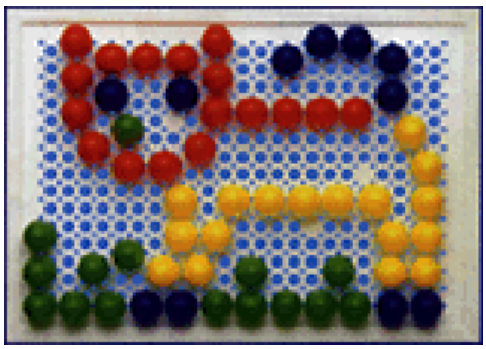
Grafica computazionale

Lezione 3

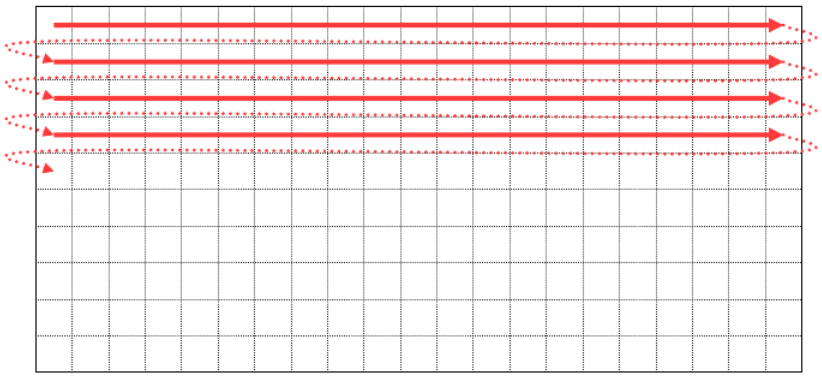
(slide parzialmente basate su
Computer Graphics - MIT Opencourseware
Grafica Computazionale - Massimiliano Corsini – Università di Siena)

Rasterizzazione

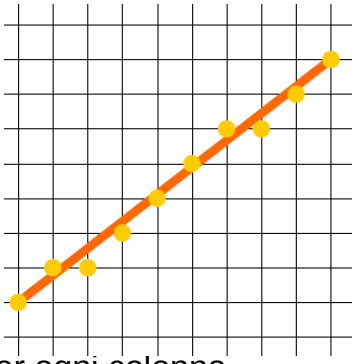
- Con il termine rasterizzazione si intende il processo che trasforma una primitiva geometrica definita in uno spazio continuo 2D nella sua rappresentazione discreta, composta da un insieme di pixel



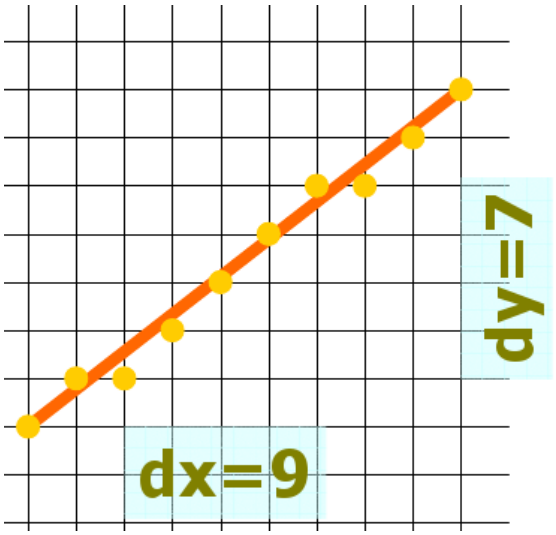
- Gli algoritmi di rasterizzazione si dicono anche di scan-conversion dal nome delle linee (scan-line) di pixel che compongono l'immagine raster sul dispositivo di output.

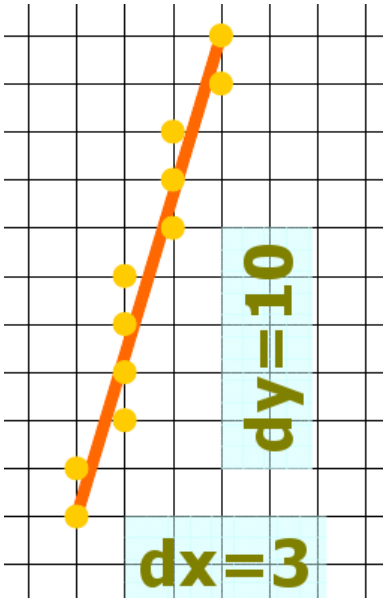


Rasterizzazione di Segmenti

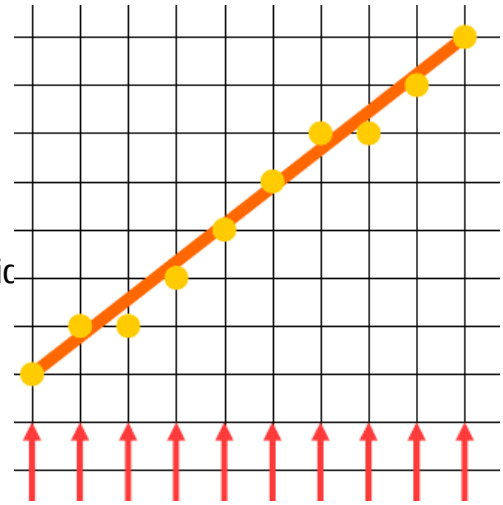


- L'algoritmo di rasterizzazione di un segmento di retta deve individuare le coordinate dei pixel che giacciono sulla linea ideale o che sono il più vicino possibile ad essa
- La sequenza di pixel deve approssimare al meglio il segmento.
- Lo spessore minimo del segmento rasterizzato (idealmente nullo) risulterà di un pixel;
- Per coefficienti angolari $|m| \leq 1$ la rasterizzazione presenta un pixel per ogni colonna.
- Per coefficienti angolari $|m| > 1$ la rasterizzazione presenta un pixel per ogni riga.





Soluzione analitica



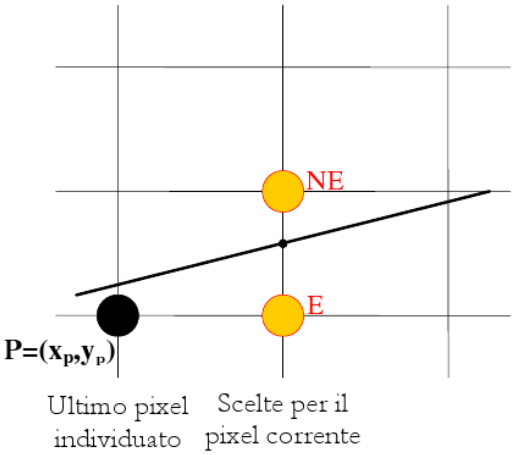
- A partire dal pixel con coordinata x minima x0:
  - Incrementare x con passo costante uguale a 1;
  - calcolare y_i come $y_i = mx_i + B$;
  - Arrotondare y_i. all'ordinata intera più vicina.
- L'algoritmo analitico seleziona il pixel più vicino alla linea ic il pixel cioè che ha distanza minima dalla linea;
- L'individuazione di un pixel implica 3 operazioni:
  - Una moltiplicazione (mx_i),
  - un'addizione (mx_i+B)
  - 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)
- Per ogni punto della linea, abbiamo:

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

- 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.

Algoritmo di Bresenham

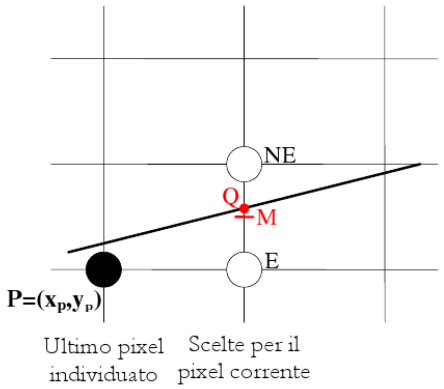
- L'algoritmo di Bresenham (detto anche algoritmo del punto di mezzo) risolve il problema dell'errore introdotto dall'uso di aritmetica floating point nell'algoritmo DDA;
- L'algoritmo di Bresenham fa uso solo di operazioni in aritmetica intera;
- E' ancora un algoritmo di tipo differenziale; fa uso 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.
- 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



Algoritmo di Bresenham

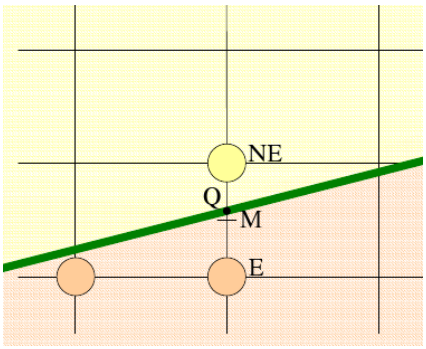
- Indichiamo con Q il punto in cui il segmento interseca la retta $x = x_p+1$.
- Il prossimo pixel è quello, tra E e NE, con distanza minima da Q.
- La scelta di un pixel è ricondotta alla misura di una distanza.
- Detto M il punto di mezzo del segmento E-NE, si deve scegliere il punto che sta dalla 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
- $y = \frac{dy}{dx}x + B \qquad F(x,y) = -dx\,y + dy\,x + dx\,B = 0$

$$F(x,y) = (x_0 - x_1)y + (y_1 - y_0)x + (x_1 - x_0)B$$



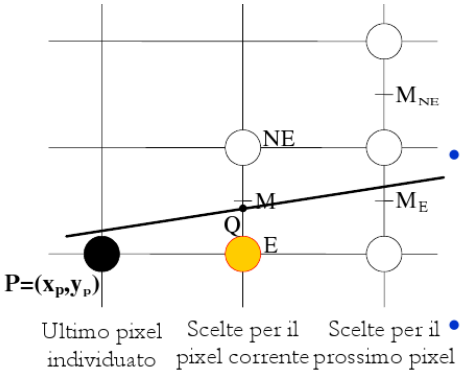
Algoritmo di Bresenham

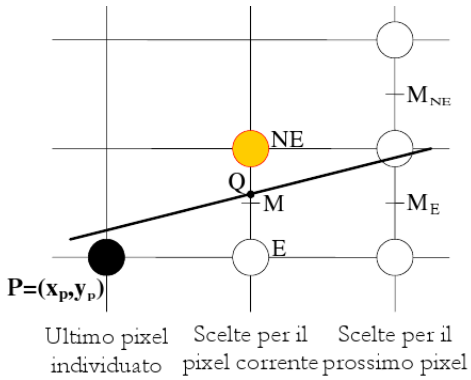
- La funzione F:
  - vale 0 per tutti i punti della retta;
  - assume valori positivi sotto la retta;
  - assume valori negativi sopra la retta.
- F(Q)=0
- La scelta tra E e NE si riduce alla valutazione del segno della funzione F nel punto M.
- Indichiamo d come variablile di decisione
  d=F(M) = F(x_p + 1, y_p + 1/2)
  - d<0 scelgo E
  - d>0 scelgo NE
  - d=0 indifferente
- L'analisi della relazione geometrica tra due punti si riduce quindi a valutare il segno di una funzione.



