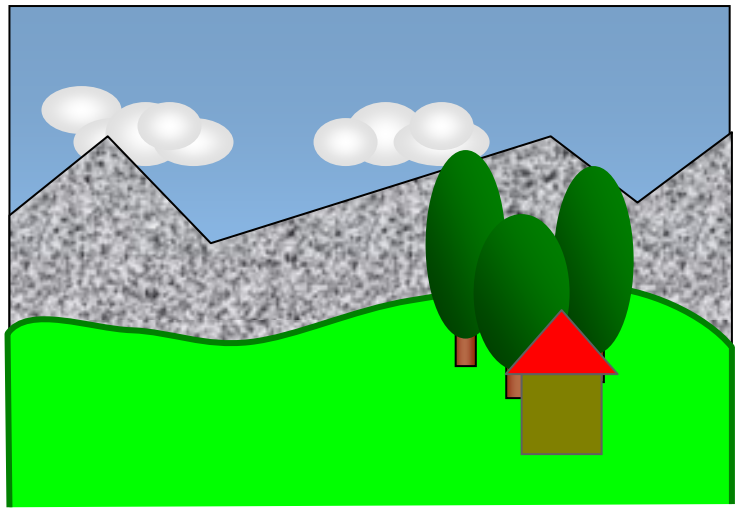


HSR: L'algoritmo *z-buffer*

- Lo z-buffer ha la stessa risoluzione del frame buffer ed ogni cella ha dimensione sufficiente per memorizzare la informazioni di profondità alla risoluzione richiesta (di solito 24 bit);
- 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;
- Generalmente implementato firmware;
- Complessità pressoché costante (ad un aumento delle primitive corrisponde in genere una diminuzione della superficie coperta).

HSR: 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.

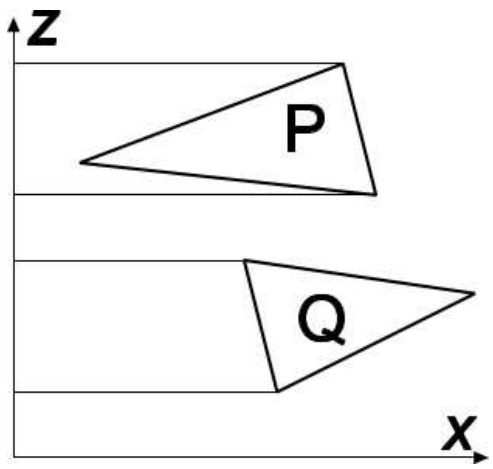


HSR: Algoritmo del pittore (*depth sort)*

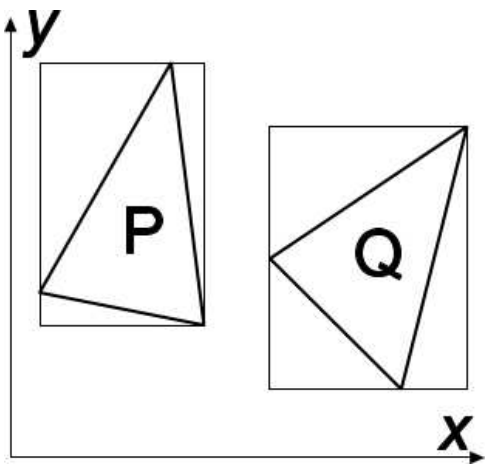
- L'algoritmo:
  - Individuare un ordinamento in profondità (lungo la direzione di vista) delle primitive della scena (*depth sort,* ordinamento in profondità). Ordinamento effettuato in object-space.
  - Visualizzare le primitive della scena in modalità *back-to-front.* Rasterizzazione delle primitive 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.

