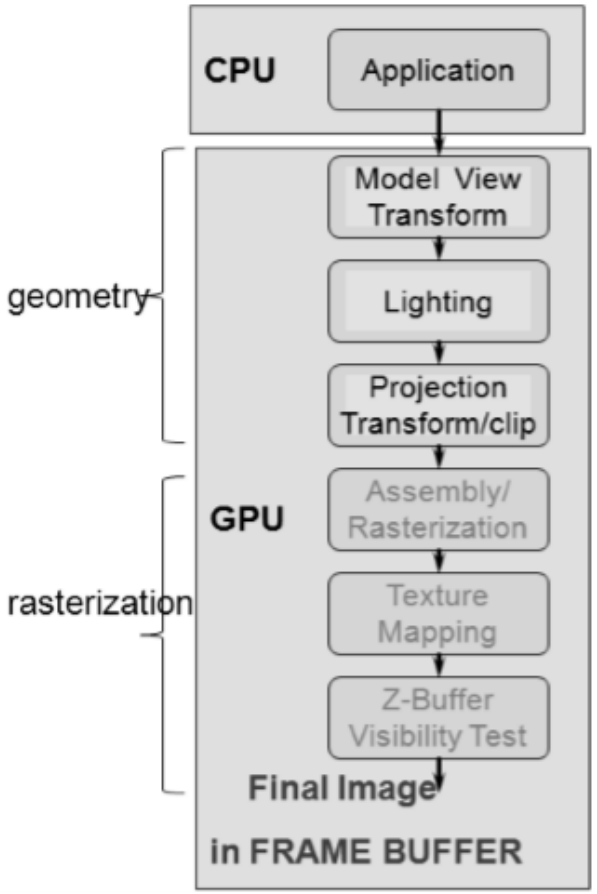
Gli algoritmi per la rimozione delle superfici nascoste (o la determinazione delle superfici visibili) si basano sulle relazioni spaziali tridimensionali tra gli oggetti.
Questo passaggio viene normalmente eseguito come parte dell'elaborazione dei frammenti.



- Sebbene ogni frammento generato dalla rasterizzazione corrisponda a una posizione in un buffer di colore, non vogliamo visualizzare il frammento colorando il pixel corrispondente se il frammento proviene da un oggetto dietro un altro oggetto opaco.
- La rimozione delle superfici nascoste (o determinazione delle superfici visibili) viene eseguita per scoprire quale parte, se presente, di ogni oggetto nel volume della vista è visibile allo spettatore o è oscurato dallo spettatore da altri oggetti.
- Descriviamo una serie di tecniche per una scena composta esclusivamente da poligoni planari. Poiché la maggior parte dei renderer avrà suddiviso le superfici in poligoni a questo punto, questa scelta è appropriata

Riassumendo:

- Il clipping comporta l'eliminazione di oggetti che si trovano al di fuori del volume di visualizzazione e quindi non possono essere visibili nell'immagine.

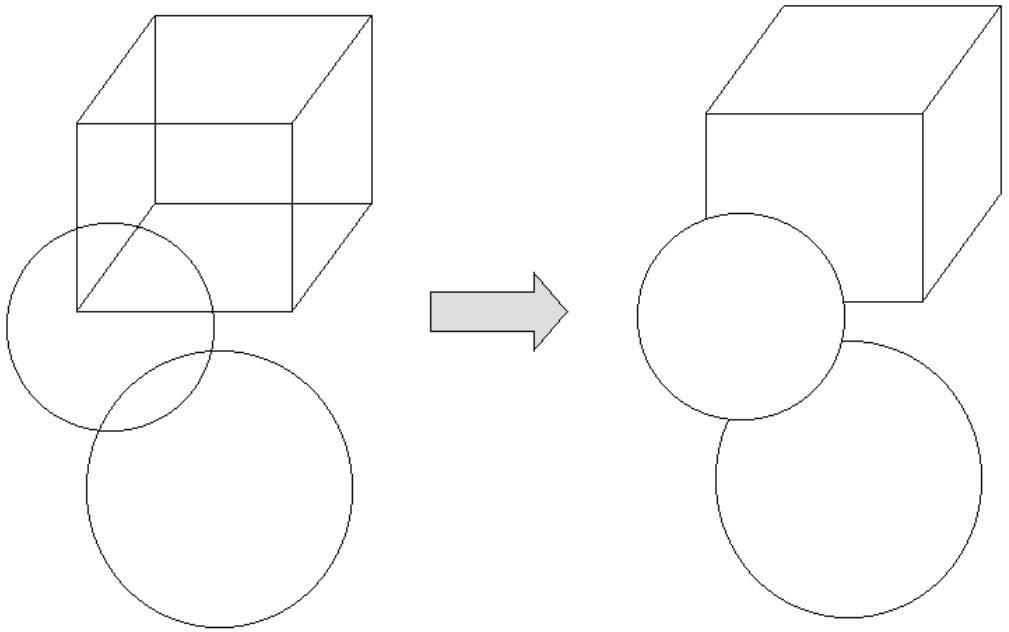
- La rasterizzazione produce frammenti dagli oggetti rimanenti. Questi frammenti possono contribuire all'immagine finale.
- La **rimozione delle superfici nascoste determina quali frammenti corrispondono agli oggetti visibili**, ovvero quelli che si trovano nel volume di visualizzazione e non sono bloccati dalla vista da altri oggetti più vicini alla telecamera.

Eliminazione delle superfici nascoste (Hidden Surfaces Removal) (HSR)

Dato un insieme di oggetti 3D e specificata una vista,

Determinare quali linee o superfici degli oggetti sono visibili, sia dal centro di proiezione (per le proiezioni prospettive) sia lungo la direzione di proiezione (per le proiezioni parallele)

Obiettivo: poter visualizzare solo le linee o le superfici visibili.



Idea semplice.

La sua implementazione richiede una significativa potenza di elaborazione

Richiesta di elevati tempi di elaborazione su macchine convenzionali.

Sviluppo di numerosi ed accurati algoritmi per la determinazione delle superfici visibili e sono state progettate architetture special-purpose per risolvere il problema.

Classificazione degli algoritmi di eliminazione delle superfici nascoste

Image-space : Algoritmi che **lavorano nel sistema di coordinate dello schermo**

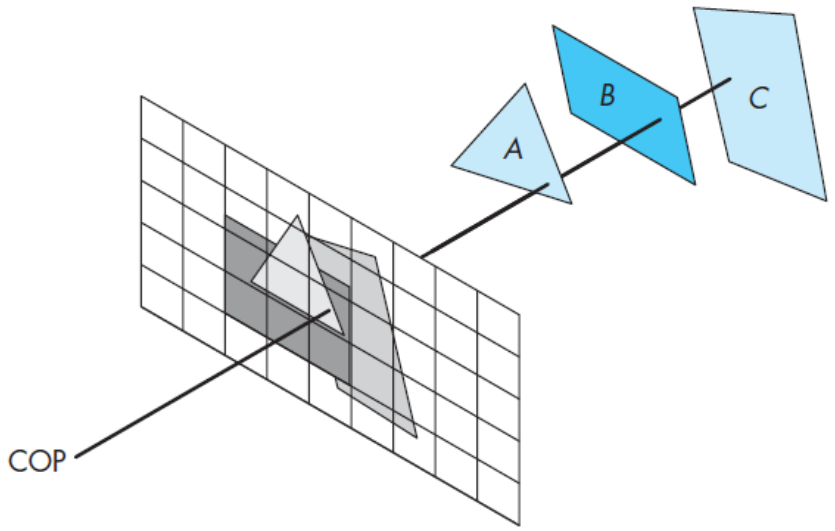
Object-space : Gli algoritmi di tipo object-space **tentano di ordinare le superfici degli oggetti nella scena** in modo tale che il rendering delle superfici in un ordine particolare fornisca l'immagine corretta. Lavorano nel sistema di coordinate di vista.

Image-space

dati *k* oggetti, determina per ognuno degli *mn* pixel nell'immagine, quale dei *k* oggetti è visibile.

•Per ogni pixel,

- si considera un raggio che parte dal centro di proiezione e passa per quel pixel.
- Il raggio viene intersecato con ciascuno dei piani determinati dai *k* poligoni per determinare per quali piani il raggio attraversa un poligono.
- Infine, si determina quale intersezione è più vicina al centro di proiezione,
- si colora il pixel in esame usando la gradazione di colore del poligono nel punto di intersezione.



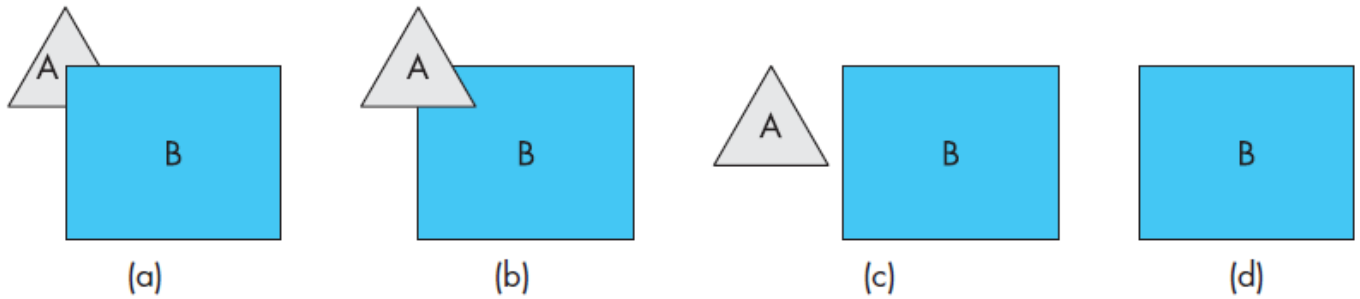
- **L'operazione fondamentale** dell'approccio image-precision è il calcolo delle **intersezioni dei raggi con i k poligoni della scena**.
- Per un display *nm*, questa operazione deve essere eseguita *nmk* volte.
- **La complessità risulta di ordine *O*(*nmk*).** Naturalmente, per aumentare l'accuratezza delle immagini visualizzate, si può considerare anche più di un raggio per pixel.

Object-space:

confronta ciascun oggetto della scena direttamente con ciascun altro eliminando oggetti interi o porzioni di oggetti che non sono visibili.

Data una scena tridimensionale composta da ***k* poligoni piatti ed opachi**, si può derivare un generico algoritmo di tipo object-precision considerando gli oggetti a coppie. Data una coppia di poligoni, ad esempio A e B, ci sono quattro casi da considerare:

- A oscura completamente B dalla fotocamera: visualizzeremo solo A;
- B oscura A: visualizzeremo solo B;
- A e B sono completamente visibili: visualizzeremo sia A che B;
- A e B si oscurano parzialmente l'un l'altro: dobbiamo calcolare le parti visibili di ciascun poligono.



•(a) B oscura parzialmente A; (b) A oscura parzialmente B; (c) A e B sono completamente visibili; (d) B oscura totalmente A.

Complessità computazionale

Per parlare di complessità, consideriamo come **singola operazione** tutto il calcolo richiesto per determinare dati due oggetti in che relazione stanno (caso (a), (b), (c) oppure d).
Si procede quindi induttivamente:

Al generico passo confrontare l'i-esima primitiva $(i = 1, \dots, k-1)$, con le rimanenti $k - i$, in modo da individuare le parti visibili.

La complessità di questo calcolo è $O(k^2)$

L'approccio object-precision funziona meglio per scene che contengono relativamente pochi poligoni.