Algoritmo di Bresenham

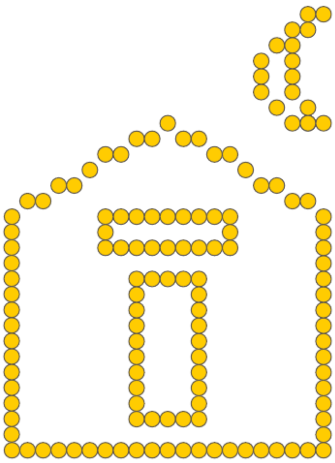
- L'algoritmo di Bresenham costruisce anche d in modo incrementale.
- A tal fine è necessario individuare il punto M al prossimo passo (x = x_p+2) sulla base della scelta fatta al passo corrente.
- Se l'ultimo pixel selezionato è stato E
  - d_{new} = F(x_p + 2, y_p + 1/2)
  - d_{new} = a (x_p + 2) + b(y_p + 1/2) + c
  - Poiché d = a (x_p + 1) + b(y_p + 1/2) + c sottraendo si ha d_{new} = d_{old} + a
  - L'incremento da aggiungere a d dopo aver scelto E è quindi: Δ E = a = dy
- Se invece l'ultimo pixel selezionato è stato NE:
  - d_{new} = F(x_p + 2, y_p + 3/2)
  - d_{new} = a (x_p + 2) + b(y_p + 3/2) + c
  - Quindi d_{new} = d_{old} + a + b da cui Δ NE = a + b = dy − dx

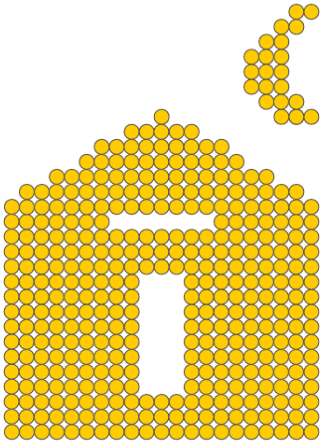




Rasterizzazione di Poligoni

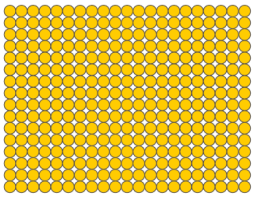
- La tecnologia raster permette di “disegnare” primitive geometriche rappresentate dal solo contorno e primitive “piene”.





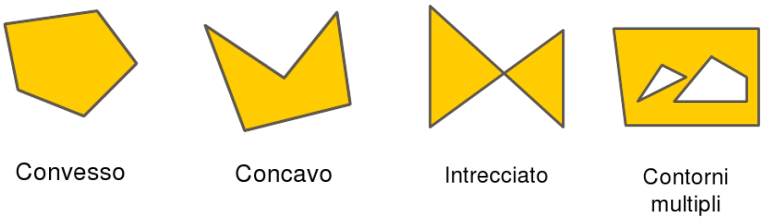
- Disegnare un rettangolo vuoto consiste nell’applicare 4 volte l’algoritmo di rasterizzazione dei segmenti che ne rappresentano i lati;

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



Rasterizzazione di Poligoni

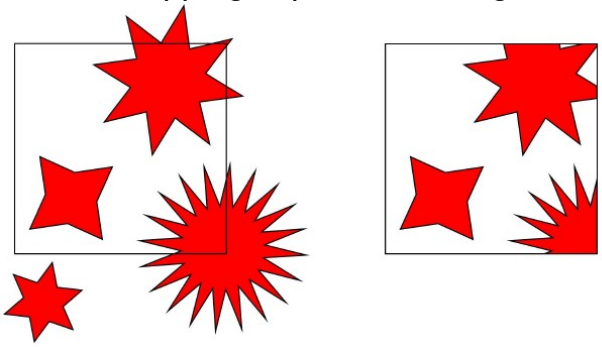
- L’algoritmo di rasterizzazione deve operare correttamente sulle diverse tipologie di primitive geometriche:
  - Poligono Convesso
  - Poligono Concavo
  - Poligono Intrecciato
  - Poligono A Contorni multipli



-
- 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 (!!)

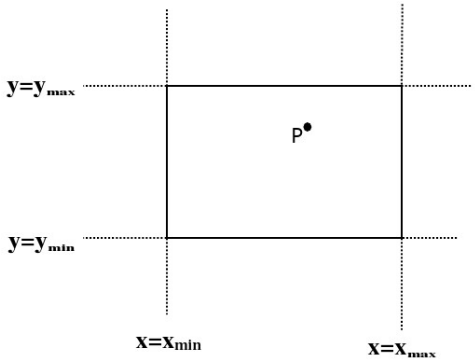
Clipping

- 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



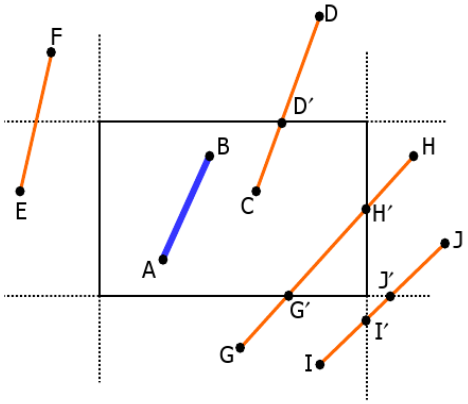
- **Clipping di un punto:** un punto è all'interno del rettangolo di clipping se e solo se sono soddisfatte le 4 disuguaglianze:

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



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.

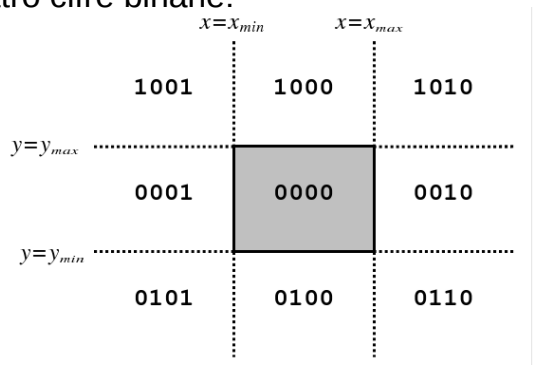


- L'approccio diretto alla soluzione del problema è quello di determinare le intersezioni tra la retta su cui giace il segmento e le 4 rette su cui giacciono i lati del rettangolo di clipping;
- Individuati i punti di intersezione occorre verificare l'effettiva appartenenza al rettangolo di clipping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

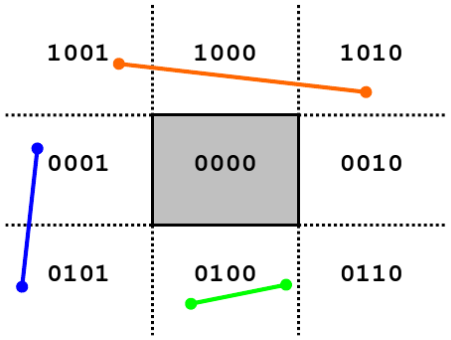
  $x = xa + t\,(xb - xa)$ $y = y\,a + t\,(y\,b - y\,a)$ $t \in [0,1]$
- Per ogni coppia segmento-lato di rettangolo si risolve il sistema di equazioni parametriche che definiscono il segmento in funzione di t_{segm} ed il lato in funzione di t_{lato};
- Se t_{segm} e t_{lato} assumono valori nell'intervallo [0, 1] allora l'intersezione appartiene al segmento ed al rettangolo di clipping;
- È necessario verificare in anticipo il parallelismo tra le linee prima di determinare l'intersezione;
- Algoritmo costoso e quindi inefficiente.

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}



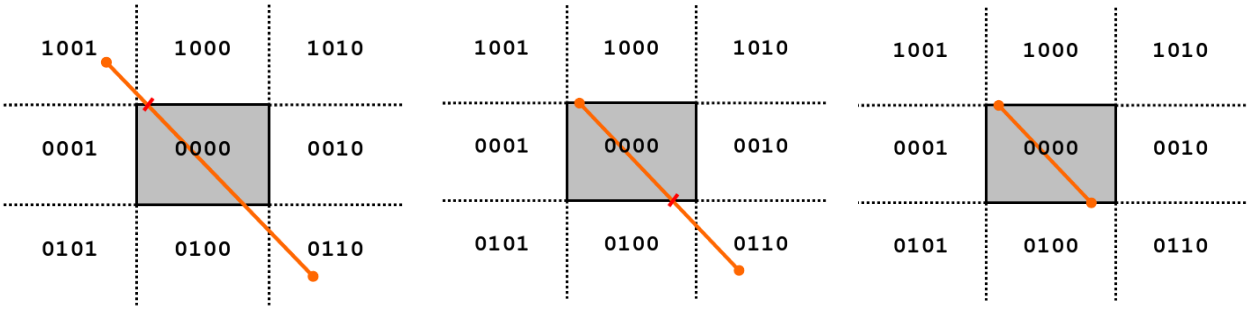
- Il clipping di un segmento prevede la codifica (e confronto) dei suoi estremi sulla base delle regioni di appartenenza;
- Se il codice di entrambi gli estremi è 0000 (OR logico tra i codici ritorna un risultato nullo), allora si può banalmente decidere che il segmento è interamente interno al rettangolo di clipping.
- 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 ∧ 1010 = 1000
0100 ∧ 0100 = 0100
0001 ∧ 0101 = 0001

Algoritmo Cohen-Sutherland

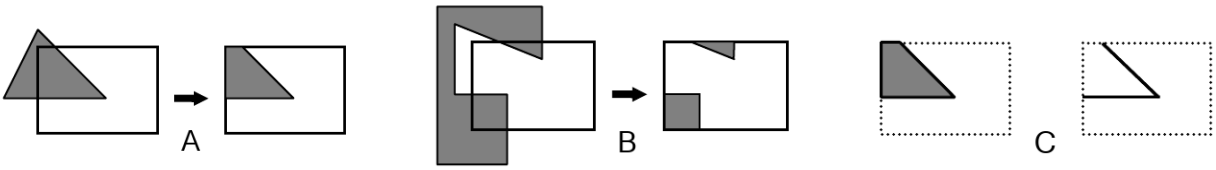
- Se il risultato dell'AND è nullo (ed almeno uno dei codici associati ai vertici è diverso da 0000):
  - Si individua l'intersezione tra il segmento ed il lato relativo al primo bit discordante tra i codici (in fig., bit 1, y=y_{max});
  - L'estremo con bit a 1 (in prima posizione nell'esempio) viene 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.