HSR: Algoritmo del pittore (*depth sort)*

- 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;

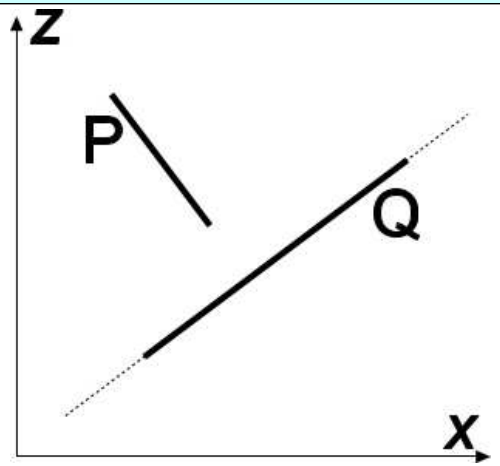


- Altrimenti, se i bounding box di P e Q sul piano xy non interferiscono, allora P precede Q;

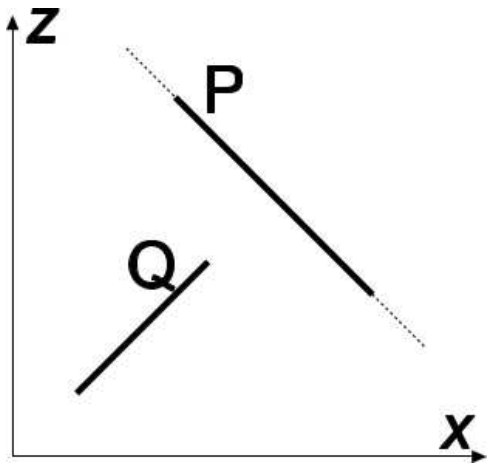


HSR: Algoritmo del pittore (*depth sort)*

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

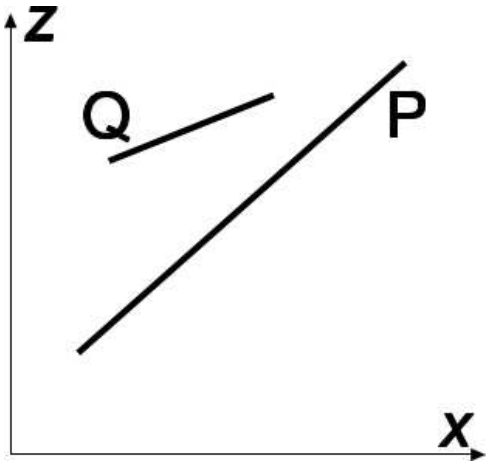


- Altrimenti, se tutti i vertici di Q si trovano dalla stessa parte dell'osservatore rispetto al piano individuato da P, allora P precede Q;
- Altrimenti, se le proiezioni di P e Q sul piano immagine non interferiscono, allora P precede Q.

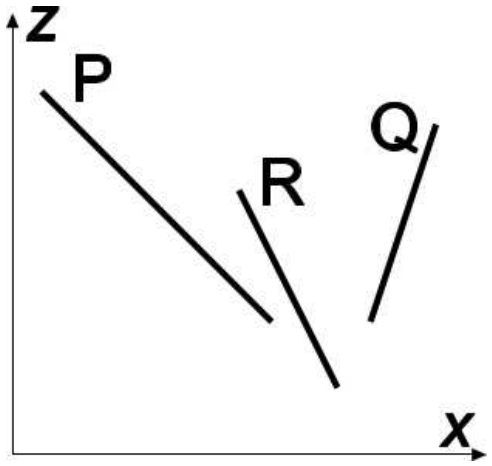


HSR: Algoritmo del pittore (*depth sort)*

- 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;