Sebbene questo approccio sembra migliore dell'approccio image precision per $k < mn$, i suoi passi sono più complessi e richiedono più tempo di elaborazione, ed è spesso più lento e più difficoltoso da implementare.

•Gli algoritmi per la eliminazione di superfici nascoste, sia nella forma image-space che object-space richiedono un elevato costo computazionale.

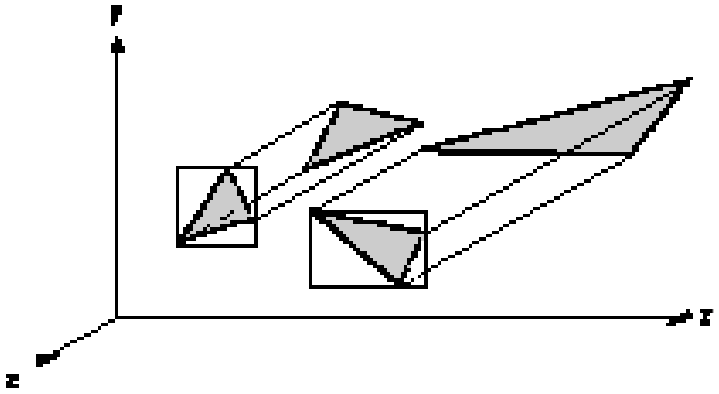
• Tecniche da utilizzare prima dell'eliminazione delle superfici nascoste, per ridimensionarne il costo computazionale.

- Extents box e bounding volume
- Back Face Culling
- Partizionamento spaziale

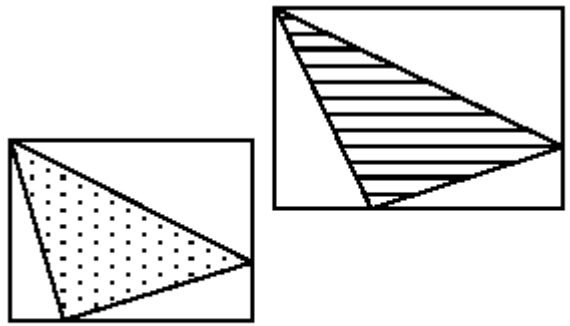
Extents e Bounding Volume

- Extent (estensione) di un oggetto, in una dimensione: intervallo tra il valore massimo ed i valore minimo dell'oggetto in quella dimensione.
- Extent di un oggetto sullo schermo: è il più piccolo rettangolo allineato con gli assi che contiene l'oggetto.

- Consideriamo due oggetti 3D e gli extent rettangolari delle loro proiezioni.
- Supponiamo di aver considerato la proiezione ortografica, per cui la proiezione sul piano (x,y) (z=0) si ottiene semplicemente ignorando la componente z nelle coordinate.

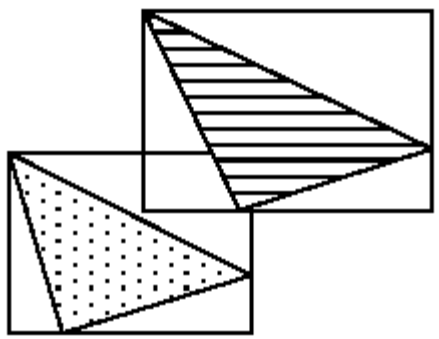


- Se gli extent non si sovrappongono, le proiezioni non dovranno essere testate.

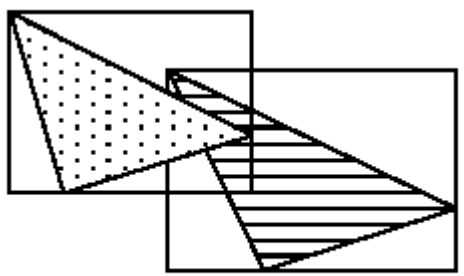


Se gli extent si sovrappongono, si possono verificare due casi:

Le loro proiezioni non si sovrappongono



Le loro proiezioni si sovrappongono

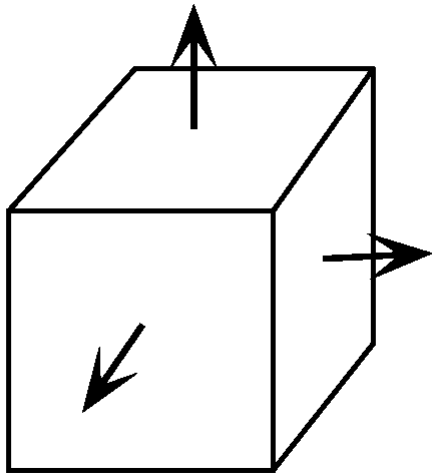


In entrambi i casi, bisogna realizzare i confronti per determinare se le proiezioni di sovrappongono.
In un caso, i confronti stabiliranno che le due proiezioni non si intersecano. La sovrapposizione degli extent è un falso allarme.

- Il test dell’extent rettangolare è detto anche test bounding box.
- Gli extent possono essere anche utilizzati per circoscrivere gli oggetti, in questo caso sono detti bounding volume.

Back face culling

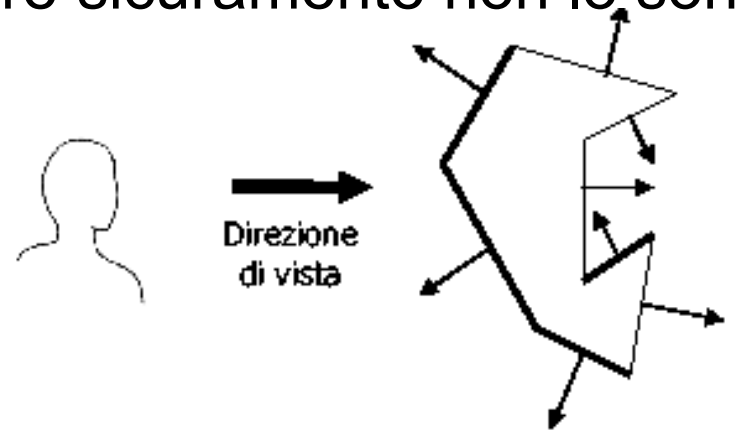
- Per oggetti solidi poliedrici:
- Definiamo le normali alle superfici per ogni faccia del poliedro (che punta fuori dal poliedro)



•Consideriamo un oggetto rappresentato da un poliedro solido chiuso, le facce poligonali del poliedro delimitano completamente il solido.

•Supponendo di aver definito i poligoni in maniera tale che le normali alle loro superfici siano tutte dirette verso l'esterno, cioè i vertici del poligono sono percorsi in senso antiorario.

•Le facce che hanno una normale che punta verso l'osservatore possono essere visibili, quelle con la normale che punta dall'altra parte rispetto all'osservatore sicuramente non lo sono.



- E’ opportuno eliminare prima tutti i poligoni il cui vettore normale è orientato verso il semipiano opposto all’osservatore.

- Questa operazione prende il nome di **eliminazione delle back-faces o back face culling.**

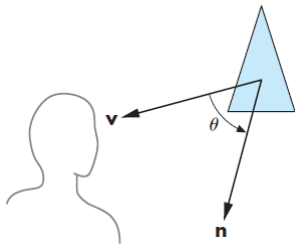
•Per determinare se un poligono deve essere eliminato, bisogna verificare se la sua normale è diretta verso l’osservatore.

•Se indichiamo con θ l’angolo tra la normale e l’osservatore, il poligono in esame definisce la parte anteriore di un oggetto se e solo se:

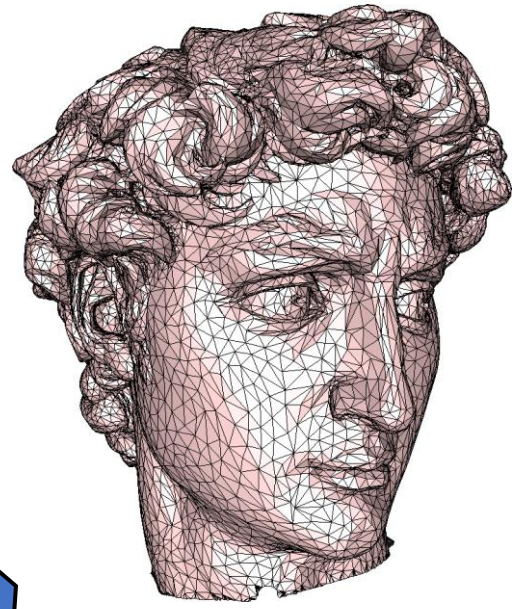
$$\bullet -90 \leq \theta \leq 90$$

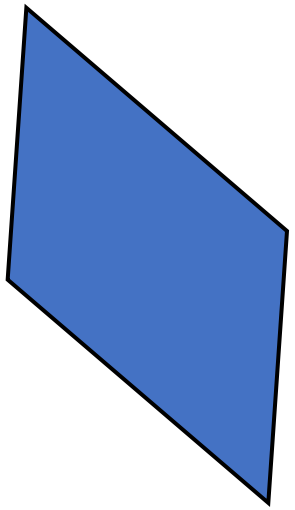
O equivalentemente $\cos \theta \geq 0$.

Equivalente a $n \bullet v \geq 0$



Backface Culling

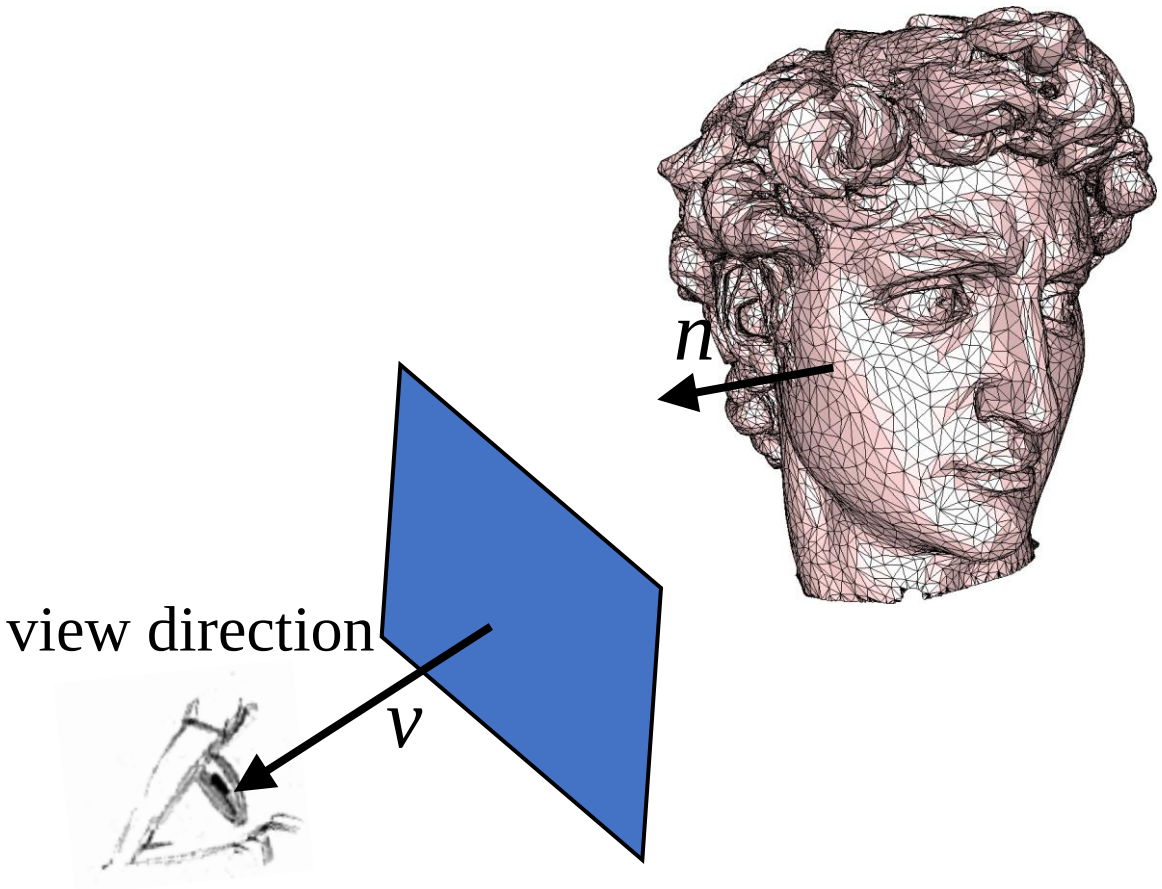




Eventuali triangoli rivolti all'indietro dovrebbero essere eliminati il prima possibile, poiché ci si aspetta che fino al 50% dei triangoli in una scena ha la normale che punta nel semipiano opposto alla direzione di vista

