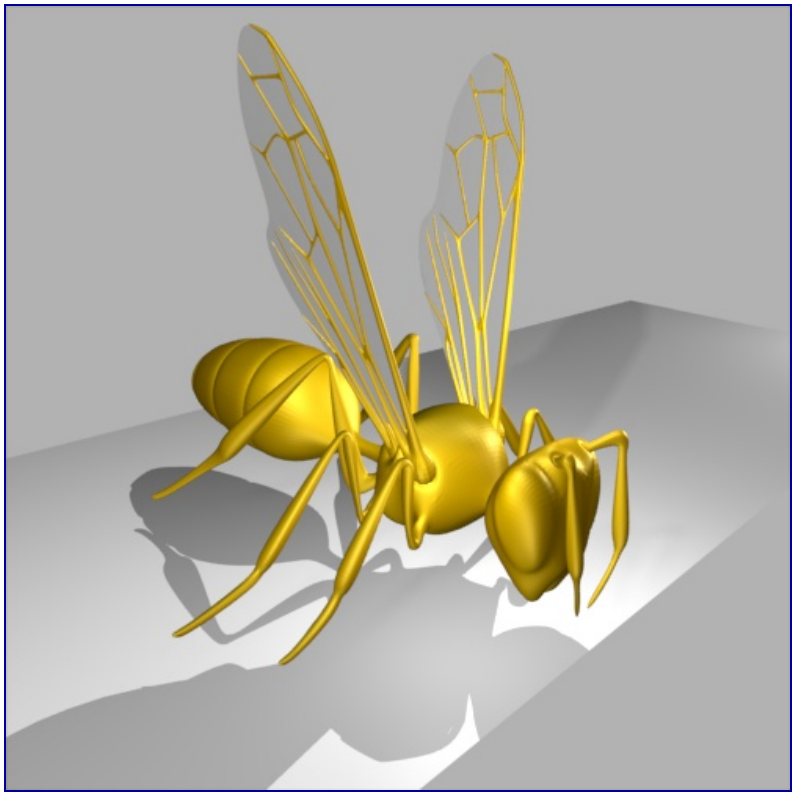
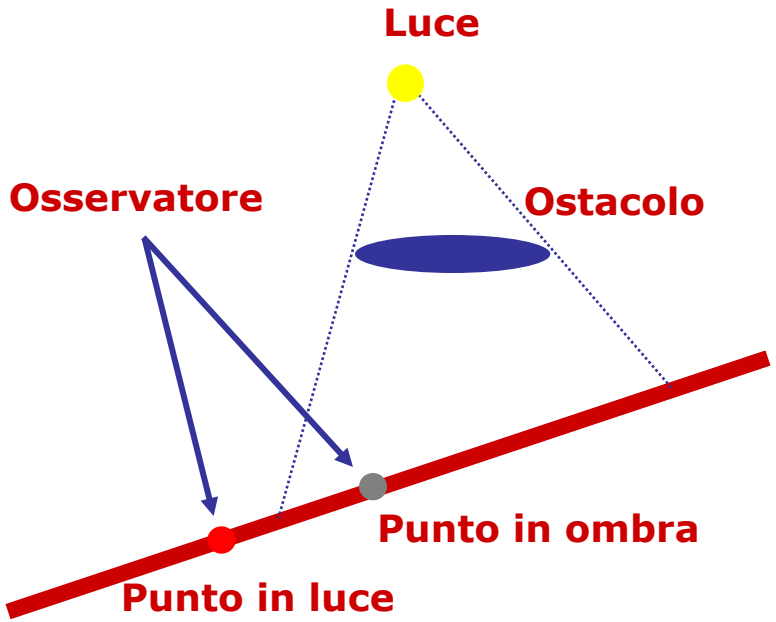
Shadowing