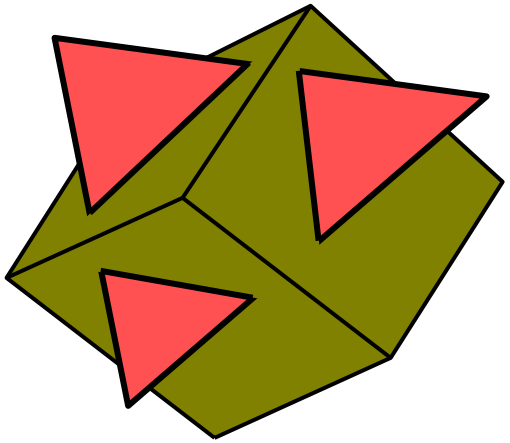


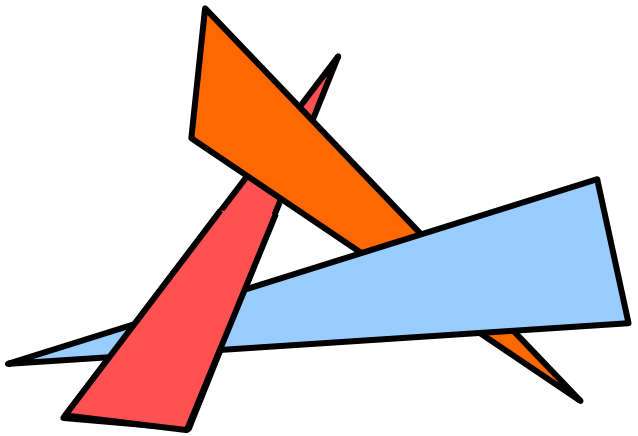
- Esempio: l'ordinamento iniziale P, Q, R è sovvertito dallo scambio di P con R e quindi dallo scambio di R con Q.



HSR: Algoritmo del pittore (*depth sort)*

- 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).