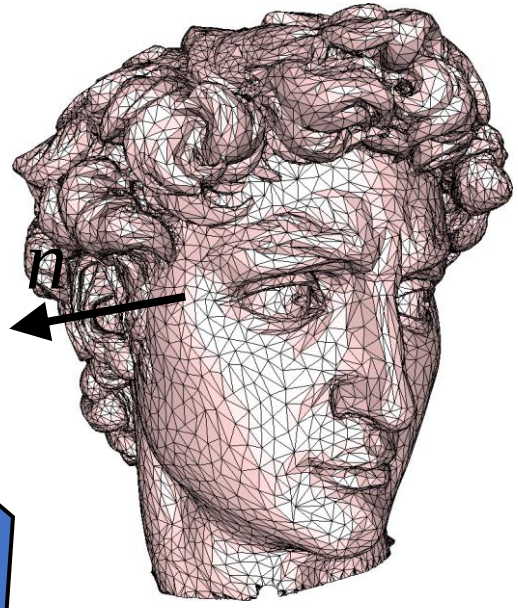
Di solito, l'eliminazione delle facce posteriori viene eseguito prima del clipping, poiché è un'operazione molto rapida e influenzerà una percentuale di triangoli molto maggiore rispetto al clipping.

Backface Culling

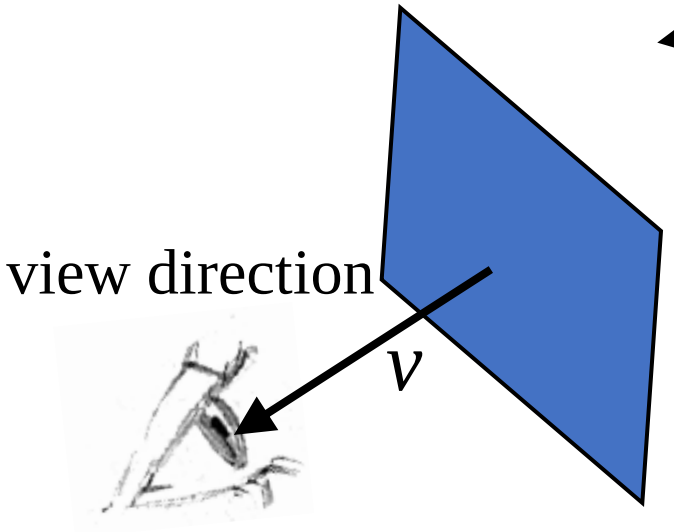


Backface Culling

$n \cdot v = \cos(\theta) \geq 0$, il poligono viene disegnato

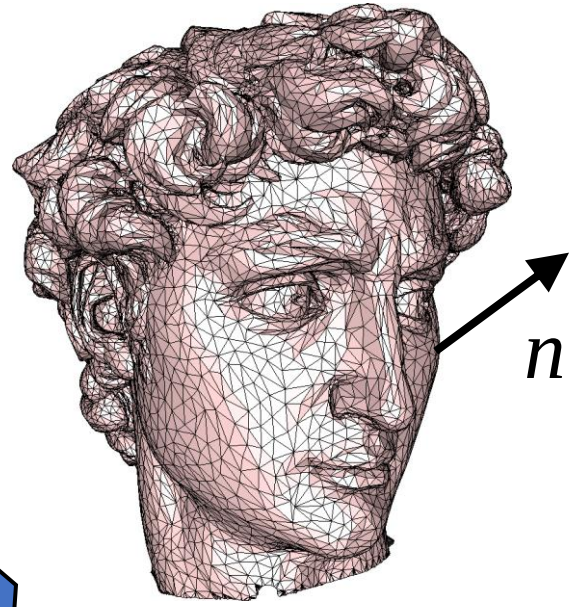


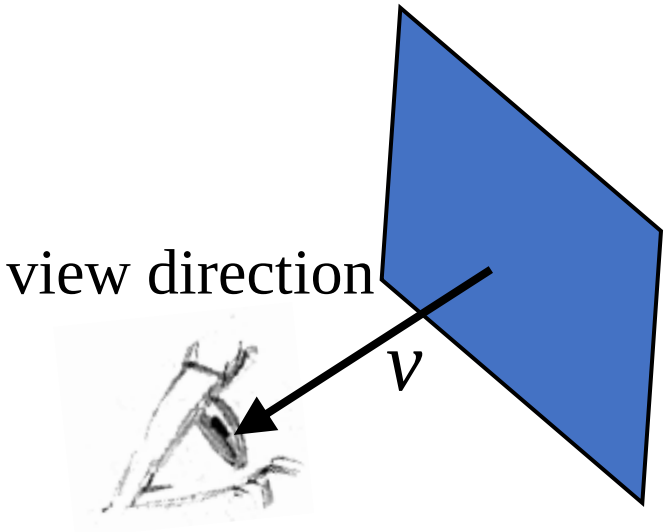
$-90 \leq \theta \leq 90$



Backface Culling

$n \cdot v < 0$, il poligono non viene visualizzato





Partizionamento Spaziale

Idea: dividere lo spazio ed assegnare ad ogni sottospazio una parte degli oggetti.

Vantaggi: confronti tra oggetti molto ridotti.

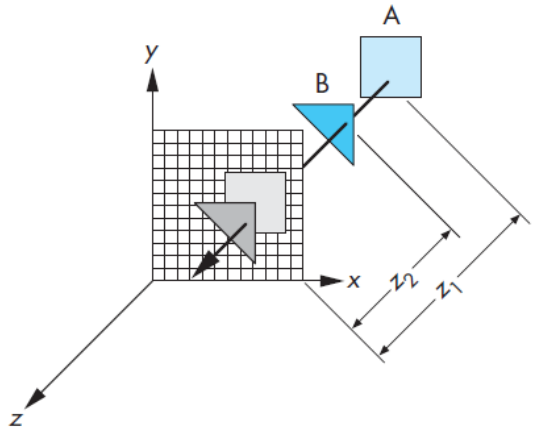
Problema: non sempre possibile dividere bene lo spazio.

Per ovviare a questo problema si usano tecniche di partizionamento adattivo. Vedremo poi degli esempi.

Algoritmi per la determinazione delle superfici visibili

Algoritmo Z-Buffer o depth-buffer

- Algoritmo image-precision
- E’ uno dei più semplici algoritmi per la determinazione delle superfici visibili da implementare sia in software che in hardware.



Supponiamo di dover rasterizzare uno dei due poligoni mostrati in figura.
Tracciamo il raggio dal centro di proiezione e un pixel.
Se **stiamo rasterizzando B**, **il suo colore apparirà sullo schermo se la distanza z2 è inferiore alla distanza z1 del poligono A**.
Al contrario, se stiamo rasterizzando A, il pixel che corrisponde al punto di intersezione non apparirà su il display.

Supponiamo di avere un buffer, **detto z- buffer o depth buffer**, con la stessa risoluzione del frame buffer e con profondità coerente con la risoluzione che desideriamo utilizzare per la distanza. Ad esempio, se abbiamo un display 1024 × 1280 e utilizziamo numeri interi standard per il calcolo della profondità, possiamo usare un buffer z 1024 × 1280 con elementi a 32 bit.

Pseudo codice

Durante il processo di proiezione per ogni vertice proiettato viene conservata la sua distanza z dal centro di proiezione (distanza dall'osservatore)

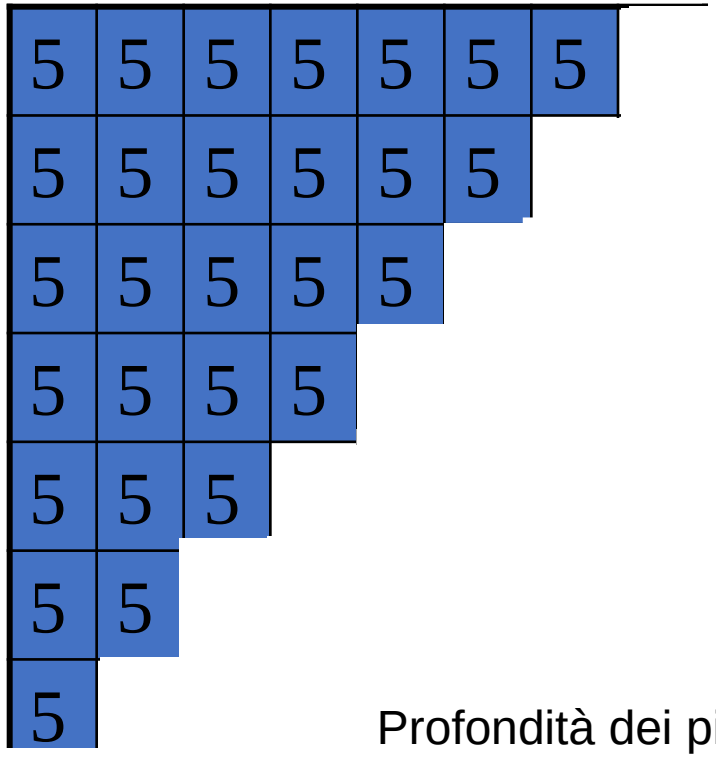
- For (ogni poligono)
  - For (ogni pixel del poligono proiettato)
    - z= distanza dall'osservatore del poligono nel punto (x,y) del frame buffer.
    - If (z >=valore di z memorizzato nel dept-buffer in posizione (x,y))
      - Disegna il colore del pixel del poligono nella posizione (x,y) del frame buffer.
      - Scrivi il valore di z nella posizione (x,y) del depth-buffer.
    - End for
  - End for

Lo z-buffer ed il frame buffer registrano l'informazione associata con la più grande z incontrata per ogni (x,y).
I poligoni appaiono sullo schermo nell'ordine in cui vengono processati.