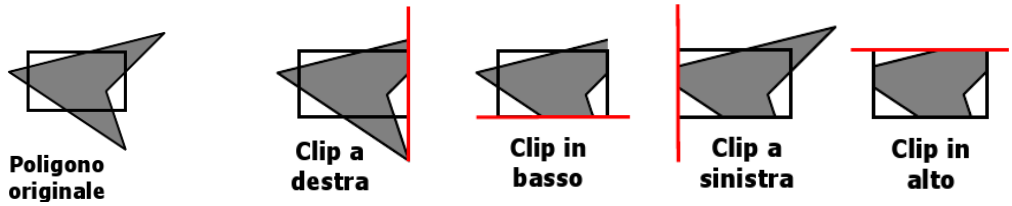
- Ad ogni iterazione si controlla l'eventuale terminazione del processo (OR logico nullo);
- L'algoritmo rimuove progressivamente le parti esterne; risulta efficiente quando molti dei segmenti da clippare sono completamente esterni al rettangolo di clipping.

Clipping di un 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).



- 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.
- Approccio divide et impera;
  - Il problema è ricondotto al clipping di un poligono generico rispetto ad una retta;
  - La procedura è applicata sequenzialmente alle 4 rette che definiscono il rettangolo di clipping.



Rimozione delle Superfici Nascoste

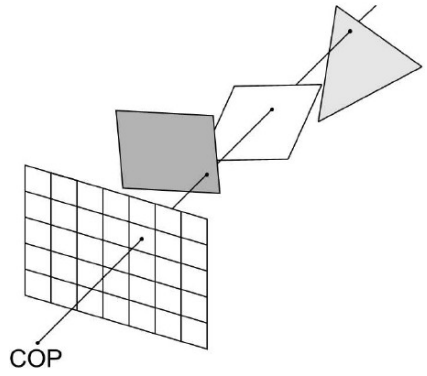
- Gli oggetti della scena sono generalmente opachi;
- Gli oggetti più vicini all'osservatore possono nascondere (occludere) la vista (totale o parziale) di oggetti più lontani;
- Il problema della rimozione delle superfici nascoste (Hidden Surface Removal) consiste nel determinare le parti della scena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

Approccio 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 oscura B; solo A deve essere visualizzata;
  - A e B sono completamente visibili; entrambe le primitive sono visualizzate;
  - A occlude parzialmente B: è necessario individuare le parti visibili di B.
- **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 $O(k^2)$

Approccio Image-Space

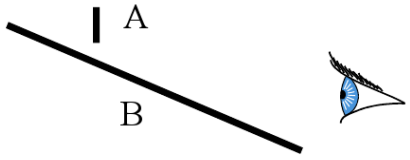
- Per ogni pixel del piano immagine si considera una semiretta avente origine nel centro di proiezione e passante per il pixel in esame. La semiretta attraversa la scena fino a colpire una primitiva o a perdersi sul fondo.
- Per ogni primitiva si calcola l'intersezione della semiretta con il piano di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

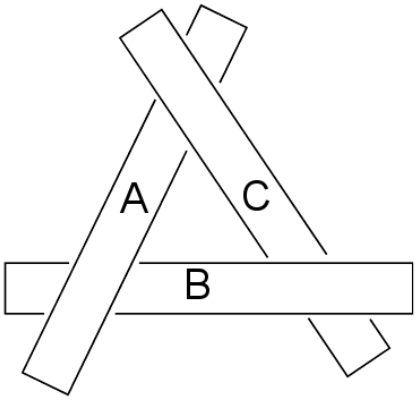


- L'operazione fondamentale dell'approccio image-space è il calcolo delle intersezioni tra semiretta e primitive (per ogni semiretta massimo si hanno k intersezioni);
- Per un display n x m, questa operazione deve essere eseguita n x m x k volte, la complessità risulta comunque, tenendo conto del numero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

Algoritmo del Pittore (depth-sort)

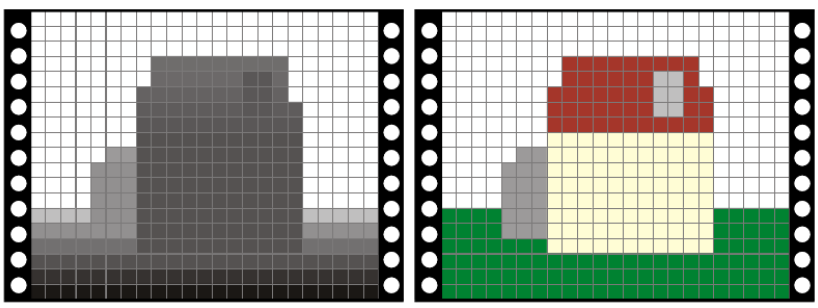
- Gli oggetti della scena 3D siano rappresentati mediante primitive geometriche (poligoni) planari;
- I poligoni planari siano ordinati sulla base della loro distanza dall'osservatore;
- L'idea di base è quella di seguire un approccio analogo a quello usato da un pittore: dipingere prima il poligono 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.
- 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;
- Come ordiniamo? Esiste sempre un ordine (NO!)





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.
- 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.



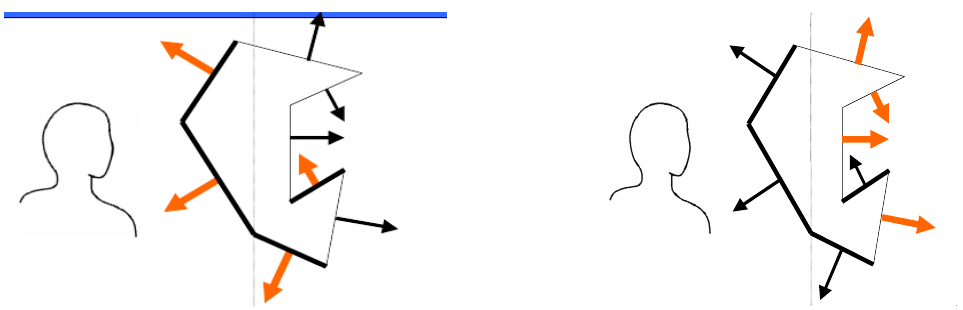
Algoritmo Z-Buffer

- 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).

- Problemi a gestire trasparenze parziali

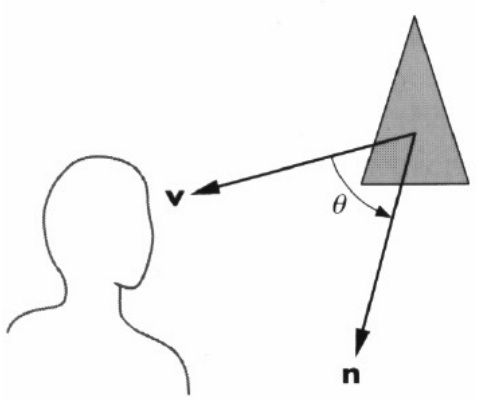
Ottimizzazione: Back-face Culling

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



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 θ possiamo valutare il prodotto scalare $n \cdot v \geq 0$
- 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.



Cos'è un pixel?

- Un pixel non è:
  - Un rettangolo
  - Un disco
  - Una minuscola lucetta
- Un pixel è un punto
  - Non ha dimensione
  - Non occupa area
  - Non può essere visto
  - Può avere delle coordinate
- Un pixel è più di un semplice punto: è un campione!
- Le informazioni geometriche sono continue, ma le rappresentazioni digitali sono discrete
- Campionamento: processo che mappa funzioni continue in funzioni discrete
- Quantizzazione: processo che mappa variabili discrete in variabili continue
- Per rappresentare una immagine digitale dobbiamo sia campionare che quantizzare