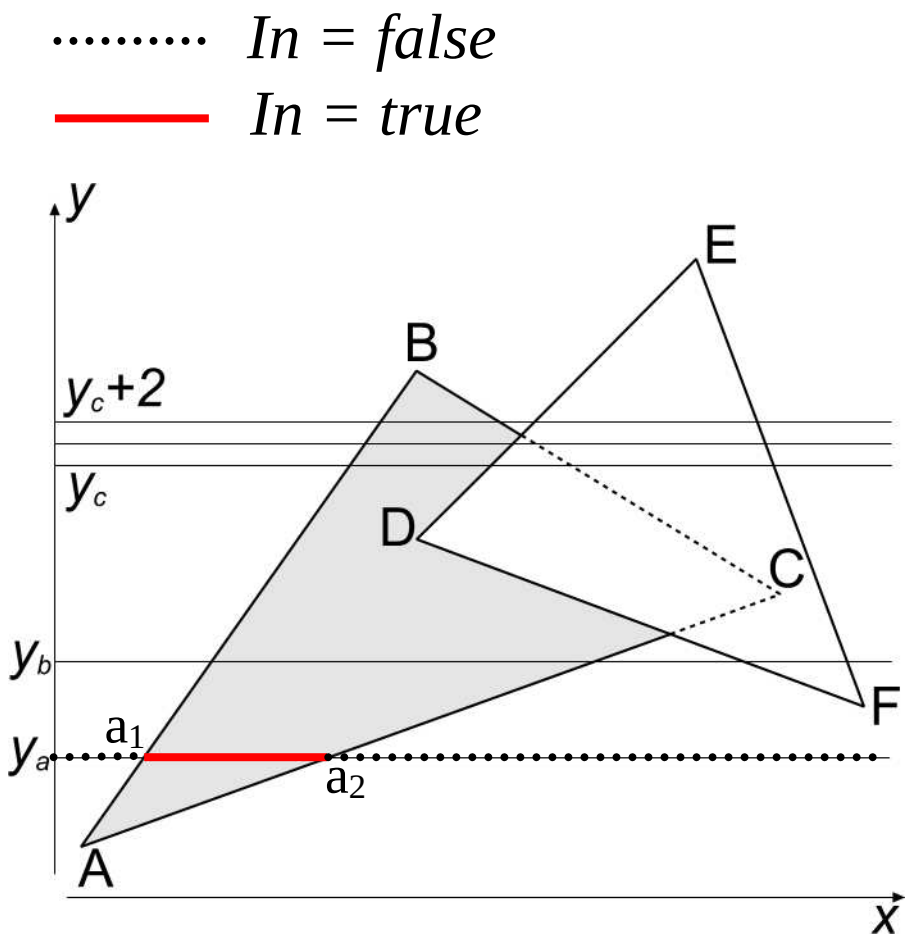


HSR: Algoritmo scan-line

- Approccio image-space (opera in spazio NDC);
- Il problema della visibilità tra oggetti è ricondotto ad un problema di visibilità tra segmenti;
- Tre strutture dati:
  - Tabella degli spigoli (gli estremi di ogni spigolo, il coefficiente angolare della retta di appartenenza e nome della primitiva di appartenenza);
  - Tabella delle primitive (coefficienti dell'equazione del piano, colore, flag *In* che assume valore True quando l'algoritmo è "dentro" la primitiva;
  - Tabella degli spigoli attivi (contiene, istante per istante, gli spigoli "attivi", cioè gli spigoli non orizzontali interessati dalla scan-line corrente. Gli spigoli sono ordinati secondo valori crescenti dell'ascissa di intersezione con la scan-line

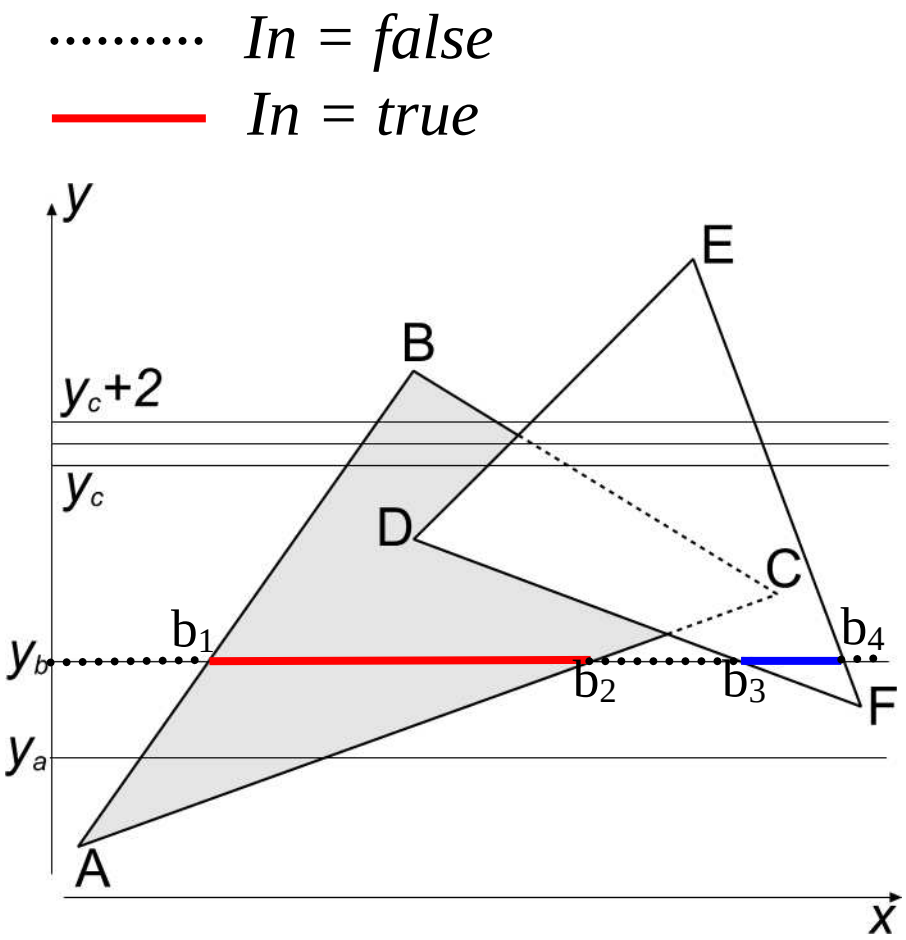
HSR: Algoritmo scan-line

- *Scan-line* $y=y_a$;
- Spigoli attivi: AB, AC;
- *In=true* tra a_1 e a_2 sullo scan line $y=y_a$;
- In questo intervallo è l'unico flag attivo quindi i pixel di ordinata $y=y_a$ e ascissa compresa tra a_1 e a_2 sono visibili.