Esempio del funzionamento dell'Algoritmo Z-buffer

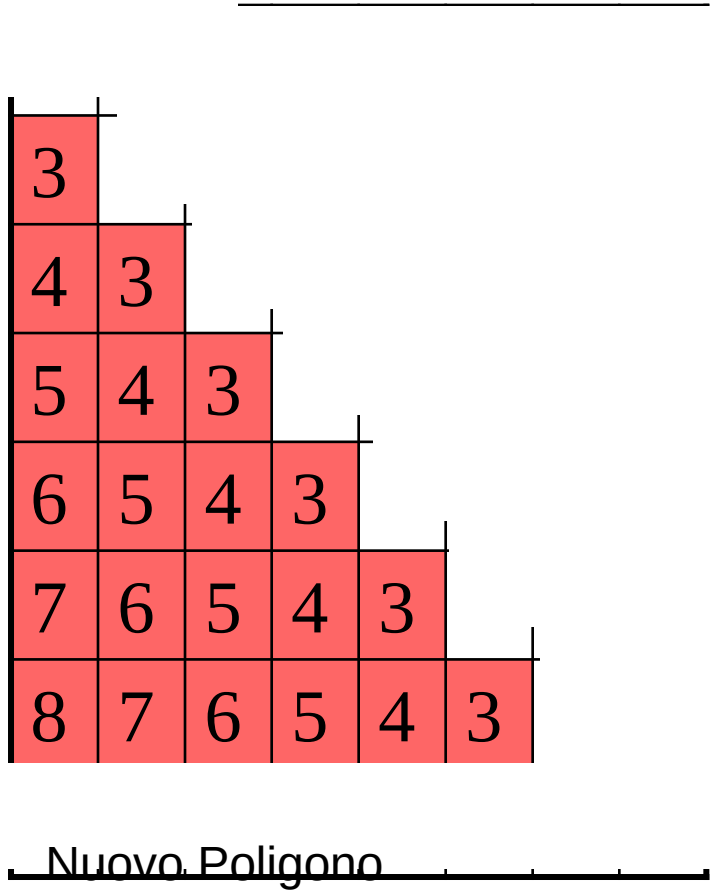
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

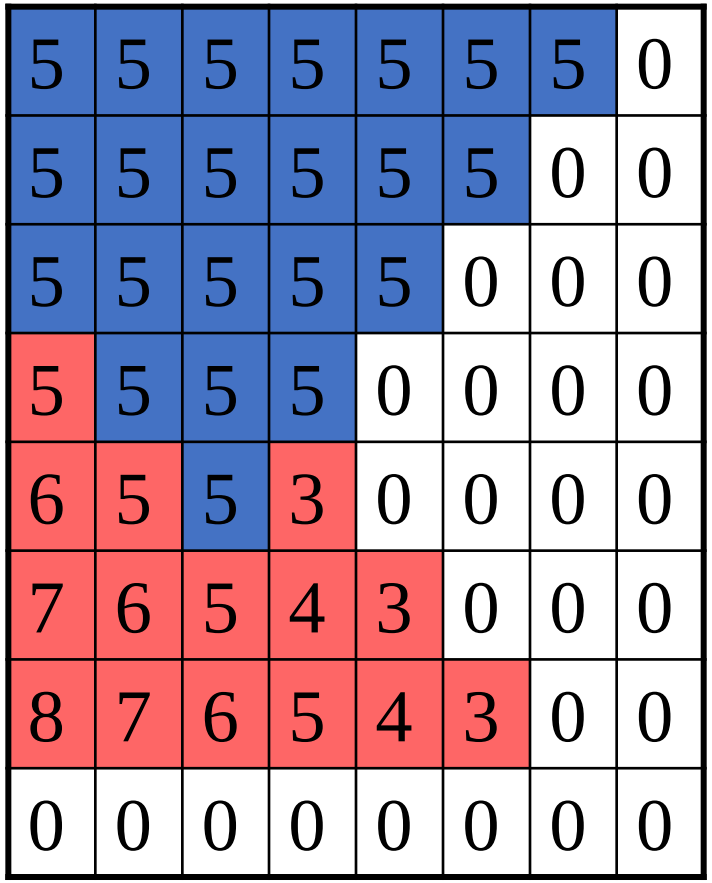
Inizializzazione dello z-buffer



5	5	5	5	5	5	5	0
5	5	5	5	5	5	0	0
5	5	5	5	5	0	0	0
5	5	5	5	0	0	0	0
5	5	5	0	0	0	0	0
5	5	0	0	0	0	0	0
5	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Frame buffer aggiornato

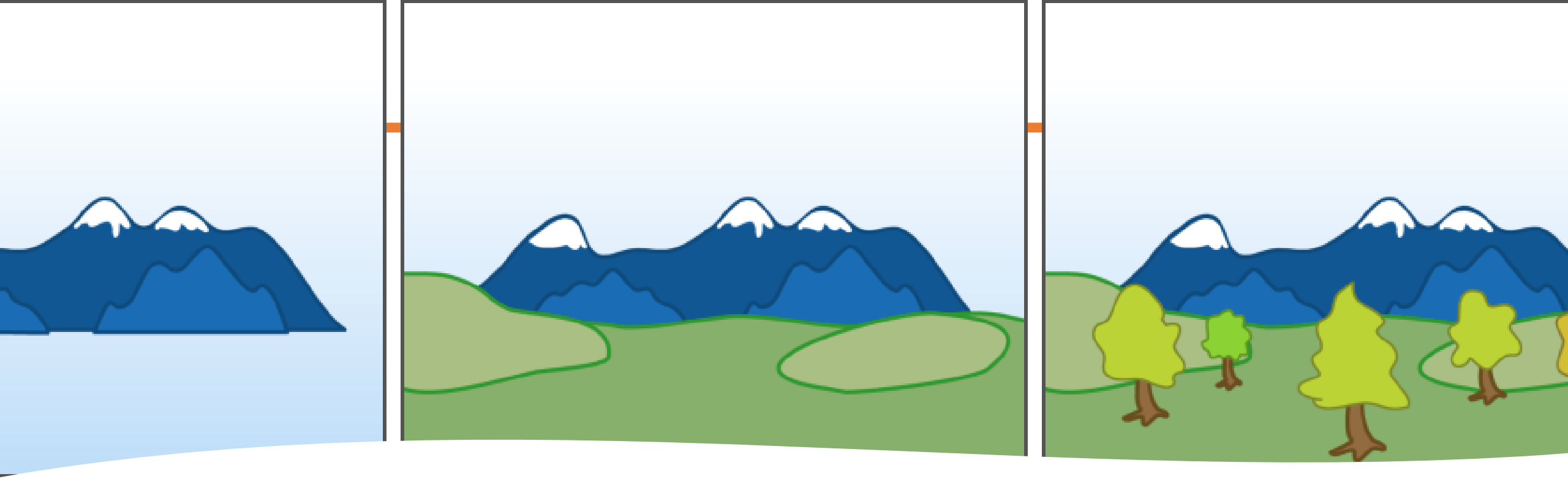




Depth buffer dopo la scan-conversion del secondo poligono

- L’algoritmo non richiede che gli oggetti siano poligoni. Può esser utilizzato per qualsiasi oggetto purchè possa essere determinato il valore di z per ogni suo punto nella sua proiezione.
- Non richiede che gli oggetti siano ordinati.
- Non richiede confronti tra gli oggetti.

- Fissato un pixel, l'ordinamento delle z avviene solo tra quei poligoni che contengono quel pixel.
- Il tempo speso per i calcoli per la determinazione delle superfici visibili tende ad essere indipendente dal numero dei poligoni, perché in media, il numero dei pixel coperti da ogni poligono decresce quando il numero dei poligoni nel volume di vista cresce.



Algoritmi che si basano sulla lista di priorità : Depth Sort

Definiscono un **ordine di visibilità** per gli oggetti ed assicurano che se gli oggetti verranno renderizzati in quell'ordine, la visualizzazione sarà corretta.

Lista di priorità:

- priorità più bassa oggetti più lontani dall'osservatore
- priorità più alta oggetti più vicini all'osservatore

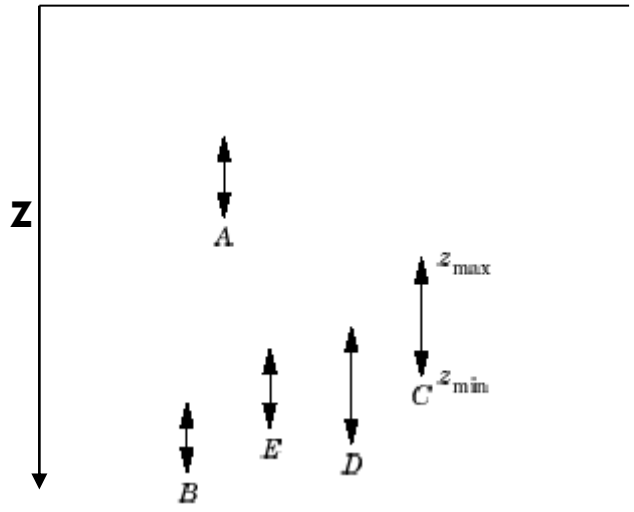
Dopo che è stata determinata la priorità, i poligoni vengono sottoposti a scan-conversion uno alla volta all'interno del frame buffer, iniziando dal poligono a priorità più bassa.

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.

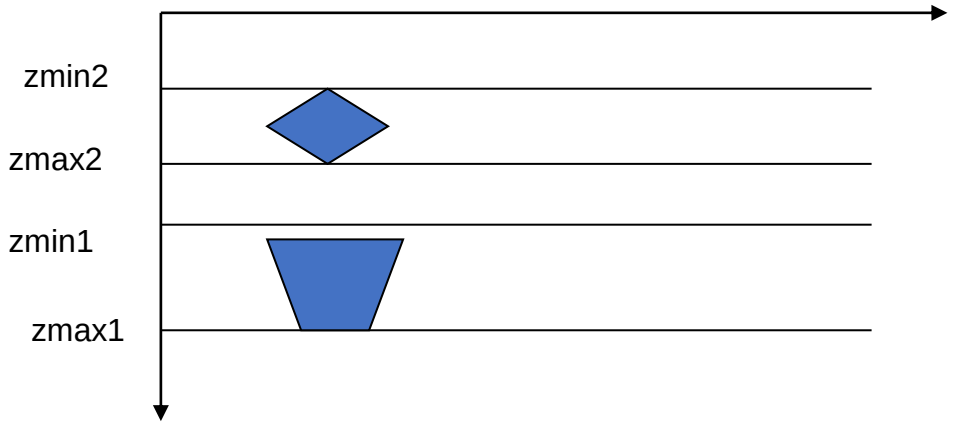
Gli elementi più lontani sono progressivamente oscurati da quelli più vicini all'osservatore.

- Se dopo l’ordinamento nella lista non compaiono neppure due poligoni adiacenti che si sovrappongono nel senso della profondità, la lista si trova ordinata correttamente.