L'Ombra

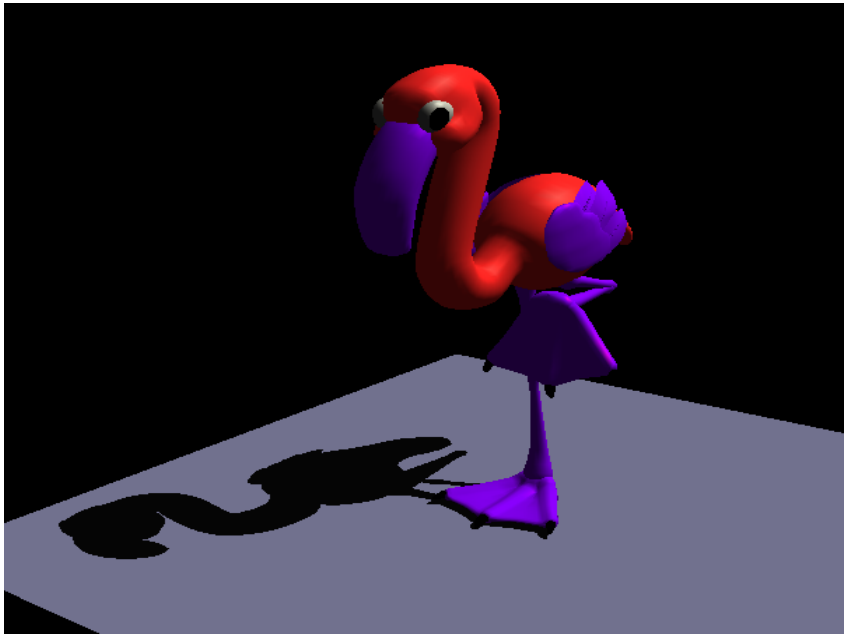
Un punto è in ombra se la luce non riesce a raggiungerlo perché bloccata da un ostacolo.



Per disegnare un pixel con "colore ombra" bisogna aver memorizzato l'informazione se un pixel è illuminato o meno.

Funzioni dell'ombra

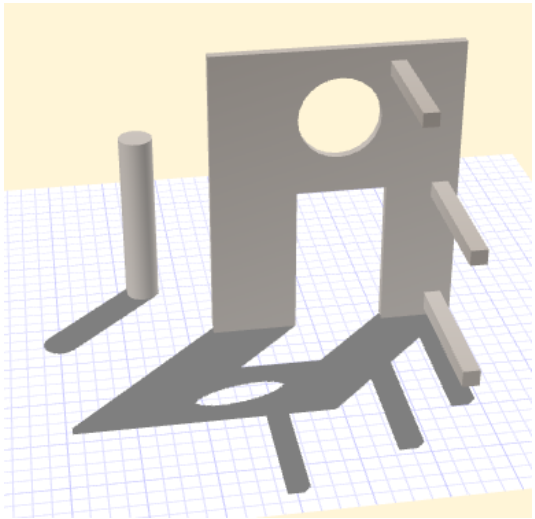
- Informa l'osservatore se l'oggetto è appoggiato su una superficie o meno;
- Enfatizza la posizione della sorgente luminosa;

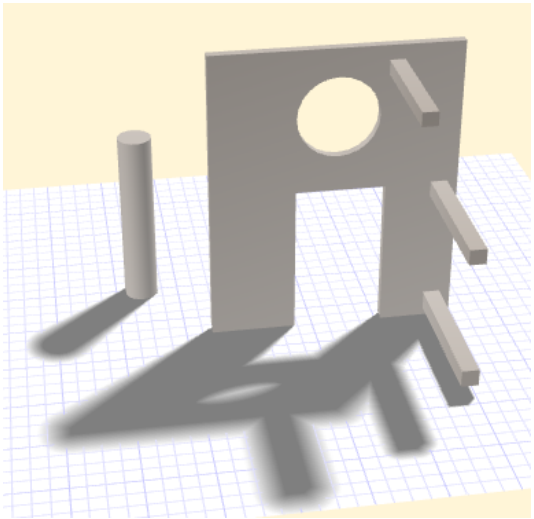


Ombra e Penombra

Le ombre variano moltissimo a seconda dell'ambiente luminoso. Possono avere contorni netti o contorni sfumati e contenere sia zone di ombra che di penombra