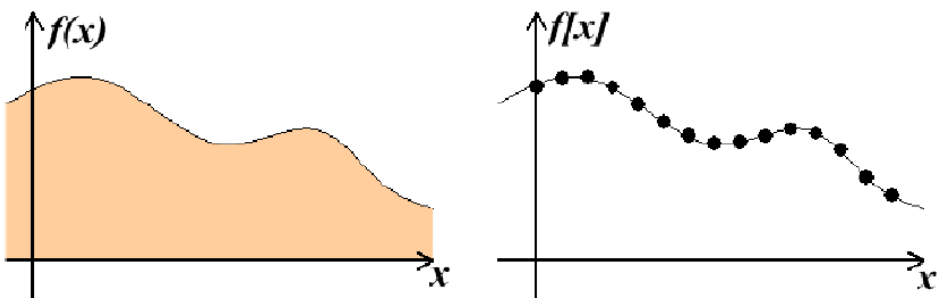
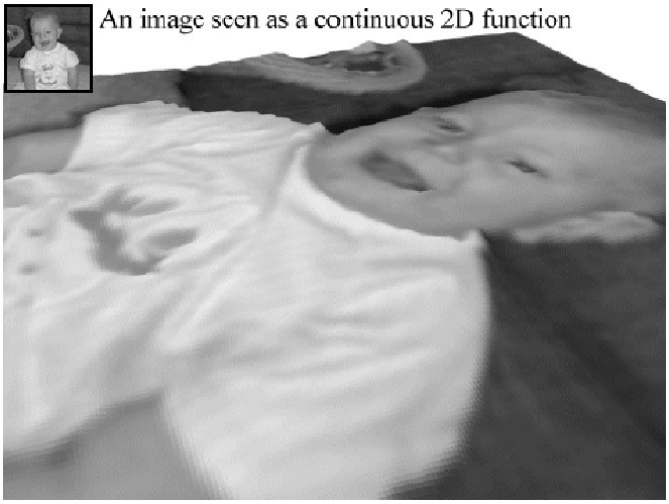
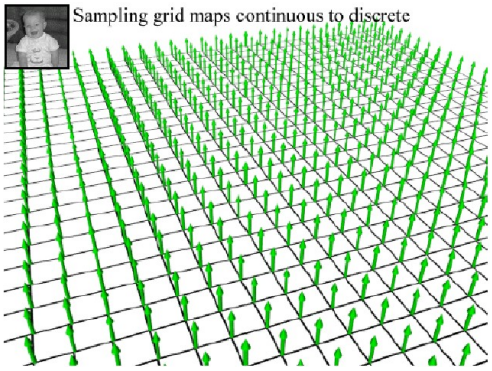
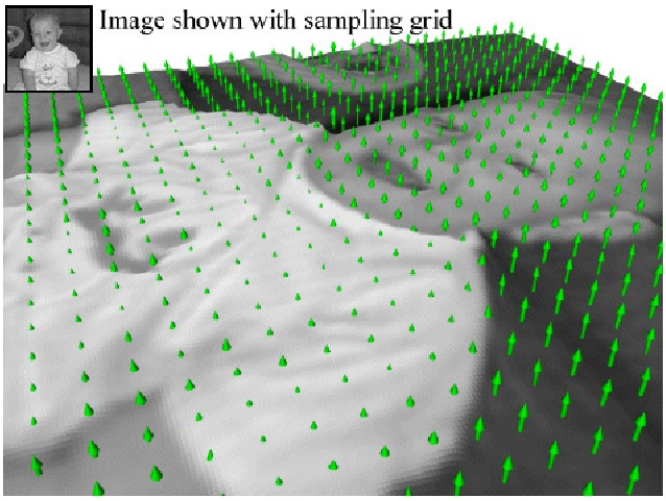


Immagine come Funzione 2D

- Una immagine ideale è una funzione I(x,y) di intensità luminose
  - Rappresentabile come campo di altezze
- In generale non può essere rappresentata come una funzione analitica continua
- La rappresentiamo come tabella di valori...
- Ma come riempiamo questa tabella?
- Guardiamo il valore in alcuni punti specificati (campionamento)
- In particolare campioniamo i valori in una griglia regolare
- Questo processo di campionamento può portare a degli errori e produrre artefatti

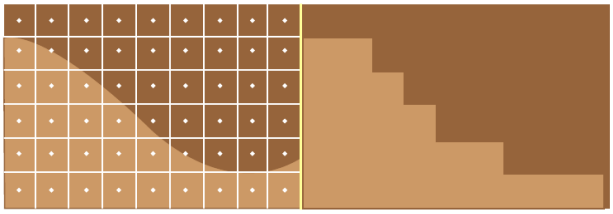




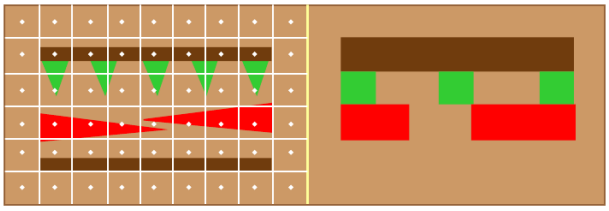


Aliasing

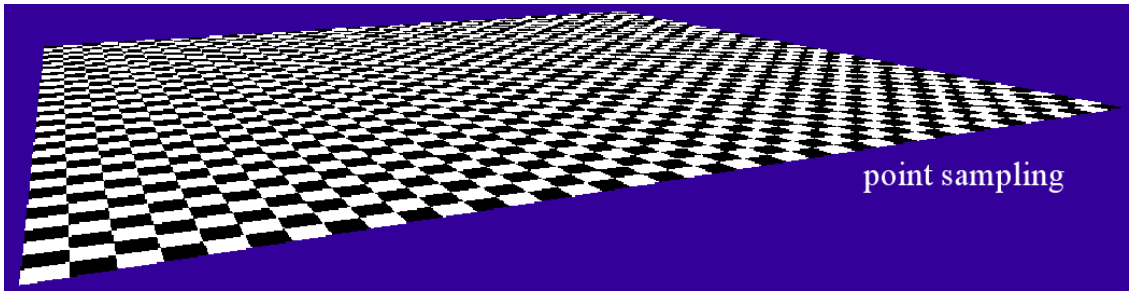
- Con il termine Aliasing si indicano tutti gli errori dovuti al campionamento di un segnale continuo:
  - Bordi scalettati



  - Resa impropria del dettaglio

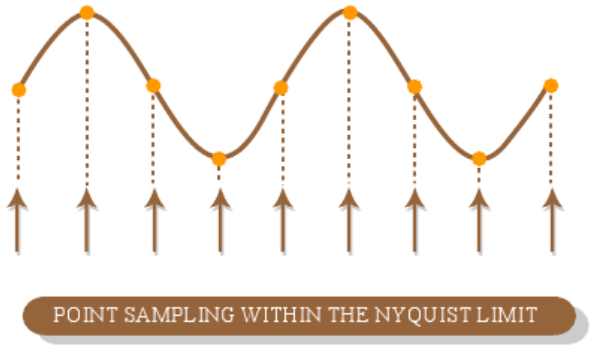


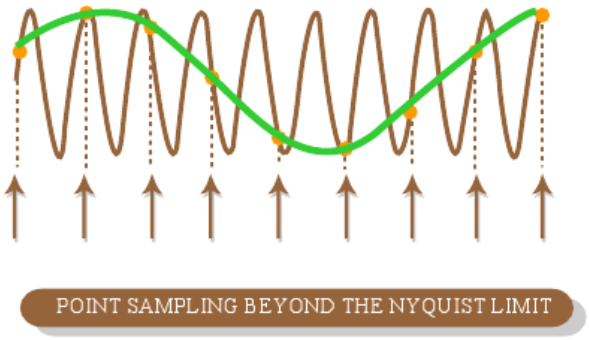
  - Errori nelle textures



Campionamento e Ricostruzione

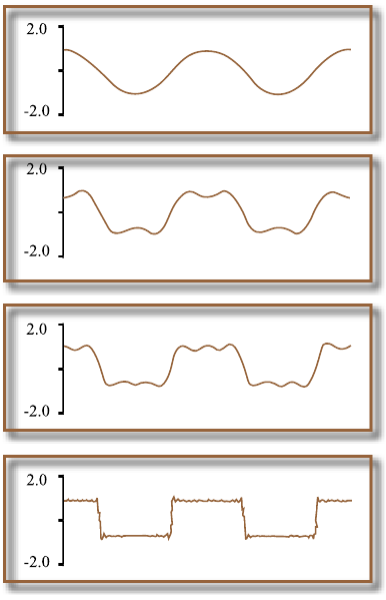
- Quanto densamente dobbiamo campionare l'immagine per catturarne l'essenza?
  - Se campioniamo insufficientemente non siamo in grado di ricostruirla accuratamente...
- Limite di Nyquist e teorema del campionamento di Shannon:
  - Se campioniamo uniformemente un segnale siamo in grado di ricostruire univocamente solo fino alle frequenze pari alla metà del numero dei campioni.
  - Se campioniamo un segnale in modo insufficiente, le alte frequenze possono essere confuse per frequenze più basse in ricostruzione (aliasing)

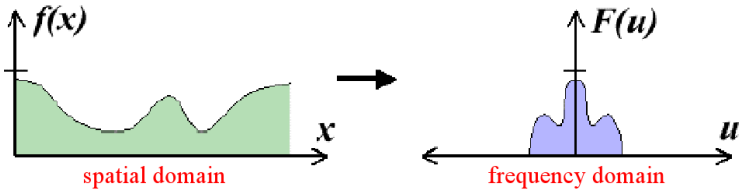




Trasformate di Fourier

- Tutti i segnali periodici possono essere rappresentati come somma di onde sinusoidali
- Ogni segnale periodico nel dominio spaziale ha un duale nel dominio delle frequenze (coefficienti della soma di onde)





- Possiamo passare da un dominio all'altro con la Trasformata di Fourier

frequency domain — spatial domain

Fourier Transform $$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\, e^{-i2\pi(ux+vy)}\, dxdy$$

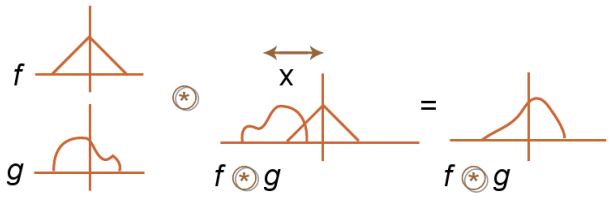
Inverse Fourier Transform $$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)\, e^{\,i2\pi(ux+vy)}\, dudv$$

- Il dominio delle frequenze risulta essere uno spazio più conveniente per studiare le proprietà di un segnale