Se la profondità minima di ogni poligono è maggiore della profondità massima del poligono situato sul retro, possiamo visualizzare i poligoni partendo da quello più in profondità, con priorità più bassa



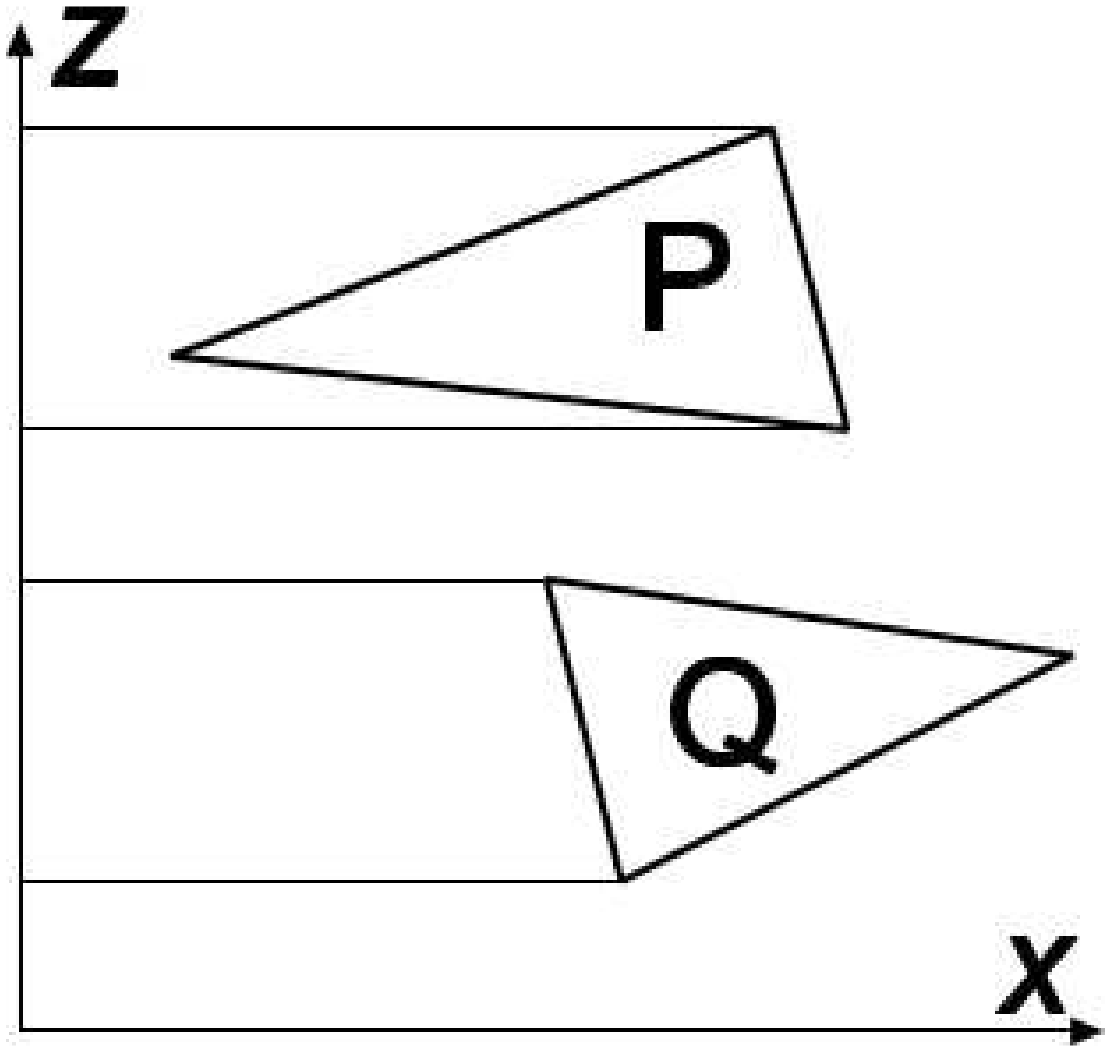
- Gli oggetti più lontani saranno oscurati da quelli più vicini come i pixel dei poligoni più vicini sovrascrivono quelli dei pixel più lontani.



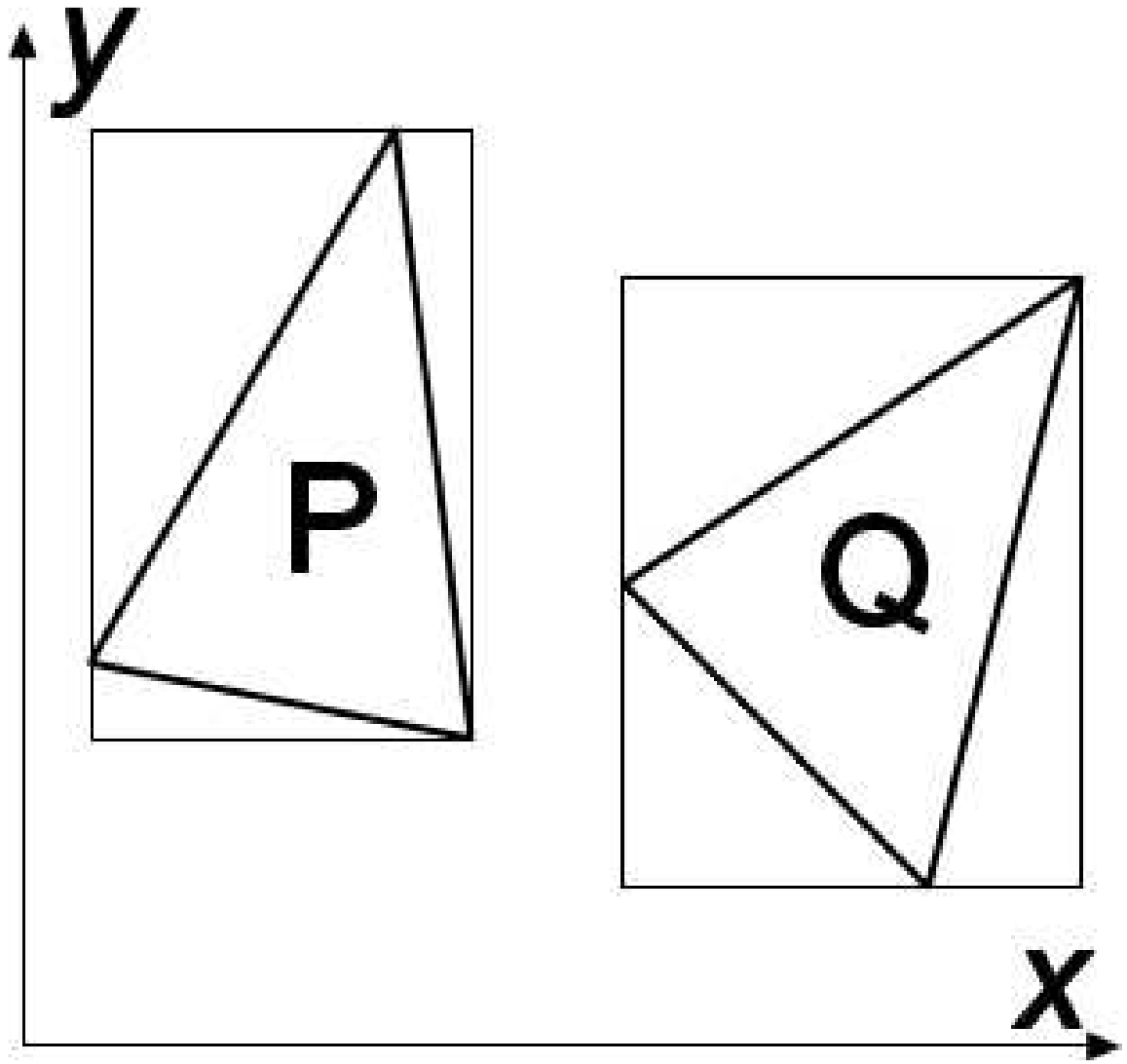
▶priorità più bassa: oggetti più lontani dall'osservatore

Occorre una strategia che permetta di risolvere i problemi legati alle eventuali sovrapposizioni in profondità delle primitive geometriche.

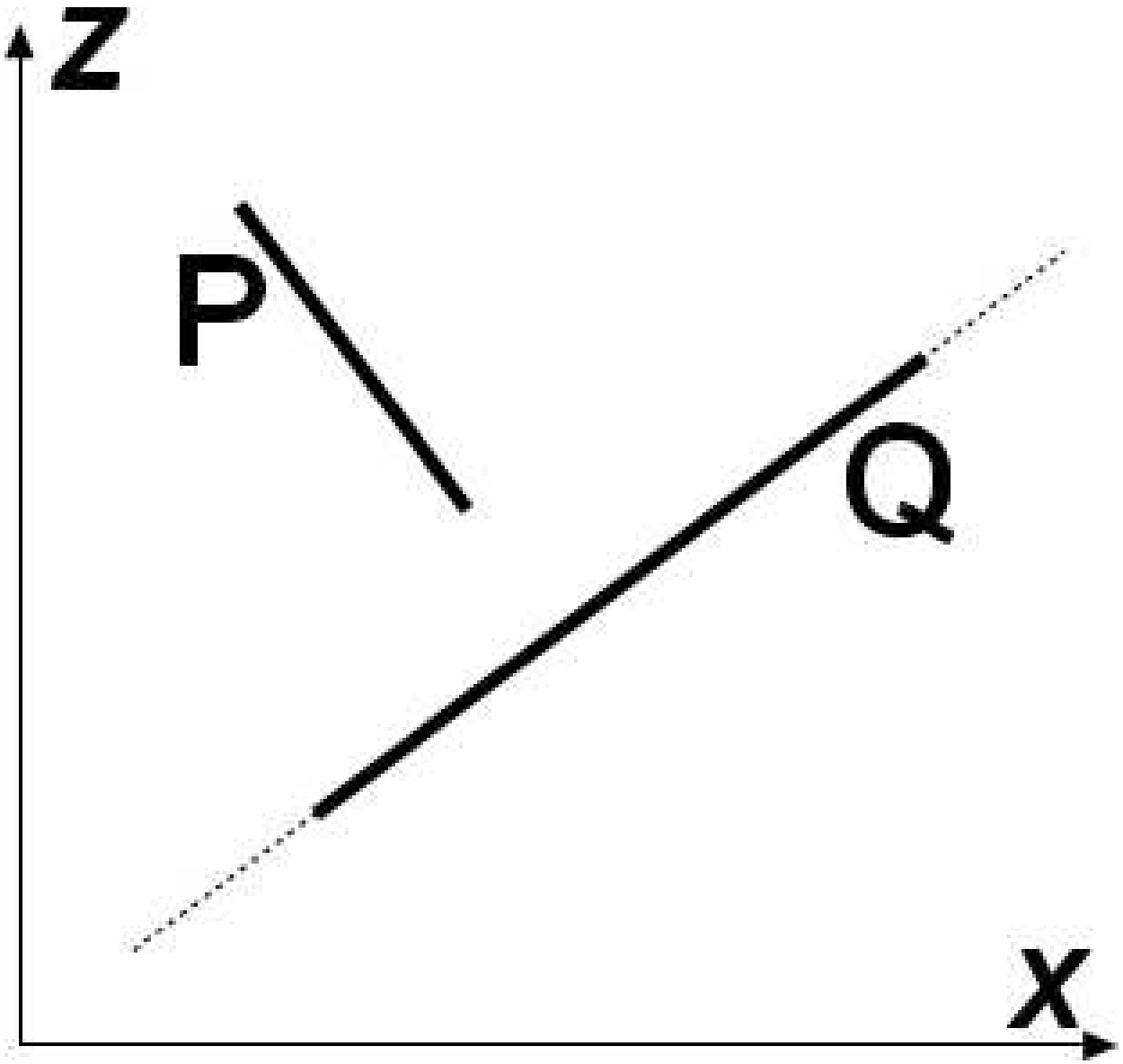
Se gli extent della primitiva a profondità maggiore (P) non si sovrappongono agli extent della primitiva che segue (Q), allora P precede Q