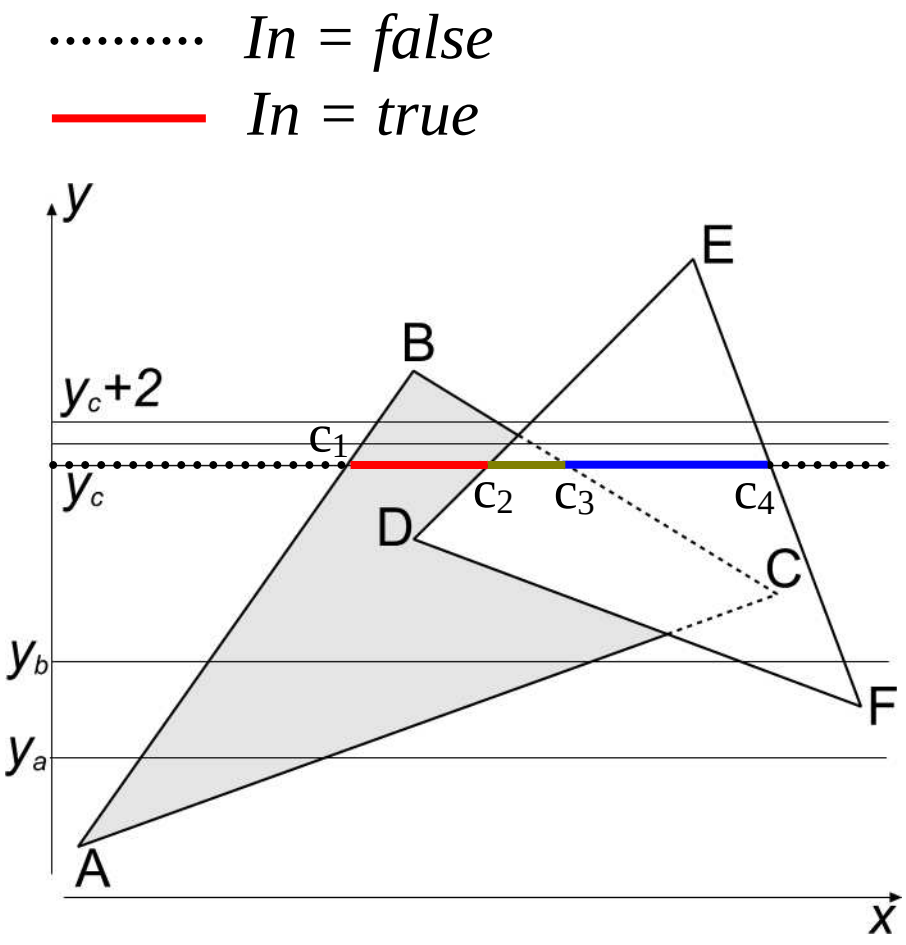
HSR: Algoritmo scan-line

- *Scan-line* $y=y_b$;
- Spigoli attivi: AB, AC, DF ed EF;
- *In=true* tra b_1 e b_2 nella primitiva ABC e tra b_3 e b_4 in DEF;
- Nessun conflitto in questi intervalli; i pixel di ordinata $y=y_a$ e ascissa compresa tra b_1 e b_2, b_3 e b_4 sono visibili.