Convoluzione

- La convoluzione descrive come un sistema con risposta d'impulso h(x) risponde ad un segnale f(x)

$$f(x) * h(x) = \int_{-\infty}^{\infty} f(\lambda) h(x-\lambda) d\lambda$$



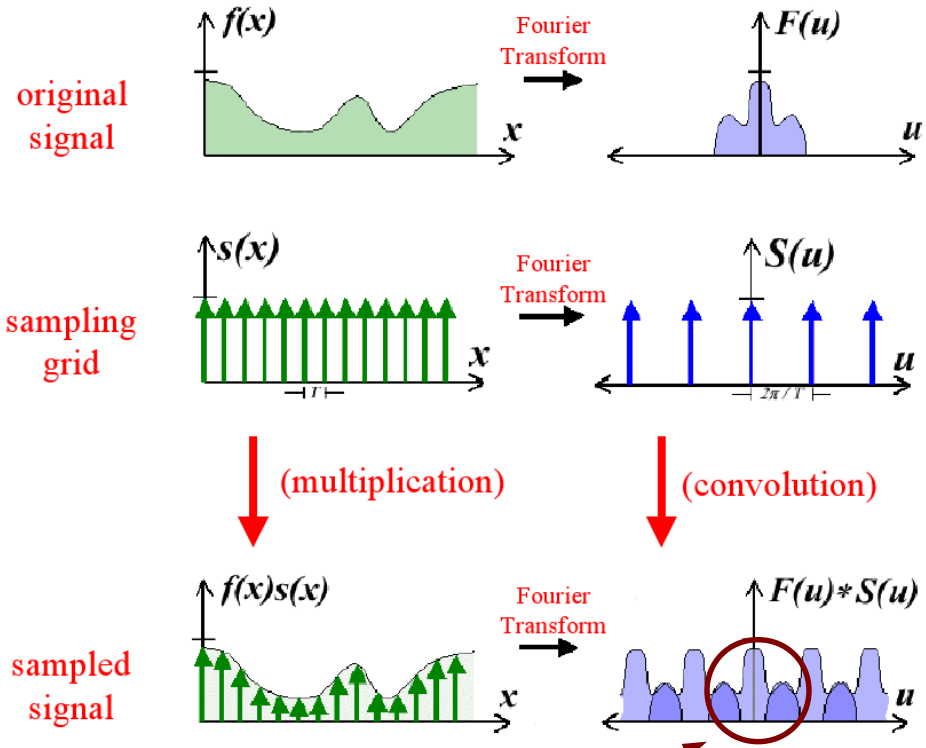
- La convoluzione nel dominio spaziale è equivalente ad una moltiplicazione in frequenza

$$f(x) * h(x) \rightarrow F(u)H(u)$$

- La convoluzione nel dominio delle frequenze è equivalente ad una moltiplicazione spaziale

$$F(u) * H(u) \rightarrow f(x)h(x)$$

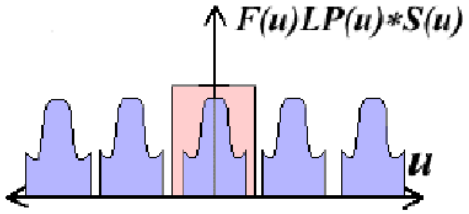
Campionamento nello spazio delle frequenze



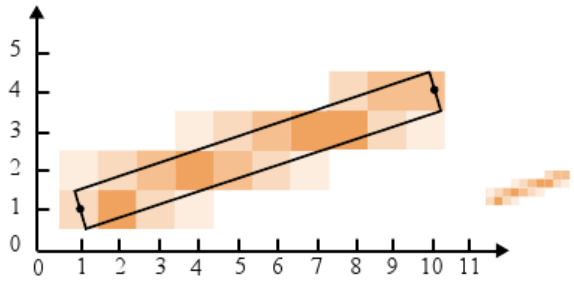
- Se siamo in grado di estrarre una copia del segnale originale dalle frequenze del segnale campionato, possiamo ricostruire il segnale originale!
- Ci può essere overlap tra le copie
- Se non siamo in grado di separare le copie avremo aliasing!

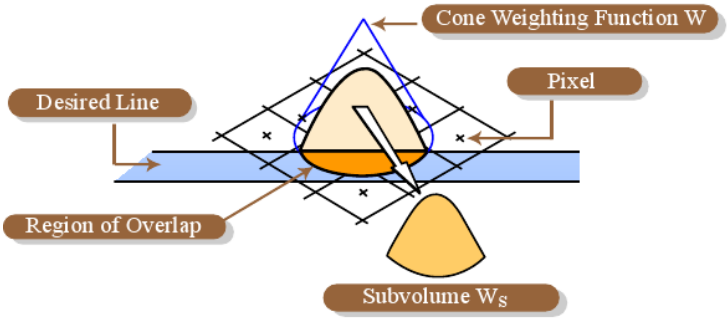
Approcci per evitare l'aliasing

- *Pre-filtering*: separare i segnali rimuovendo le alte frequenze dal segnale originale



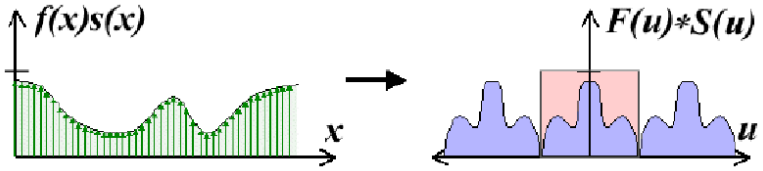
  - Filtriamo le primitive continue
  - Trattiamo un pixel come un area e calcoliamo la quantità di overlap
    - Che funzione di peso usare?



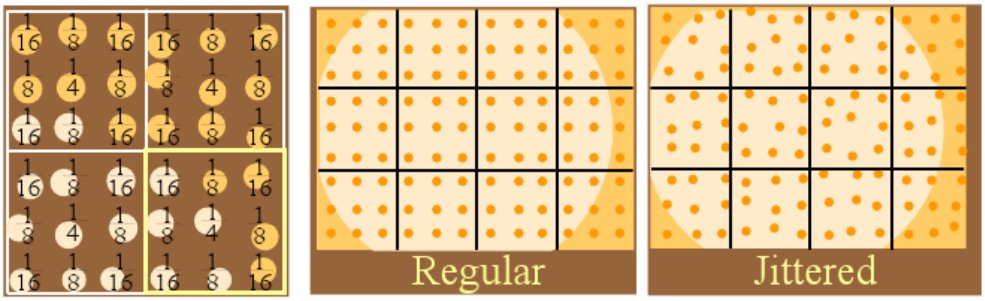


Approcci per evitare l'aliasing

- *Super-sampling*: separare i segnali aumentando la densità di campionamento

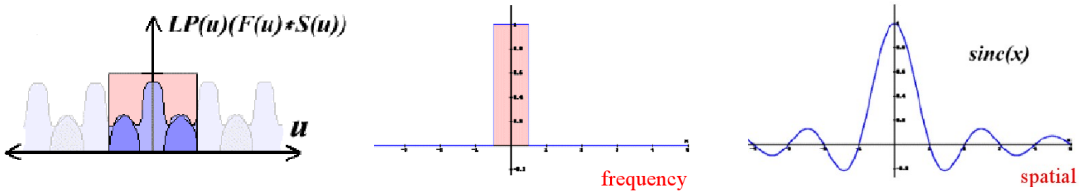


  - Filtriamo i campioni
  - Calcoliamo la media pesata di più campioni
    - Campionamento Regolare o Jittered (meglio)

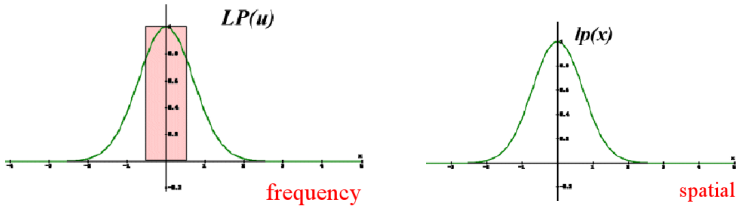


Filtri di ricostruzione

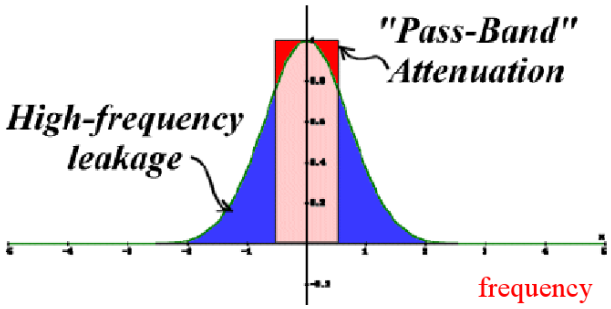
- *Filtro ideale*: sfortunatamente ha estensione spaziale infinita - Troppo costoso!



- *Filtro Gaussiano*: simile a quanto fa il CRT (o la retina)



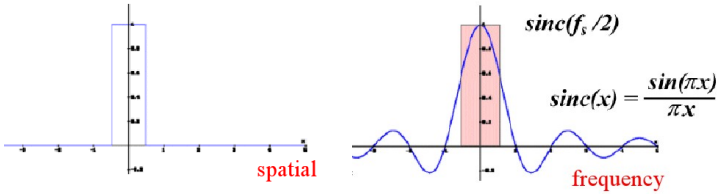
- Molti artefatti sono dovuti da filtri di ricostruzione inadatti
  - Eccessiva attenuazione passa-banda dà immagini sfuocate
  - Eccessivo leakage di alte frequenze provoca “ringing”

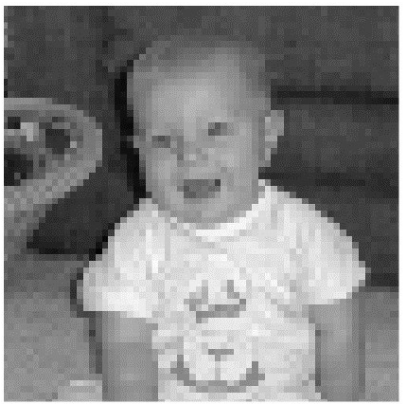


33

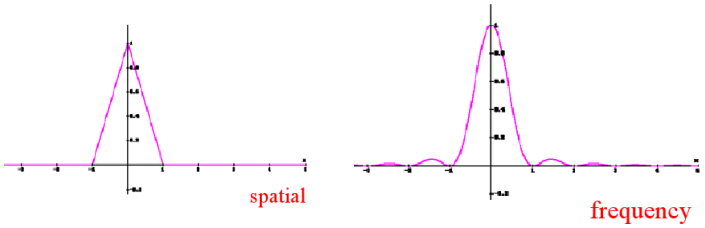
Filtri di ricostruzione

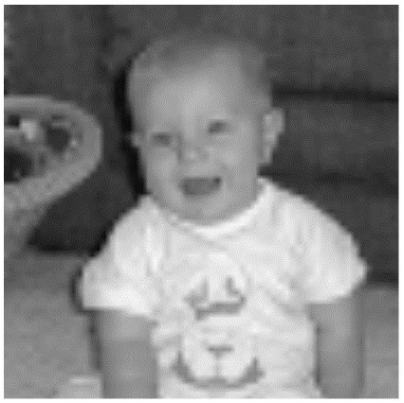
- Box Filter/Nearest Neighbor:
  - si fa finta che i pixel siano piccoli rettangoli
  - Immagine a “quadrettoni”