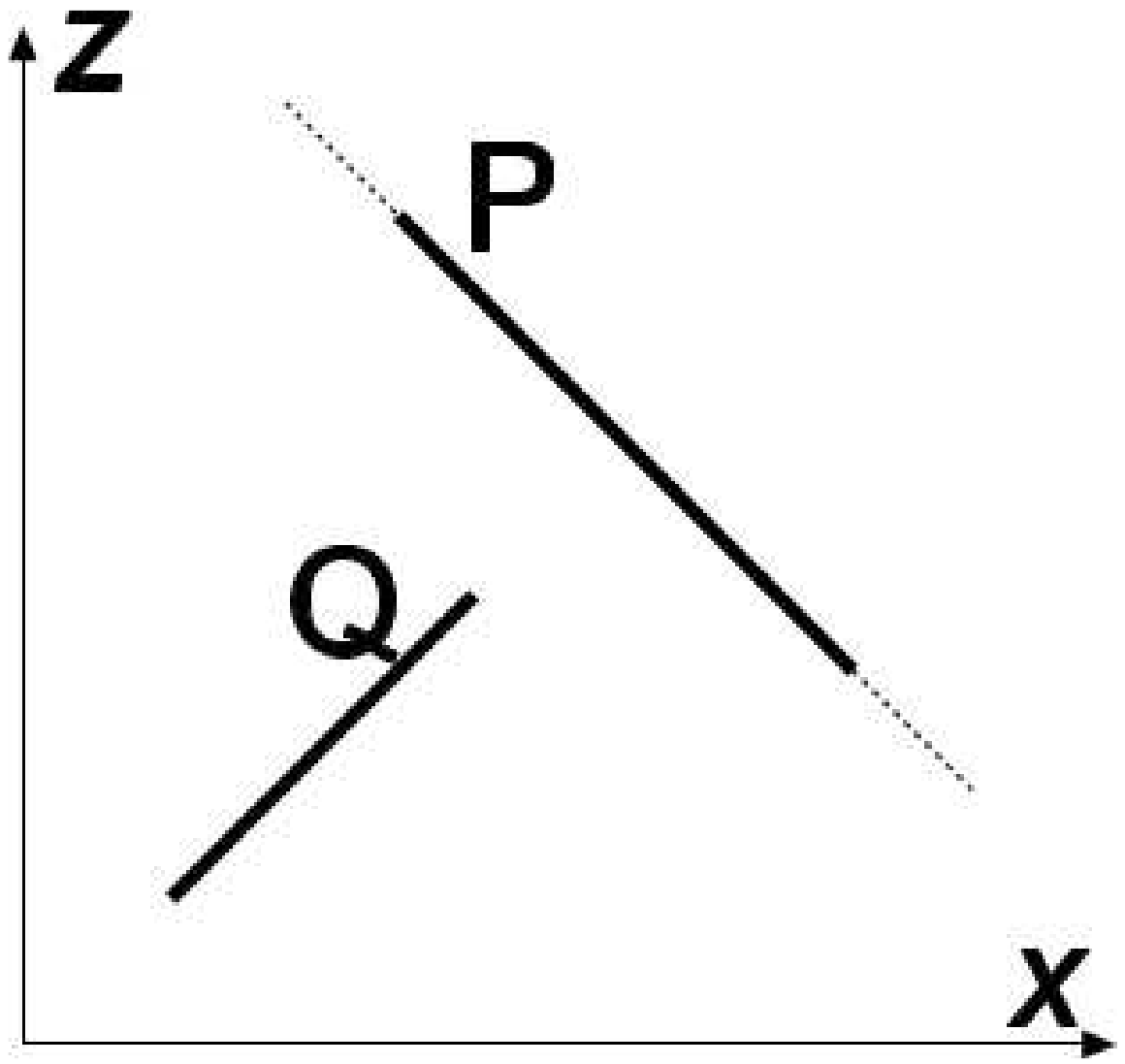
- Altrimenti, se i bounding box di P e Q sul piano xy non interferiscono, allora P precede Q;



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

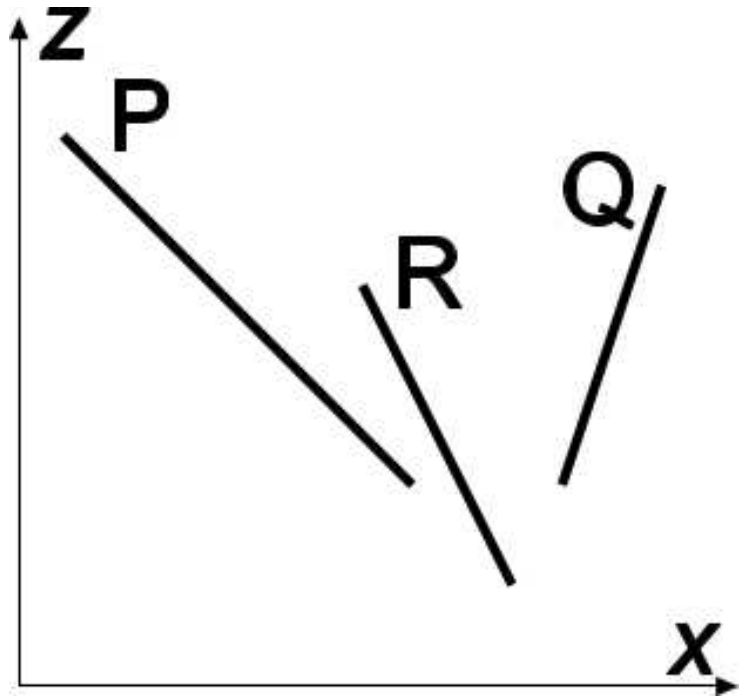


- Altrimenti, se tutti i vertici di Q si trovano dalla stessa parte dell'osservatore rispetto al piano individuato da P, allora P precede Q

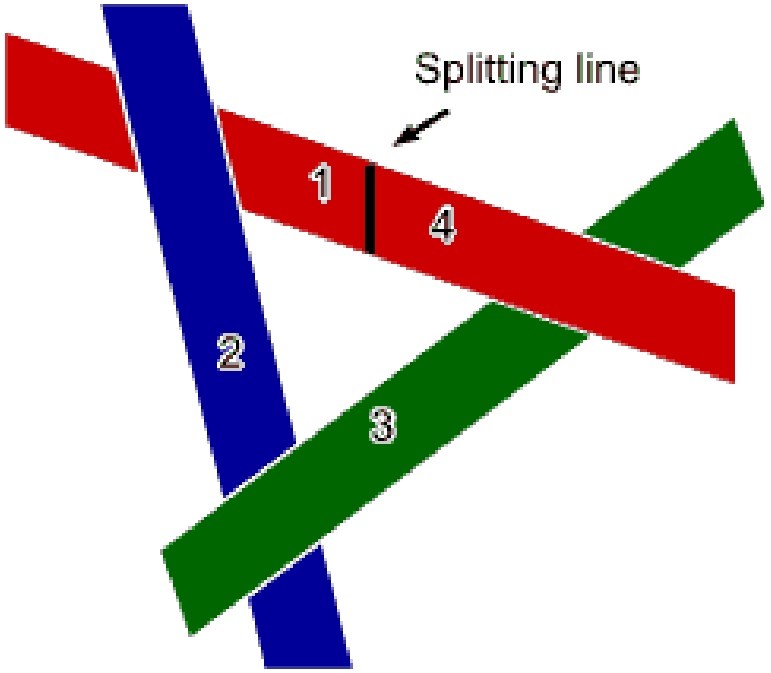


- Se tutti i test precedenti forniscono esito negativo allora si procede allo scambio di P con Q nell'ordinamento e, nuovamente, all'esecuzione dei test;

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



Caso di oggetti con sovrapposizione ciclica

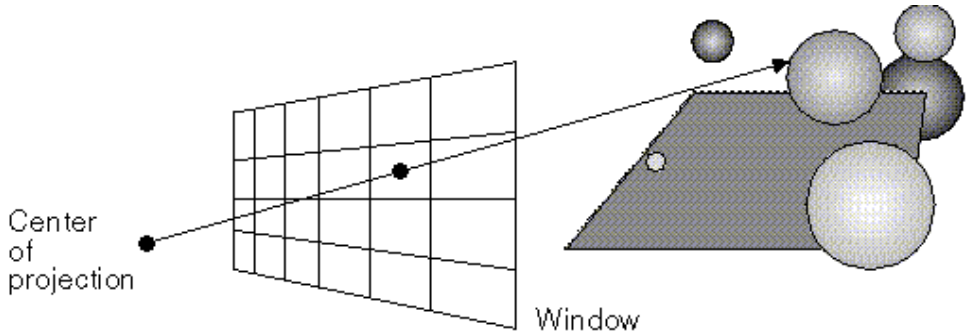


- L’algoritmo di depth-sorting è di tipo ibrido:
  - Object precision nella fase relativa al confronto delle profondità.
  - Image precision nella fase di scan-conversion dei poligoni nel frame buffer.

Ray Casting

E' un meccanismo image-precision che consente il rilevamento delle superfici visibili

.



L'intero processo fa riferimento ad un centro di proiezione e ad una window posizionata su un view plane arbitrario e pensata come una griglia regolare, i cui elementi corrispondono alla dimensione dei pixel della risoluzione desiderata.

- Vengono tracciati raggi dal centro di proiezione (occhio dell’osservatore) verso gli oggetti presenti nella scena, uno per il centro di ogni pixel della scena.
  Il colore di ogni pixel e' settato a quello dell'oggetto piu' vicino in corrispondenza del punto di intersezione tra il raggio e gli oggetti trovati lungo il percorso dello stesso raggio.

Pseudocodice

- Seleziona il centro di proiezione e la finestra sul piano di vista.
- For (ogni scan line nell'immagine)
  - For (ogni pixel nella scan-line)
    - Determina il raggio dal centro di proiezione che passa attraverso il pixel;
      - For (ogni oggetto nella scena)
        - Se l'oggetto è intersecato dal raggio ed è più vicino all'osservatore registra l'intersezione e il nome dell'oggetto

Definisce il colore del pixel come quello dell'intersezione con l'oggetto più vicino

Calcolo delle intersezioni

- Il cuore di un algoritmo di ray-casting è la determinazione dell'intersezione di un raggio con un oggetto.
- Per gli oggetti della scena è possibile scrivere delle routine di intersezioni con il raggio.
- Gli oggetti della scena possono essere costituiti da un insieme di poligoni planari, volumi poliedrici o volumi definiti da superfici parametriche quadratiche o polinomiali.

- Dal momento che un algoritmo di ray-casting impiega il 75-95% dei suoi calcoli per la individuazione delle intersezioni, l'efficienza della routine di intersezione pesa sull'efficienza dell'algoritmo.
- Individuare l'intersezione di una linea arbitraria nello spazio con un particolare oggetto può essere dispendioso.

Efficienza del ray-casting

- Un grosso problema relativo al ray-casting e' dovuto all'efficienza. Infatti, anche la versione piu' semplice richiede di **intersecare ogni raggio con ogni oggetto nell'ambiente**.
- Questo processo puo' diventare molto oneroso nel caso in cui gli oggetti siano numerosi.
- L'unica cosa possibile per migliorarne l'efficienza e' cercare di velocizzare i calcoli d'intersezione o cercare di evitarli in assoluto quando possibile.

- Un’immagine 1024x1024 con 100 oggetti richiede 100M di calcoli per le intersezioni.
- Dal 75 al 95% del tempo del ray-casting è speso nel calcolo delle intersezioni.