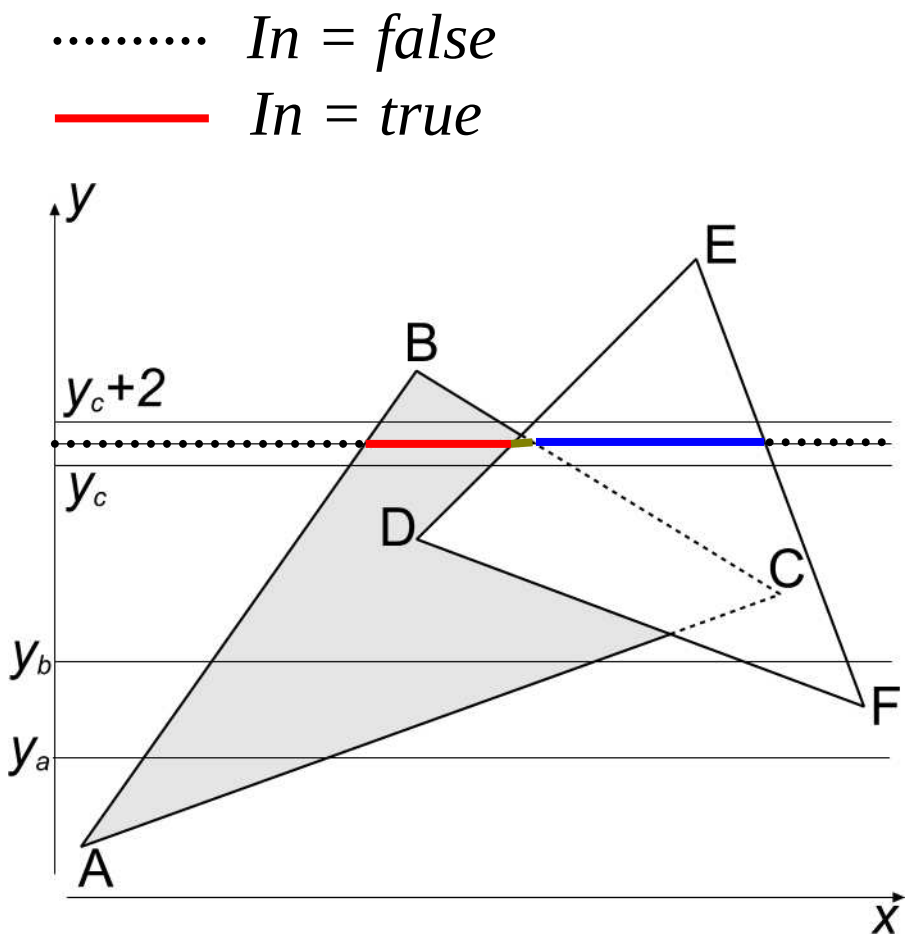
HSR: Algoritmo scan-line

- *Scan-line* $y=y_c$;
- Spigoli attivi: AB, DE, BC ed EF;
- Un solo flag *true* tra c_1 e c_2 e tra c_3 e c_4;
- Tra c_2 e c_3 due flag *In* risultano *true.* Occorre valutare la profondità di entrambe le primitive per decidere chi è visibile. E' sufficiente il test in un solo punto (c_2 nell'esempio) .