- Tent Filter / Interpolazione Bi-Lineare:
  - Facile da implementare
  - Buona qualità

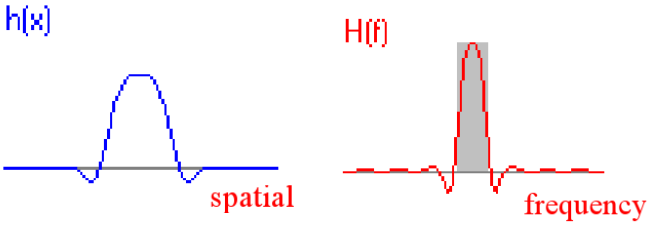




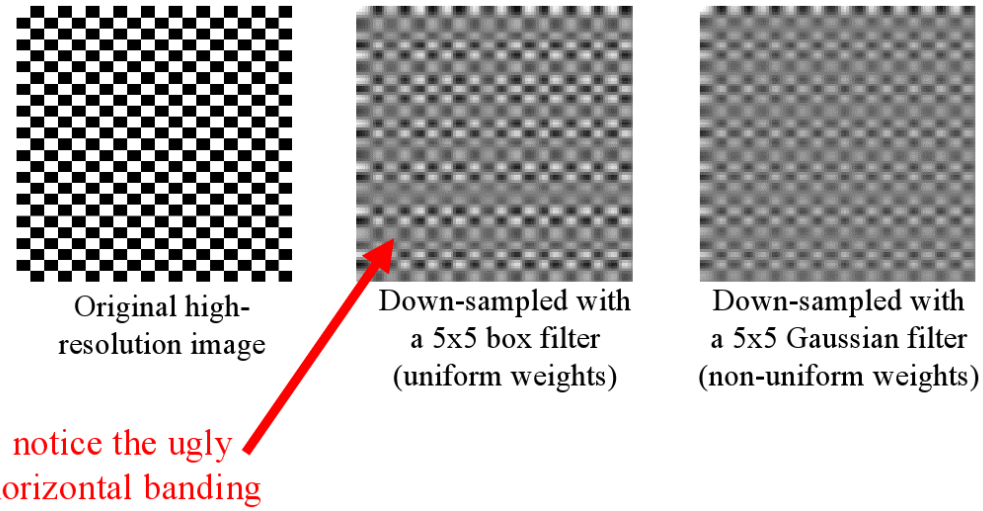
34

Filtri di ricostruzione

- Interpolazione Bi-Cubica
  - Inizia ad approssimare filtro ideale

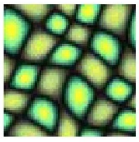


- Perché il box-filter funziona male?

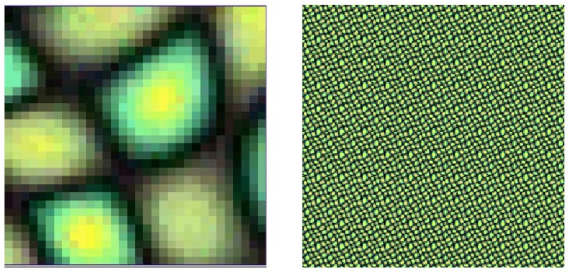


Campionamento e Textures

- Usando texture mapping è raro che la densità di campionamento dello schermo sia uguale a quella della texture

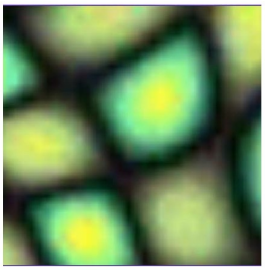

64x64 pixels

Original Texture


Magnification for Display

Minification for Display

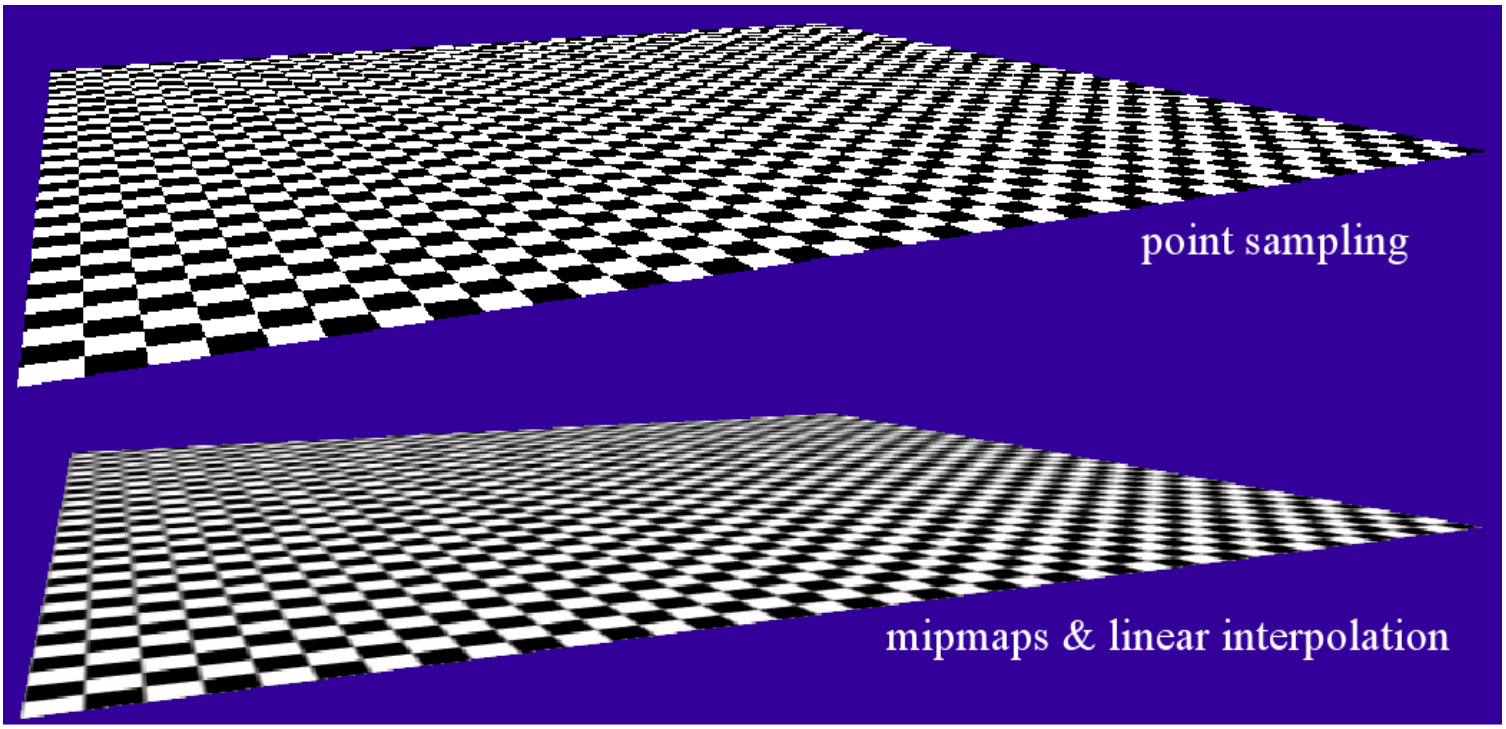
for which we must use a reconstruction filter


Tent filter

- Pre-fitering: Rimuove le alte frequenze che generano artefatti in riduzione
  - Calcola una integrazione spaziale sull'estensione del campione
  - Costoso da fare durante la rasterizzazione, ma può essere precomputata

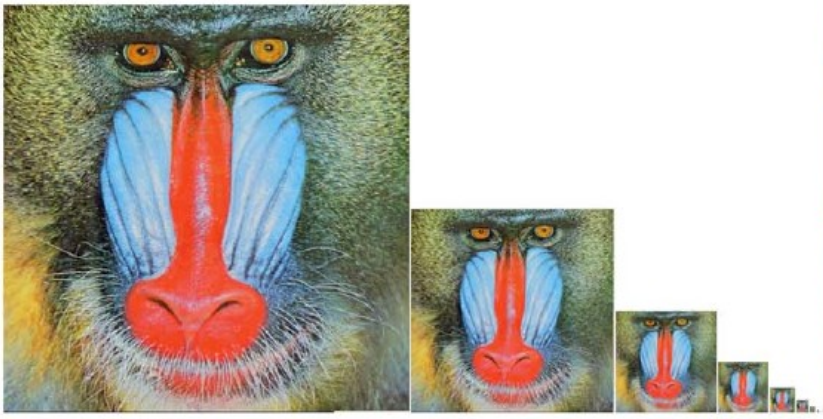
MIP Mapping (Multimum In Parvo)

- Costruiamo una piramide di immagini che sono pre-filtrare e ricampionate a 1/2, 1/4, 1/8, … della risoluzione originale
- Durante la rasterizzazione calcoliamo l'indice della immagine decimata campionata alla densità più vicina a quella desiderata

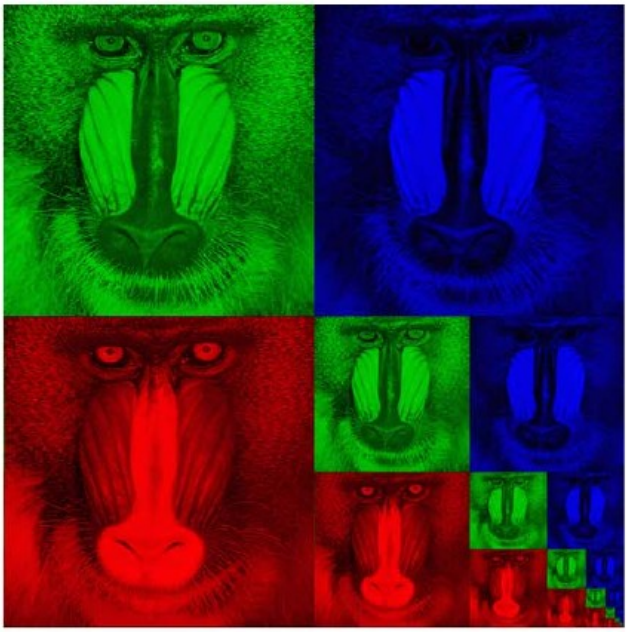


Layout delle MIP map

- Le MIP map possono essere memorizzate in modo efficiente con un overhead di solo 1/3