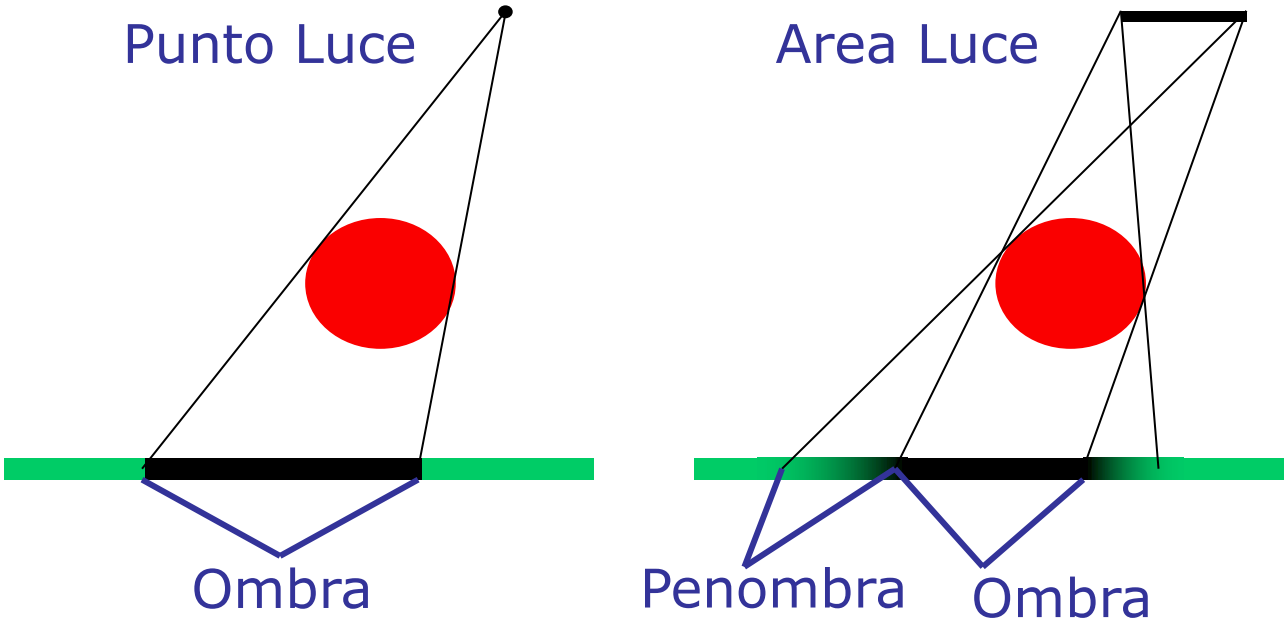
La dimensione relativa dell'ombra/penombra è una funzione della dimensione e forma della sorgente luminosa e della distanza dall'oggetto.





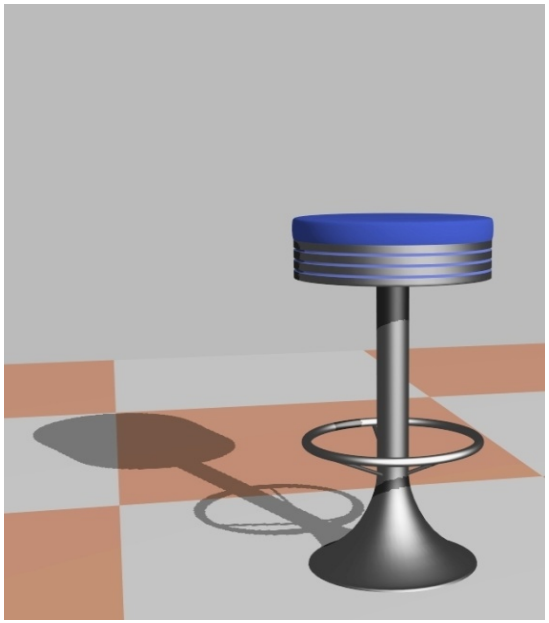
Ombra e Penombra

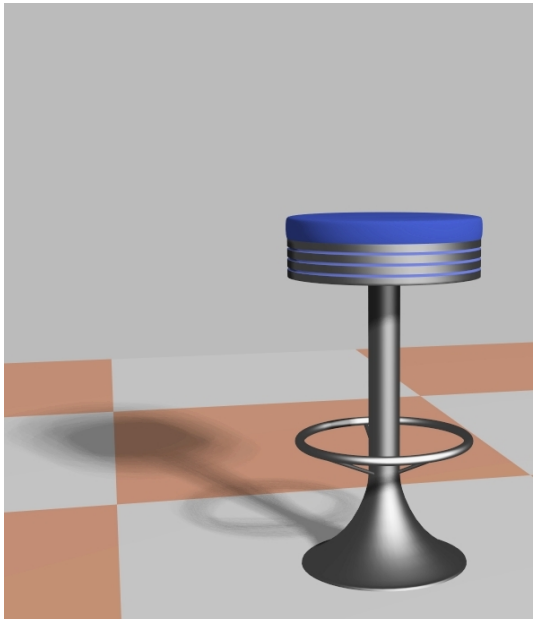
L'ombra è quella parte completamente non illuminata dalla sorgente, mentre la penombra è un'area che riceve una certa luce dalla sorgente; la penombra si sovrappone all'ombra e c'è sempre un graduale passaggio di intensità dall'una all'altra.