Ottimizzazione dei calcoli di intersezione

- **Ottimizzazione delle intersezioni**
- I bounding volumes, diminuiscono l'ammontare del tempo speso nei calcoli di intersezione.
- Un oggetto, che è particolarmente difficoltoso da testare per le intersezioni, può essere racchiuso in un bounding volume, per il quale il test dell'intersezione è meno costosa.
- I bounding volumes piu' comuni oltre alle sfere, ai rettangoli solidi e agli ellissoidi.
- L'oggetto non deve essere testato se il raggio fallisce l'intersezione con il bounding volume che lo contiene.

- Si considera un bounding volume che circoscrive l'oggetto.
- Si esamina l'intersezione del raggio con il volume che circoscrive l'oggetto.
- Se il raggio non interseca questo volume, l'oggetto non viene considerato ulteriormente.
- Come bounding volume si possono usare un parallelepipedo o una sfera.

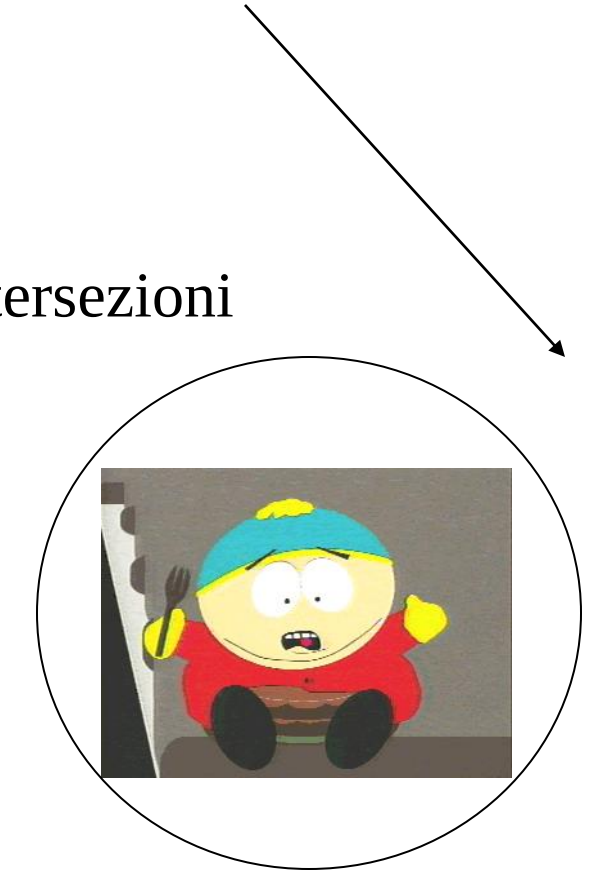
- Determinare se un raggio interseca una sfera è molto semplice.
- Se la distanza tra il centro della sfera ed il raggio è maggiore del raggio della sfera, allora non c'è intersezione: il raggio non interseca l'oggetto inscritto nella sfera.
- Il test della sfera circoscritta si riduce alla determinazione della distanza di un punto da una retta.

Bounding Volume

Cartman contiene 100,000 poligoni

Intersezione Raggio-Cartman = 100,000 intersezioni raggio-poligono

Anche se il raggio non interseca Cartman



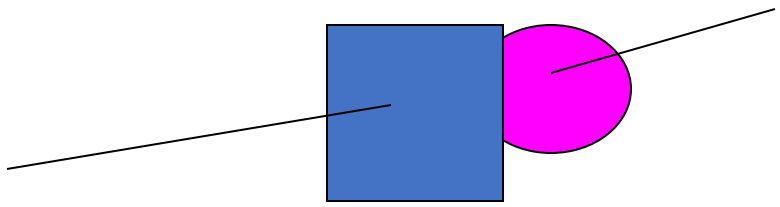
Soluzione

Circoscrivere Cartman con una sfera

Se il raggio non interseca la sfera, allora il Raggio non interseca Cartman

Evitare i calcoli di intersezione

- Ogni raggio dovrebbe essere testato per l'intersezione solo con gli oggetti che interseca.
- Inoltre sarebbe bello che ogni raggio fosse testato solo con quell'oggetto la cui intersezione con il raggio è più vicina all'origine del raggio.

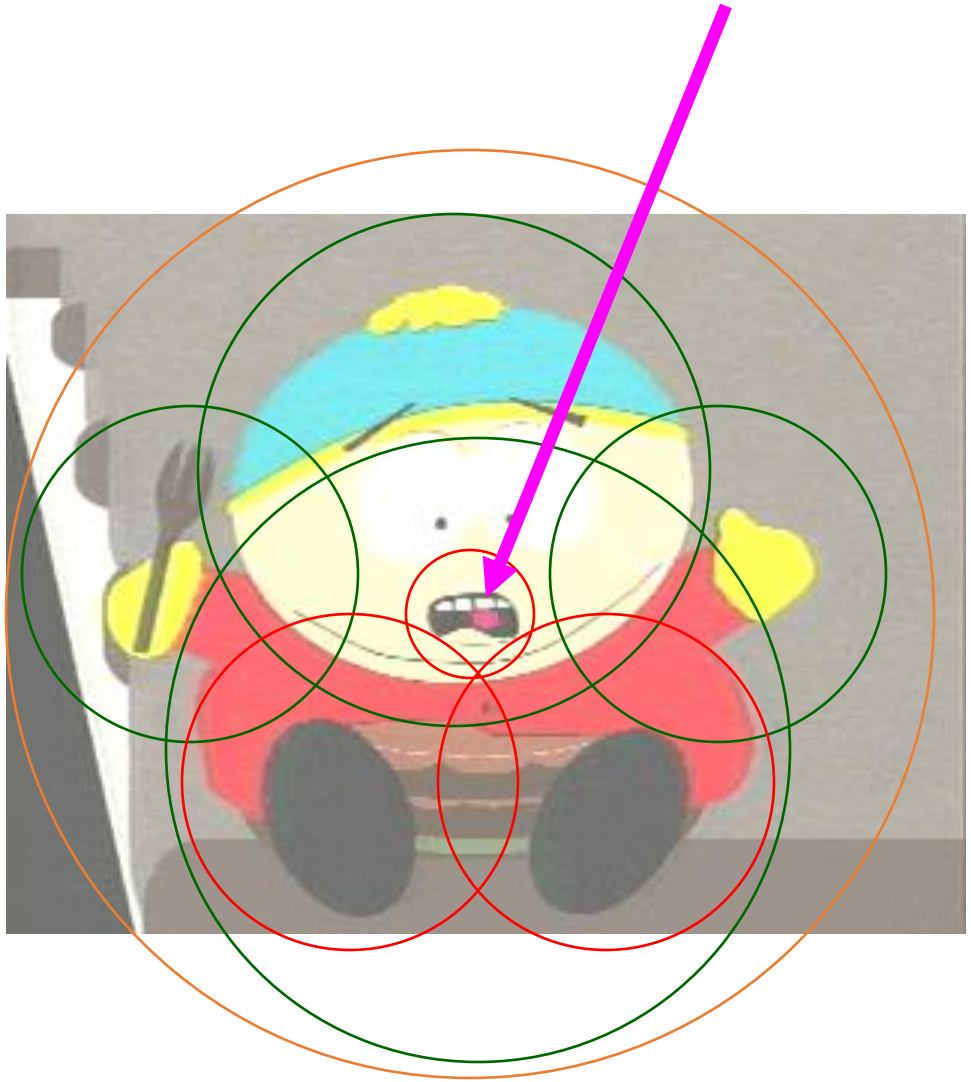


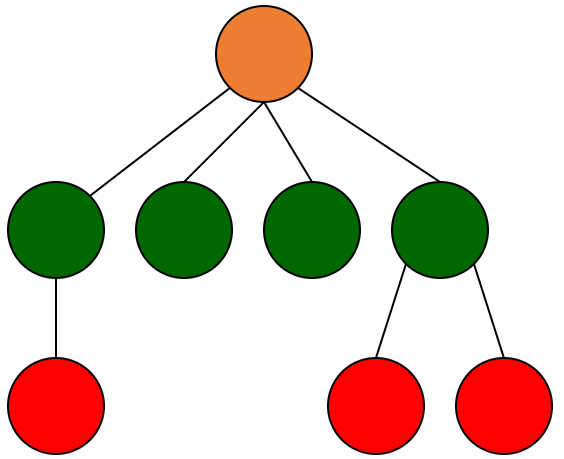
- Sono state formulate varie tecniche allo scopo di limitare il numero di intersezioni che devono essere realizzate: Gerarchie Partizionamento Spaziale.

Gerarchie

- Sebbene i bounding volume non determinino da essi stessi l'ordine o la frequenza dei test di intersezione, i bounding volume possono essere organizzati in gerarchie annidate, in maniera tale che ogni nodo interno sia il bounding volume dei suoi figli, mentre le foglie corrispondono ai singoli oggetti.