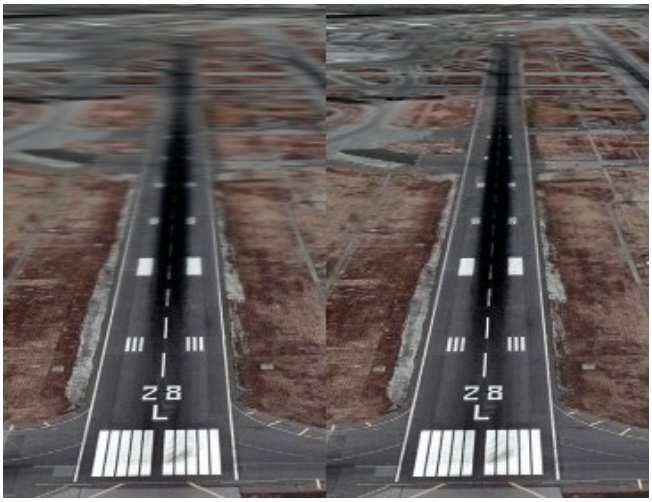
10-level mip map



Memory format of a mip map

Filtro Anisotropico

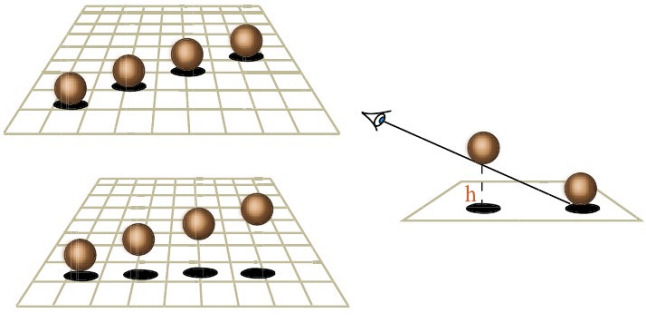
- Cosa succede quando la superficie è molto inclinata?



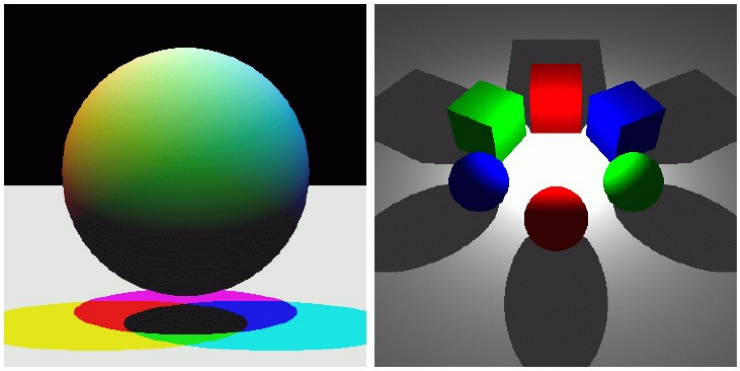
- Possiamo usare mipmap differenti per le 2 direzioni
- Possiamo scegliere il livello del mipmap con risoluzione massima e filtrare nell'altra direzione (filtro anisotropico)
  - Supersampling quindi costoso

Ombre

- Le ombre sono un fenomeno di illuminazione globale
  - Interazione tra luce e più superfici
- Percettivamente molto importante
  - Informazione sulla profondità

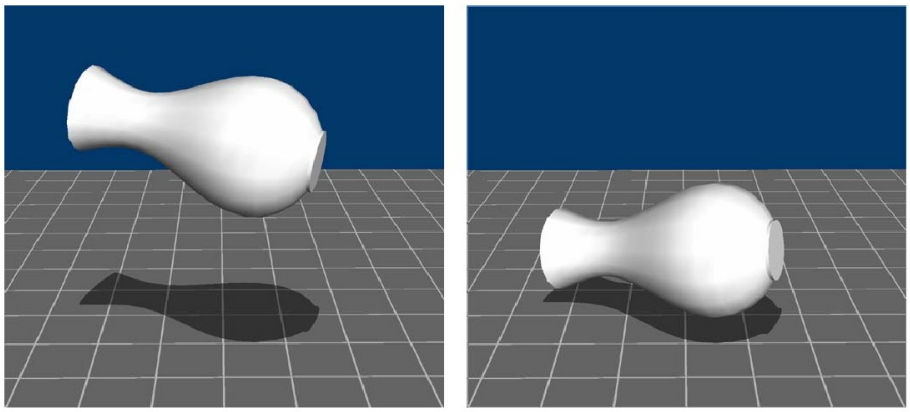


  - Informazione circa l'illuminazione
    - Direzione e distanza della sorgente luminosa

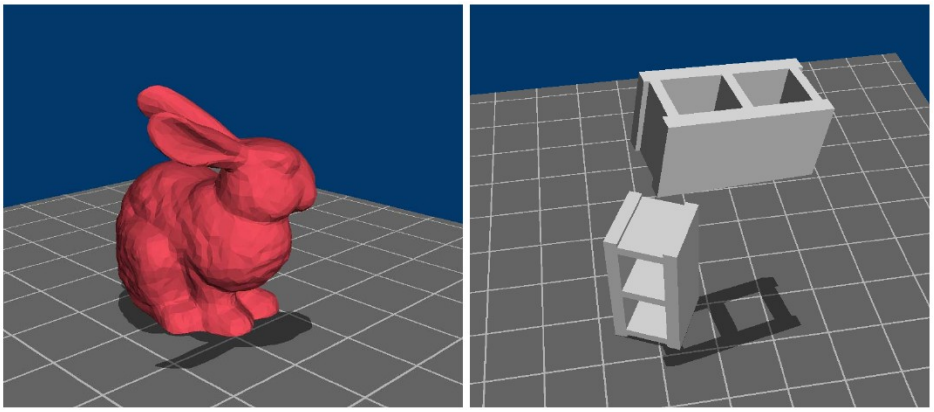


Ombre portate su superfici piane

- Disegnare le primitive geometriche una seconda volta proiettate sul suolo

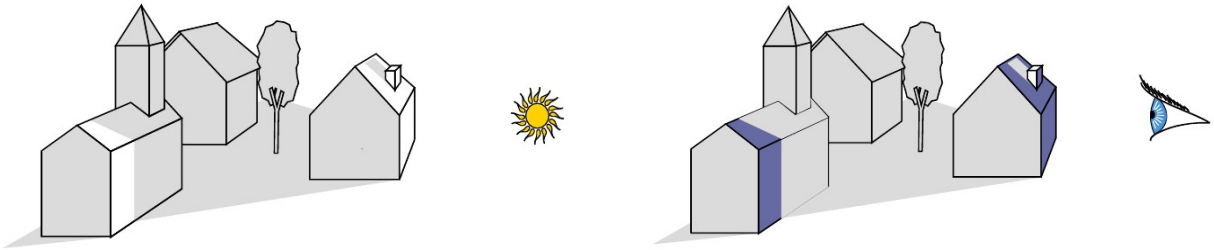


- Limiti:
  - No ombre auto-portate
  - No ombre su altri oggetti
  - No ombre su superfici curve

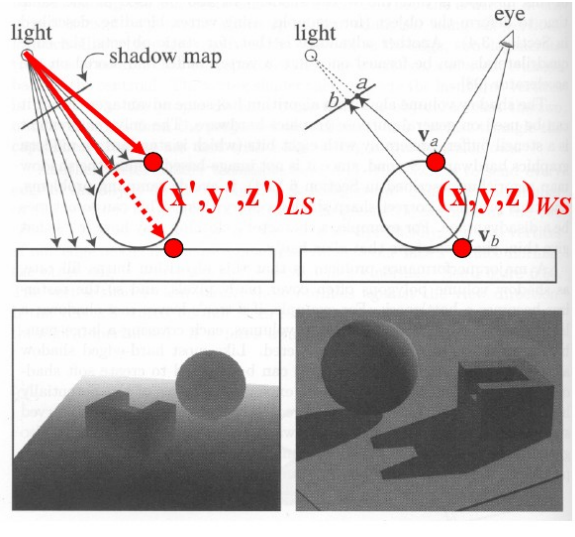


Shadow Mapping

- Dualità ombra vista: un oggetto è in ombra se non è visibile dalla sorgente luminosa



- Creiamo una immagine vista dalla sorgente luminosa
  - Solo z-buffer usando una texture come target di rendering
- Al momento di creare l'immagine vista dal punto di vista dell'osservatore...
  - Per ogni pixel calcoliamo la distanza dalla sorgente luminosa
    - Matrice di proiezione delle luce per ottenere (x', y', z') rispetto alla sorgente luminosa
  - Controlliamo rispetto al valore dello z-buffer (z-test)
    - ShadowMap(x',y') > z'
    - Se passiamo lo z-test in luce
    - Se non passiamo lo z-test in ombra



Shadow Mapping

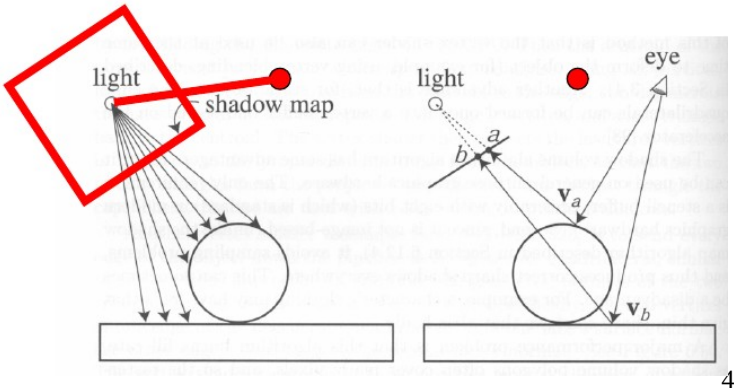
- Può essere calcolato in hardware usando supporto per texture mapping
  - Coordinate texture u,v,w generate usando matrice 4x4 (mapping proiettivo)
  - Hardware moderno permette test sui valori delle texture