Ombra e Penombra

In Computer Graphics, solitamente si considera una sorgente luminosa puntiforme ad una certa distanza (o all'infinito) e ci si limita al caso più semplice di oggetti che producono un'ombra netta (hard shadow).





Anche in questa situazione di sorgente puntiforme, causa effetto diffrazione della luce dietro l'oggetto, l'ombra non dovrebbe avere dei contorni netti, ma graduali (soft shadow).

Osservazioni

- Se l'osservatore e la sorgente luminosa coincidono come posizione, non ci sono ombre;
- Per scene statiche, le ombre sono fisse e non cambiano al cambiare della posizione dell'osservatore;
- Se la posizione relativa dell'oggetto e della sorgente cambiano, le ombre devono essere ricalcolate. Questo porta ad un alto costo del rendering dove le ombre sono importanti per percepire la profondità e il movimento.
- Molti algoritmi di shadowing si occupano solo di modelli poligonali e questi producono ombre poligonali (con contorni poligonali).

Algoritmi di Shadowing

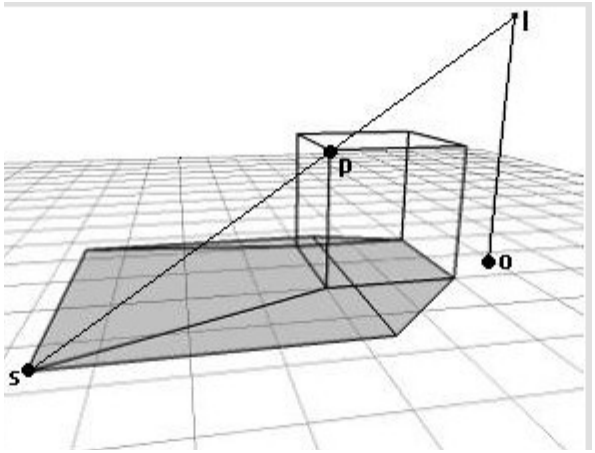
- Ombre su piani (proiezione)
- Shadow Buffer/Map
- Shadow Volume (non lo vedremo)

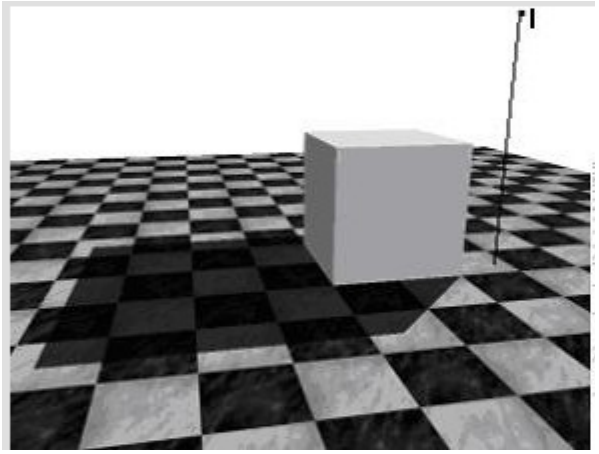
Ombre su piani (Blinn 1988)

Obiettivo: generare una geometria piatta per proiezione sul piano, quindi disegnarla con colore ombra.

Algoritmo: Proiezione di un oggetto su un piano (pavimento/parete)

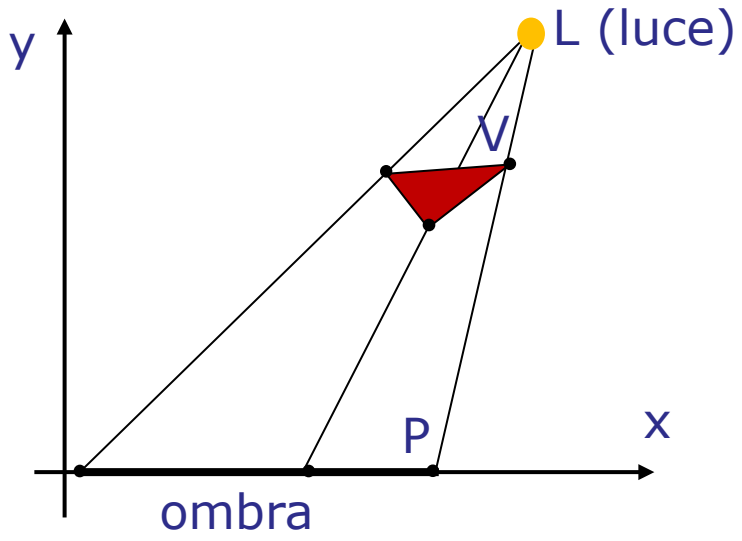
- Si costruisce la matrice di proiezione dalla luce al piano
- Si applica tale matrice alla geometria 3D di cui si vuole ottenere l'ombra
- Si disegna l'oggetto piatto con colore ombra





Ombre su piani

Caso speciale: l'ombra deve essere generata su un piano cartesiano; basta proiettare i vertici su quel piano e considerare l'oggetto piatto formato dalle facce proiettate.