Gerarchie di Bounding Volume

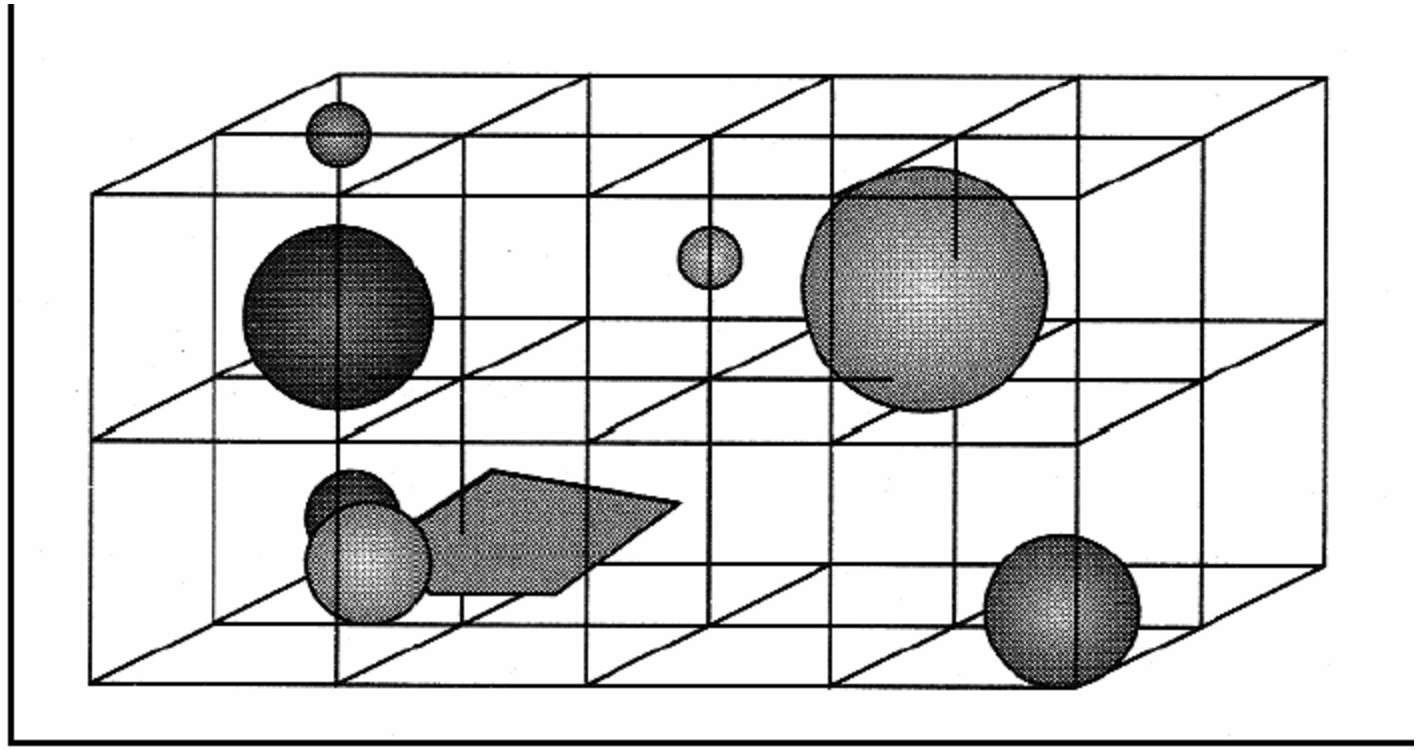




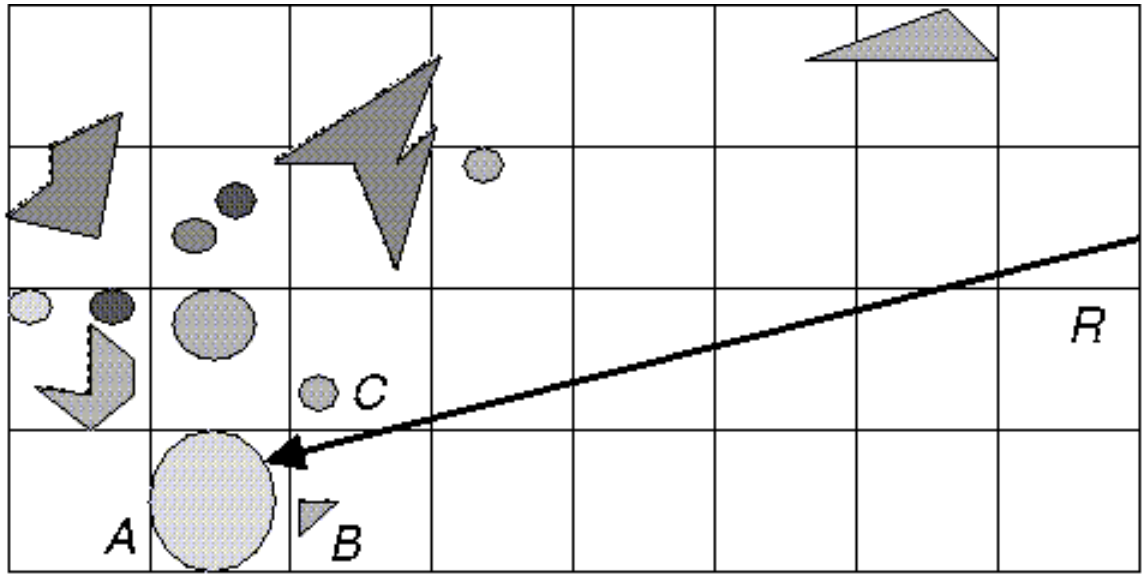
- Un volume figlio non sarà intersecato con un raggio se il suo genitore non lo è.
- Così se il test dell'intersezione comincia con la root, parecchi rami della gerarchia (e quindi parecchi oggetti) possono essere respinti banalmente.

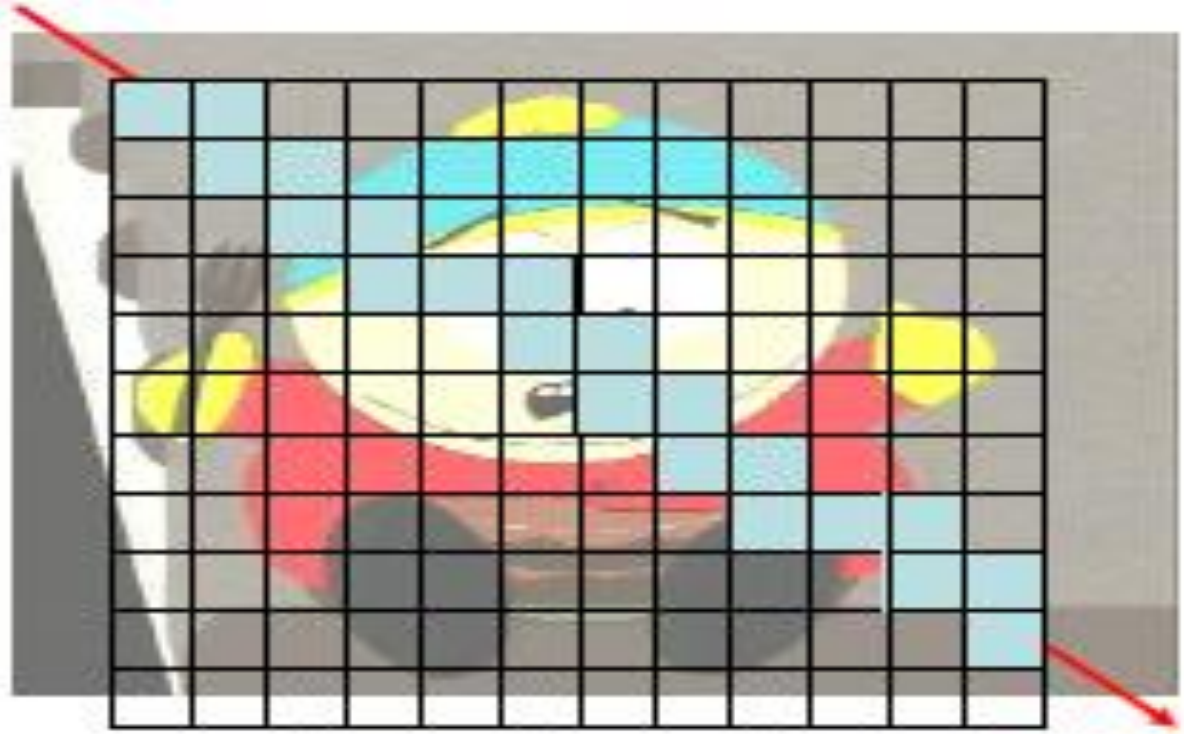
Spatial partitioning

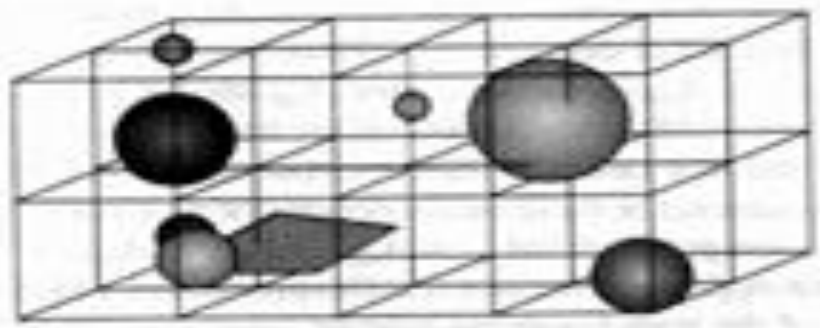
- La scena viene partizionata in una griglia regolare di volumi della stessa dimensione.
- Ad ogni cella viene assegnata una lista di oggetti interamente o parzialmente contenuti in essa.
- Nello spatial partitioning oggetti sono associati a volumi



- Un raggio deve essere intersecato solo con quegli oggetti che sono contenuti dentro le celle in cui passa



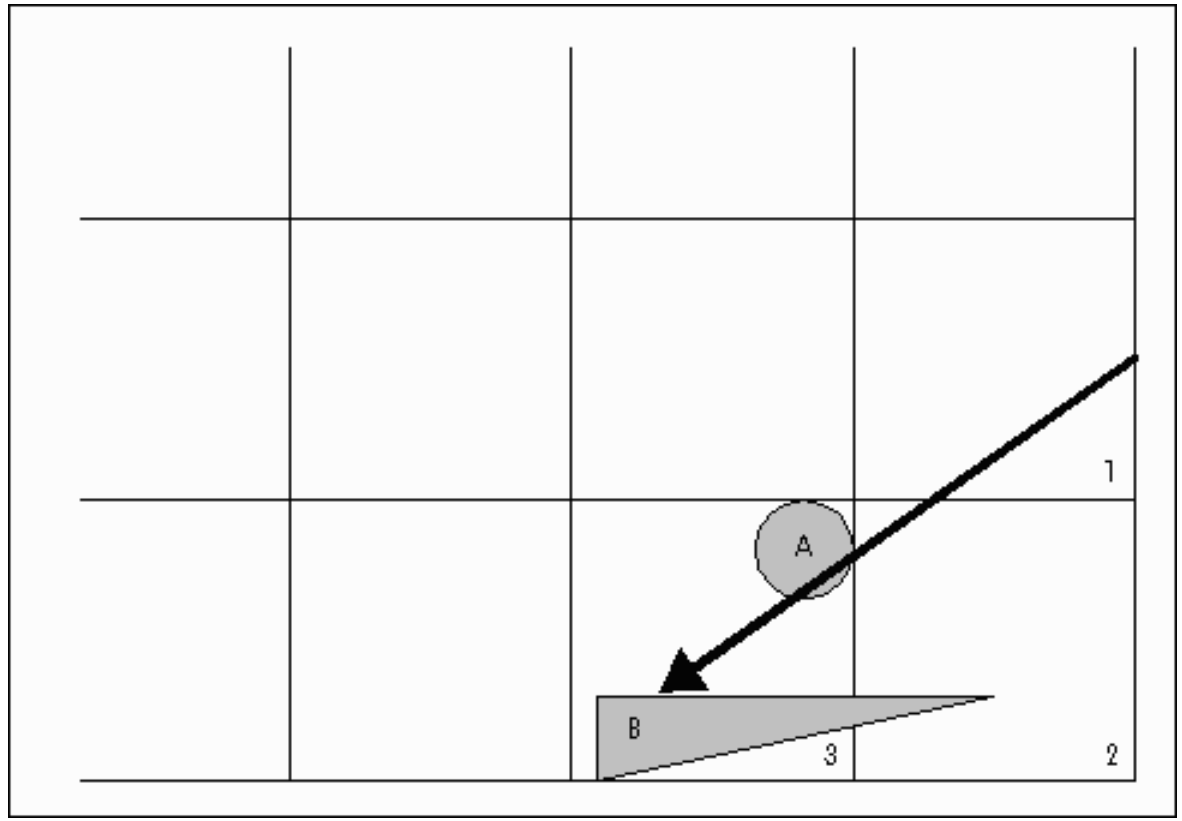




- Prima si controlla se il raggio interseca la partizione e solo in un secondo momento si verifica se esiste intersezione con l'oggetto.
- Poiche' le partizioni vengono esaminate in base all'ordine con cui il raggio le incontra, non appena ne trova una in cui esiste un'intersezione con l'oggetto, nessun'altra viene esaminata (modalita' front-to-back).

- Procedendo in questo modo il processo viene notevolmente sveltito: l'intersezione infatti non viene eseguita per ogni oggetto presente nell'ambiente, ma solo per quelli appartenenti alle partizioni in cui esiste l'intersezione.

- Quando un raggio interseca un oggetto in una partizione, dobbiamo assicurarci che il punto di intersezione appartenga alla partizione in esame;
- E’ possibile che l’intersezione che è stata trovata potrebbe essere ancora lungo il raggio in un’altra partizione e che un altro oggetto potrebbe avere un’intersezione più vicina.
- Potrebbe infatti capitare che l'oggetto sia contenuto contemporaneamente in partizioni diverse e che in una di queste esista un altro oggetto la cui intersezione con il raggio e' piu' vicina della precedente


1
A
B
3
2