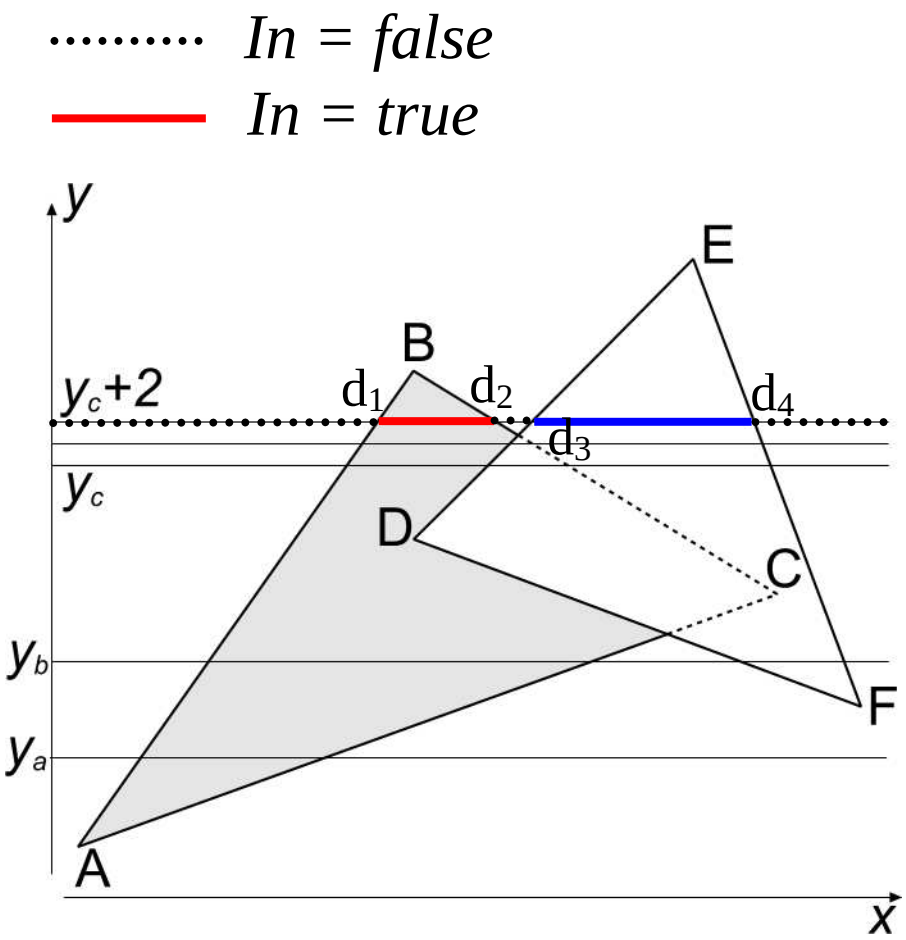
HSR: Algoritmo scan-line

- *Scan-line* $y=y_{c;+1}$
- Spigoli attivi sono gli stessi (e nello stesso ordine) dei precedenti;
- Coerenza di linea: nessun cambiamento rispetto alla scan-line precedente; sufficiente aggiornare le ascisse delle intercette.



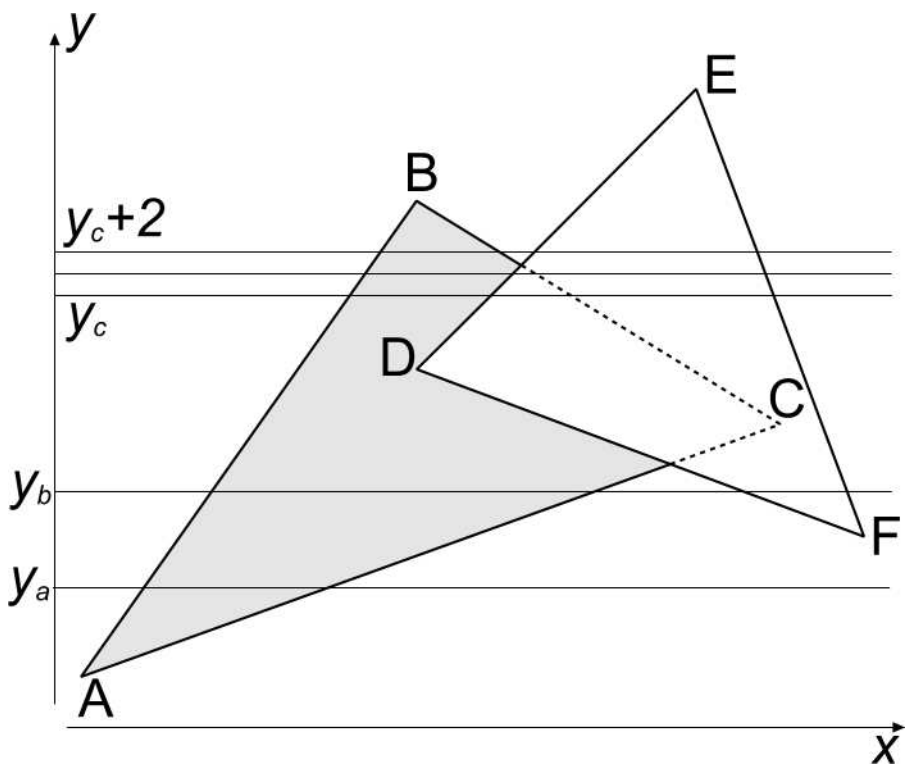
HSR: Algoritmo scan-line

- *Scan-line* $y=y_{c+2;}$
- Gli spigoli attivi (AB, BC, DE ed EF) sono cambiati (nell'ordine) rispetto ai precedenti;
- Non è sfruttabile la coerenza rispetto alla scan-line precedente.