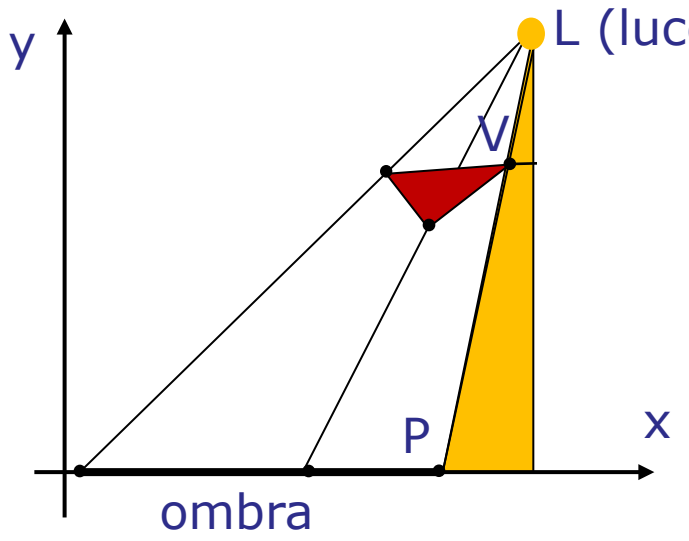


Dati:
- posizione della luce: L
- posizione del piano: y=0
- posizione del vertice: V

Calcolare la proiezione P di V sul piano.

Ombre su piani

Usiamo la similitudine fra triangoli per determinare P:



$$(Lx - Px) : (Lx - Vx) = Ly : (Ly - Vy)$$

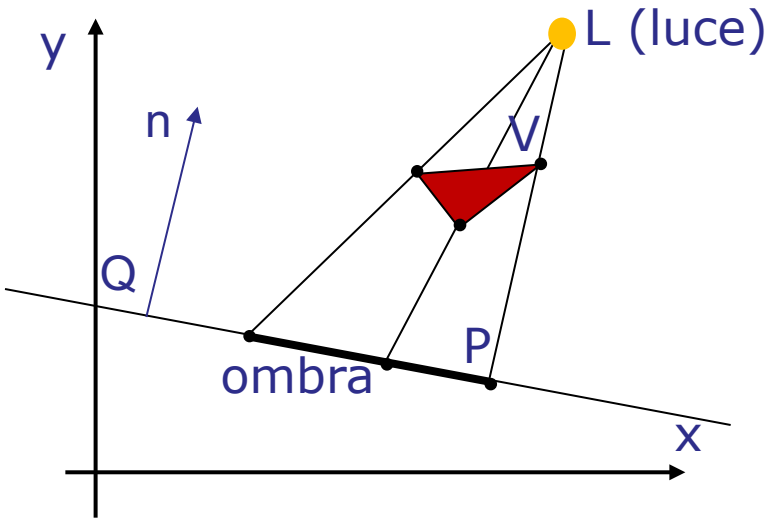
$$\frac{(Px - Lx)}{(Vx - Lx)} = \frac{Ly}{(Ly - Vy)}$$

$$Px - Lx = \frac{Ly\,(Vx - Lx)}{(Ly - Vy)}$$

$$Px = \frac{Ly\,Vx - LxVy}{(Ly - Vy)}$$

Ombre su piani

Caso generale: l'ombra deve essere generata su un piano arbitrario; bisogna proiettare i vertici su quel piano e considerare l'oggetto piatto formato dalle facce proiettate.