Limiti

- Field of View
- Bias (Epsilon)
- Aliasing

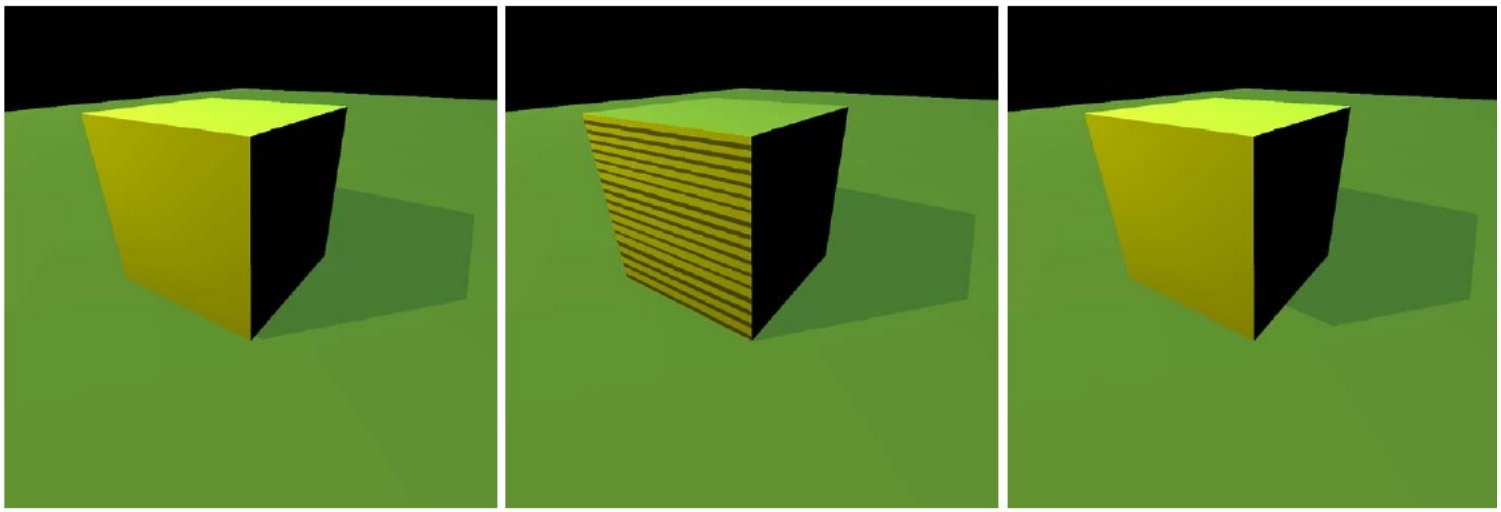
Field of View

- Cosa succede se un punto è fuori del campo visivo della shadow map
  - Usare shadow map cubica
  - Usare solo spotlight



Bias (Epsilon)

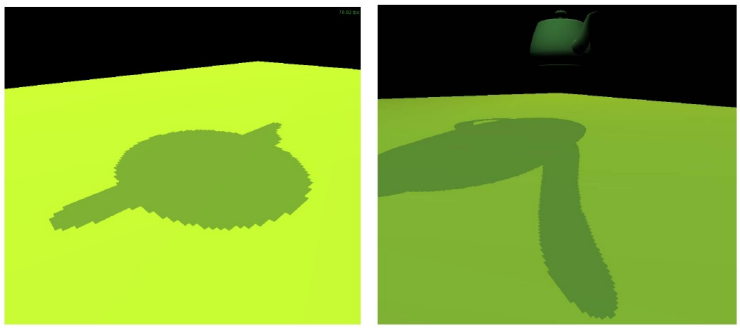
- Per un punto illuminato ShadowMap(x',y')=z'
  - Problemi numerici potrebbero far fallire lo z-test
- Come evitare ombre erronee?
  - Aggiungere un bias (epsilon)
  - ShadowMap(x',y') + ε > z'
- Scegliere il bias giusto può essere problematico!



Correct image | Not enough bias | Way too much bias

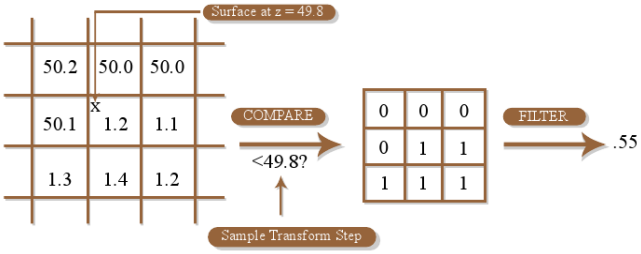
Shadowmap Aliasing

- Densità di campionamento limitata dalla risoluzione del render target
  - Sotto-campionamento della shadow-map
  - Aliasing di riproiezione – in particolare quando luce e punto di vista puntano l'uno verso l'altro



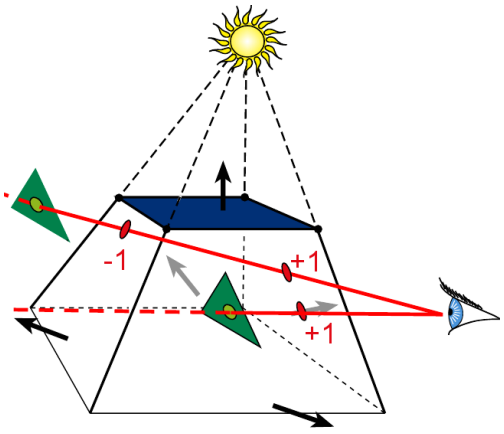
Filtraggio delle shadow map

- Filtrare la densità non ha senso! Meglio mediare le risposte dello z-test



Shadows Volume (Stencil Shadows)

- Rappresentiamo esplicitamente il volume di spazio in ombra
- Per ogni poligono
  - Piramide con vertice sulla sorgente luminosa
  - Aggiungere testa e coda per chiudere
- Shadow test simile al clipping
- Fare partire un raggio dall'osservatore ad un punto visibile
  - Incrementare/decrementare un contatore (stencil) ogni volta che intersechiamo il volume dell'ombra
  - Se il contatore ≠ 0, il punto è in ombra

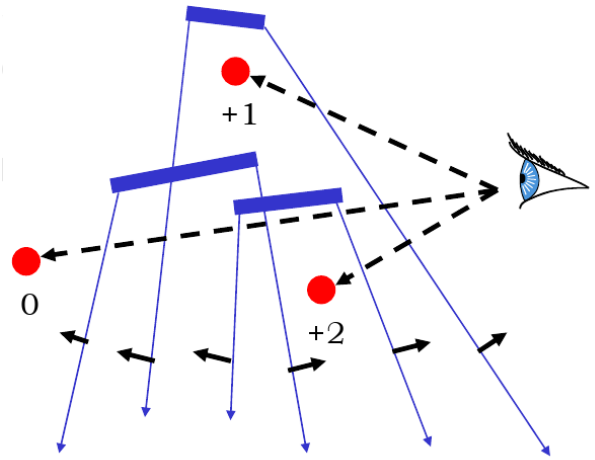


Stencil Buffer

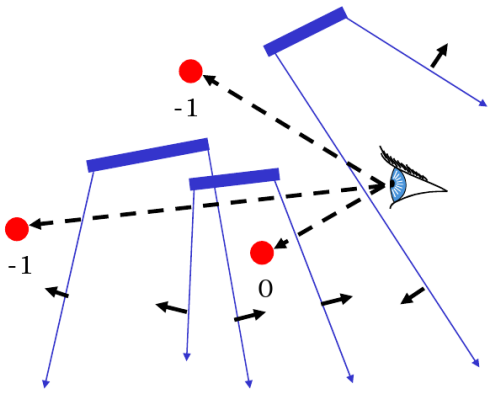
- Buffer intero della dimensione dello schermo usato per “taggare” i pixel per varie operazioni
- Si possono specificare differenti operazioni di rendering per ciascuno dei seguenti test
  - stencil test fallisce
  - stencil test passa & depth test fallisce
  - stencil test passa & depth test passa

Stencil Shadows

- Inizializza stencil buffer a 0
- Disegna la scena con solo la luce ambientale
- Escludi l'update del frame-buffer e dello z-buffer
- Disegna i poligoni d'ombra rivolti verso l'osservatore (front-facing)
  - Se z-pass incrementa il contatore
- Disegna i poligoni d'ombra rivolti dalla parte opposta rispetto all'osservatore (back-facing)
  - Se z-pass decrementa il contatore
- Riattiva l'update del frame-buffer e l'illuminazione
- Ridisegna i pixel con contatore = 0

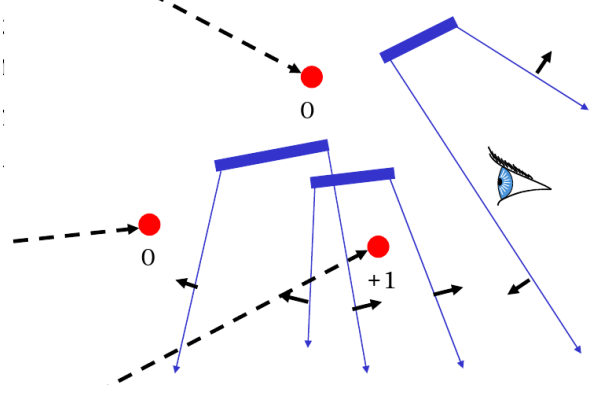


- Osservatore in ombra
- Se l'osservatore è in ombra contatore=0 non vuol dire illuminato!
- Soluzioni
  - Testare l'osservatore rispetto alla luce e inizializzare lo stencil-buffer di conseguenza
    - Costoso!
  - Variante Z-fail



Z-fail Shadow Volumes

- Partire dall'infinito
- Disegnare ombre front-facing
  - Se z-fail, decrementare il contatore
- Disegnare ombre back-facing
  - Se z-fail, incrementare il contatore
- Introduce problemi con la far clipping plane
  - Risolte limitando la profondità durante il clipping
- Richiede che il volume sia chiuso con
  - disegnare testa e coda degli shadow volumes



Shadow Volume: Limiti e problemi

- Introduce un sacco di nuova geometria
- Produce triangoli lunghi e stretti difficili da rasterizzare
- Precisione limitata dello stencil buffer può portare a overflow con scene molto complesse
- La rasterizzazione di poligoni che condividono un lato non devono avere overlap ne gap

Scelta dell'algoritmo per le ombre

- Shadow buffer
  - Precomputato
  - molto efficiente per parti fisse
- Stencil shadow
  - calcolato al volo
  - ideale per componenti animate (ma non solo)