HSR: Algoritmo scan-line

- Rispetto alla tecnica *z-buffer* il numero di controlli sulla profondità è molto inferiore;
- L'algoritmo *z-buffer* non necessita di strutture dati la cui dimensione dipende dalla complessità della scena.

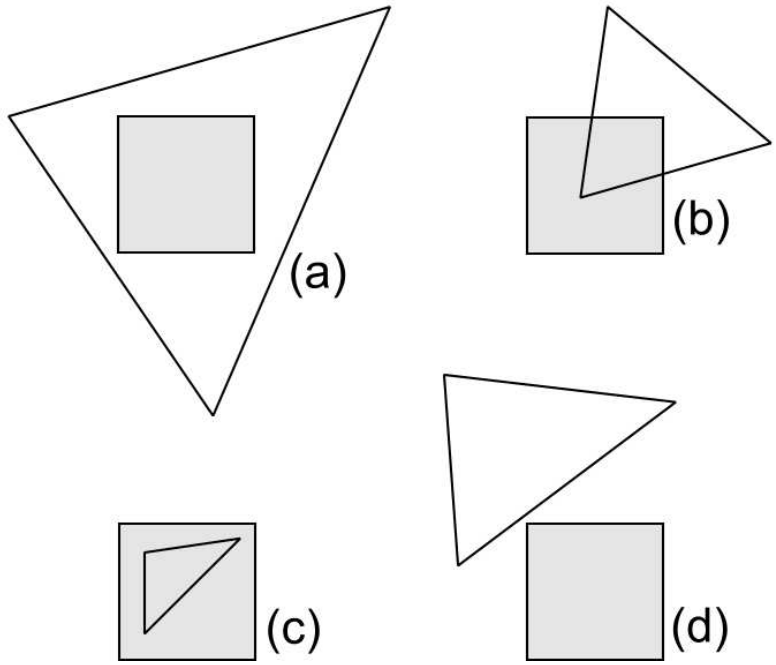


HSR: Algoritmo di Warnock

- Adotta un approccio *image-space* con strategia *divide-et-impera* basata su una tecnica di *suddivisione spaziale;*
- L'algoritmo suddivide il piano immagine in porzioni distinte:
  - Se per la singola porzione del piano immagine risulta facile decidere le primitive visibili, allora per quella porzione si procede alla restituzione;
  - Altrimenti si divide ricorsivamente la porzione in modo da ridurre il numero di primitive che insistono su di essa.

HSR: Algoritmo di Warnock

- L'algoritmo di Warnock assume che il piano immagine sia di forma quadrata e che ogni suddivisione successiva generi 4 quadrati;
- La relazione tra la proiezione di una primitiva ed il generico quadrato può essere:
  - (a) contenente;
  - (b) intersecante;
  - (c) contenuta;
  - (d) disgiunta.



HSR: Algoritmo di Warnock

- Il generico quadrato non necessita ulteriore suddivisione se:
  1. Tutte le primitive sono *disgiunte* rispetto al quadrato. Il quadrato assume il colore del fondo;
  2. Per il quadrato esiste una sola primitiva *intersecante* oppure una sola primitiva contenuta. Il quadrato assume il colore del fondo e quindi si procede al “disegno” della parte della primitiva interna ad esso;
  3. Per il quadrato esiste una sola primitiva *contenente*. Il quadrato assume il colore della primitiva;
  4. Esistono più primitive in relazione *contenente*, *intersecante* o *contenente* ma solo una è *contenente* e più vicina delle altre all’osservatore. Il quadrato assume il colore di questa primitiva.

HSR: Algoritmo di Warnock

- Per la verifica del test 4 (più primitive che insistono sul quadrato) si procede al computo della profondità delle singole primitive nei 4 vertici del quadrato;
- Nelle situazioni di indecidibilità si suddivide il quadrato in quattro quadrati e si reiterano i test precedenti.