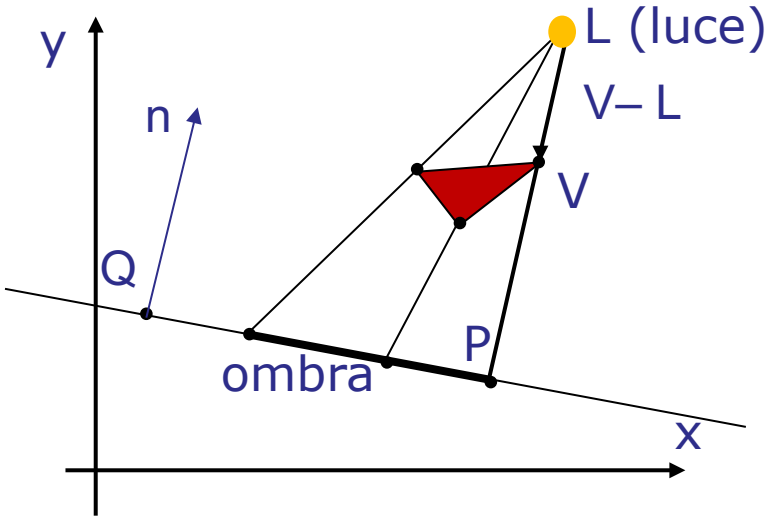


Dati:
- posizione della luce: L
- un punto sul piano: Q=(qx,qy,qz)
- normale al piano: n=(a,b,c)
- posizione del vertice: V

Equazione del piano: ax+by+cz+d=0
con d = – a qx – b qy – c qz

Calcolare la proiezione P di V sul piano arbitrario

Ombre su piani



Dati:
- posizione della luce: L
- un punto sul piano: Q=(qx,qy,qz)
- normale al piano: n=(a,b,c)
- posizione del vertice: V

Equazione del piano: ax+by+cz+d=0
con d = – a qx – b qy – c qz

Sarà: $P = L + scal\ (V - L)$

Sappiamo che: $n \cdot P + d = 0$, perché P deve stare sul piano; allora possiamo risolvere per trovare il valore dello scalare *scal*

$n \cdot (L + scal\ (V - L)) + d = 0$

$n \cdot L + scal\ n \cdot (V - L)) + d = 0$

$$scal = -\frac{d + n \cdot L}{n \cdot (V - L)}$$

Ombre su piani in WebGL

Luce: $L = (Lx, Ly, Lz, Lw)$ in coord. omogenee
Normale al piano: $n = (a,b,c)$
Punto sul Piano: $Q = (qx, qy, qz)$

$$d = - a\ qx - b\ qy - c\ qz$$

$$\text{Sia } \ dot = a\ Lx + b\ Ly + c\ Lz + d\ Lw$$

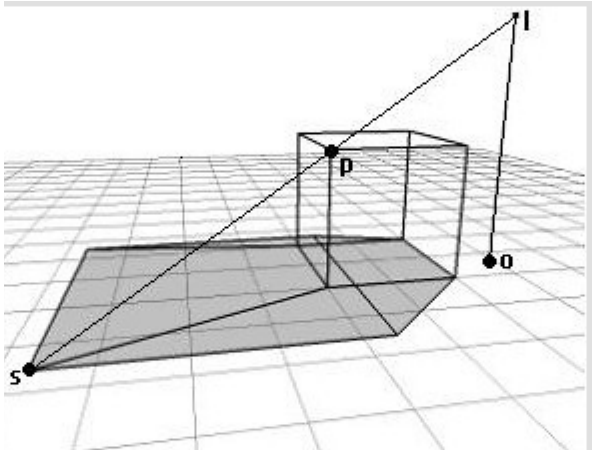
$$M = dot\ I_{4x4} - \begin{pmatrix} a\ Lx & a\ Ly & a\ Lz & a\ Lw \\ b\ Lx & b\ Ly & b\ Lz & b\ Lw \\ c\ Lx & c\ Ly & c\ Lz & c\ Lw \\ d\ Lx & d\ Ly & d\ Lz & d\ Lw \end{pmatrix}$$

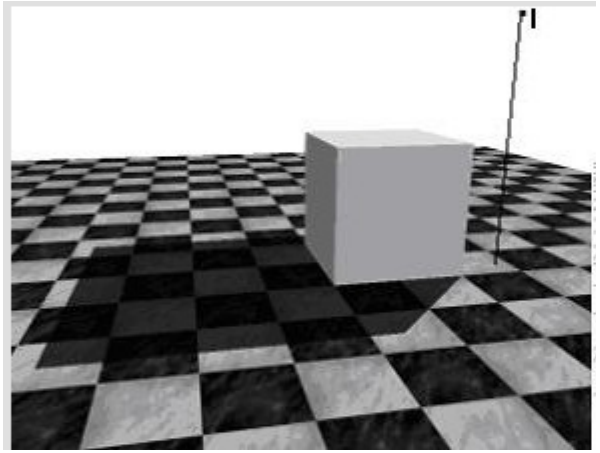
$$P = M\ V$$

Ombre su piani

Vantaggi e Svantaggi:

- Veloce e semplice
- Funziona solo per superfici piane
- Non funziona per auto-ombre





Ombre su piani

HTML5_webgl_2/cube_shadows.html e .js

Ombre nello Z-buffer

Il più semplice algoritmo per le ombre, facilmente integrabile in uno Z-Buffer, è lo Shadow Buffer sviluppato da Williams nel 1978.

Questo metodo richiede un buffer per le ombre, diverso dallo Z-Buffer, per ogni sorgente luminosa.

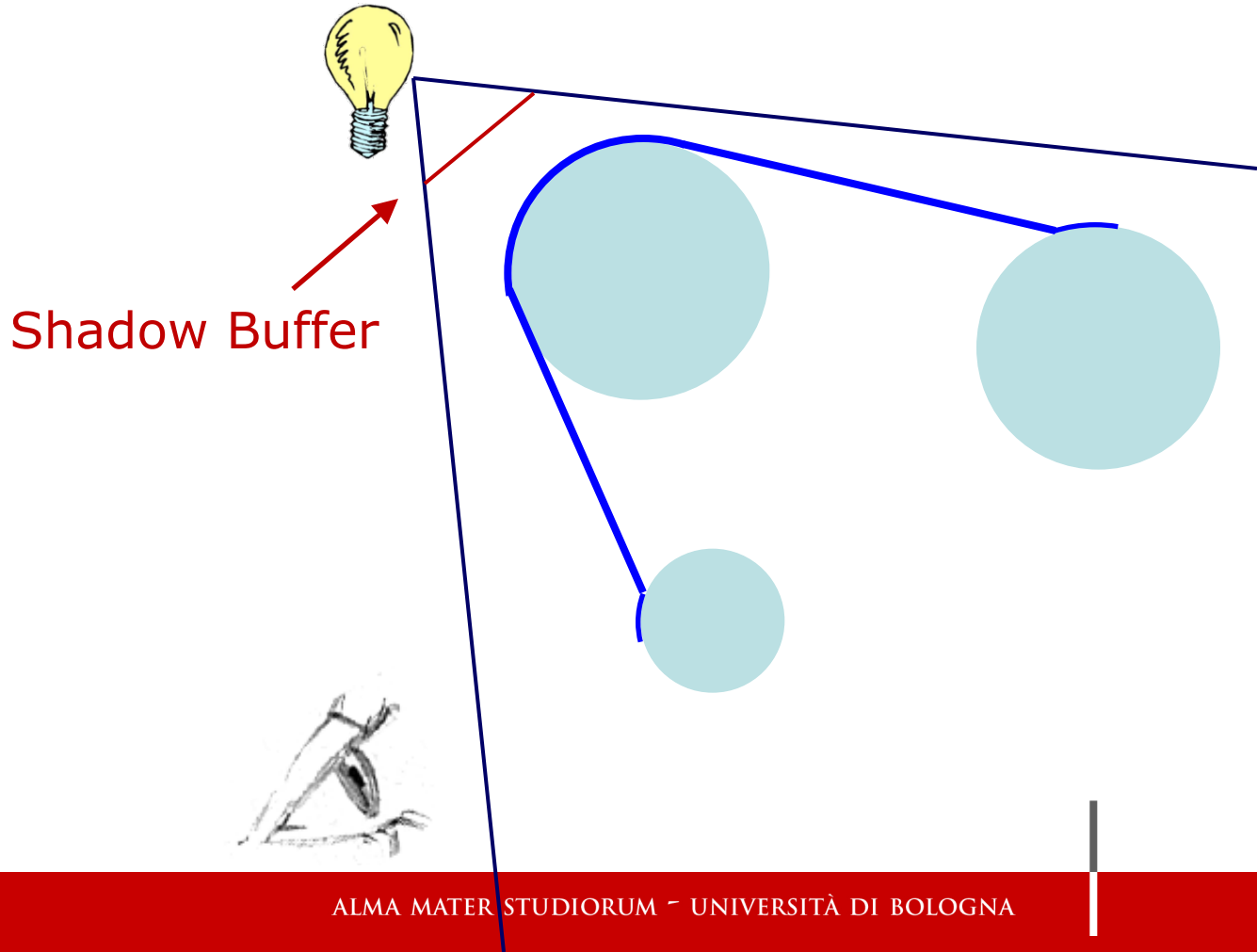
Nella sua forma base, è efficiente solo per scene illuminate da una sola sorgente puntiforme. In alternativa, si può usare un solo buffer per le ombre anche per più luci, ma l'algoritmo deve essere eseguito per ogni luce, e questo lo rende inefficiente e lento.

L'algoritmo consiste in un processo a due passi.

Shadow Buffer

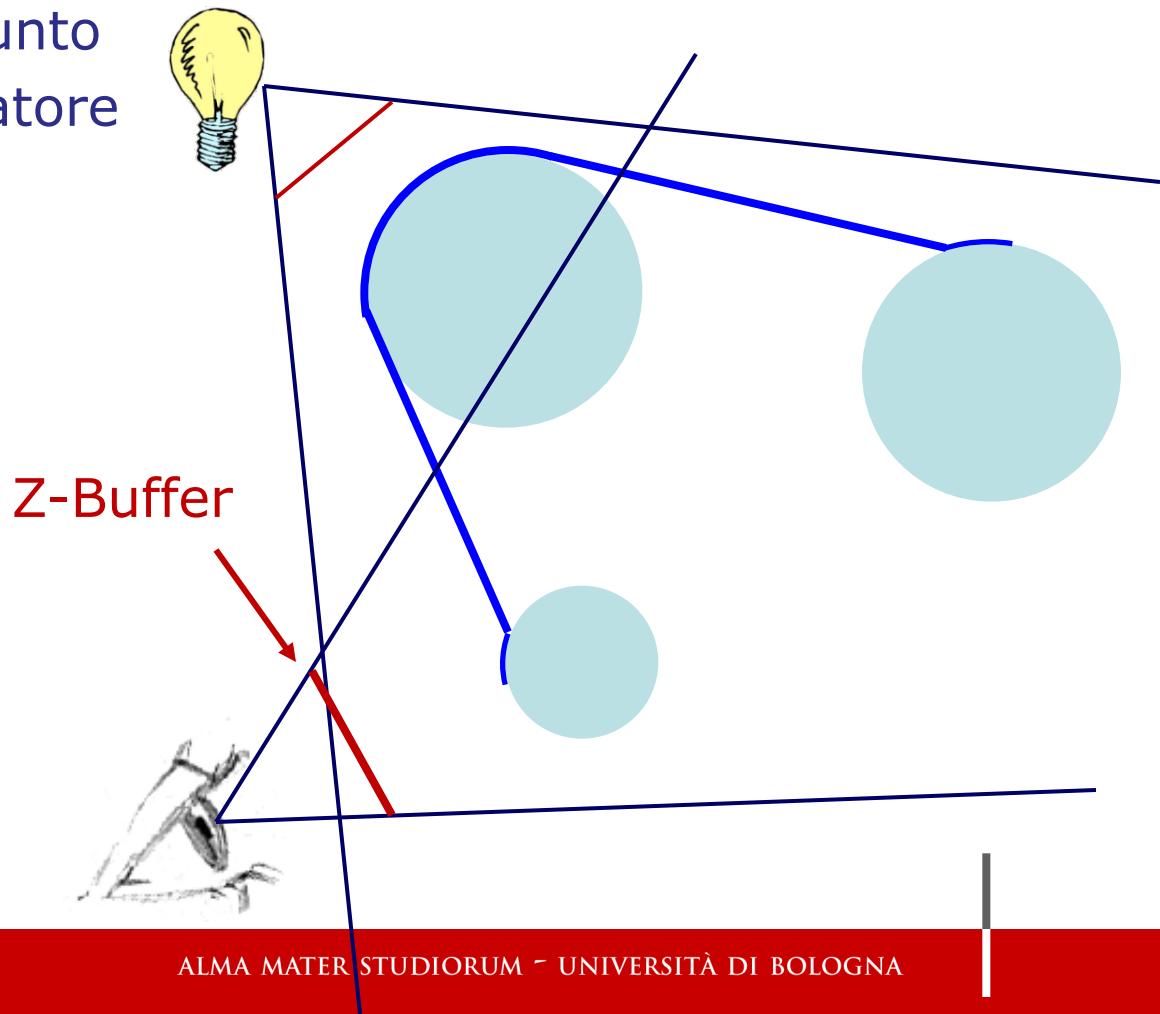
Shadow Buffer

- Rende la scena dal punto di vista della sorgente luminosa



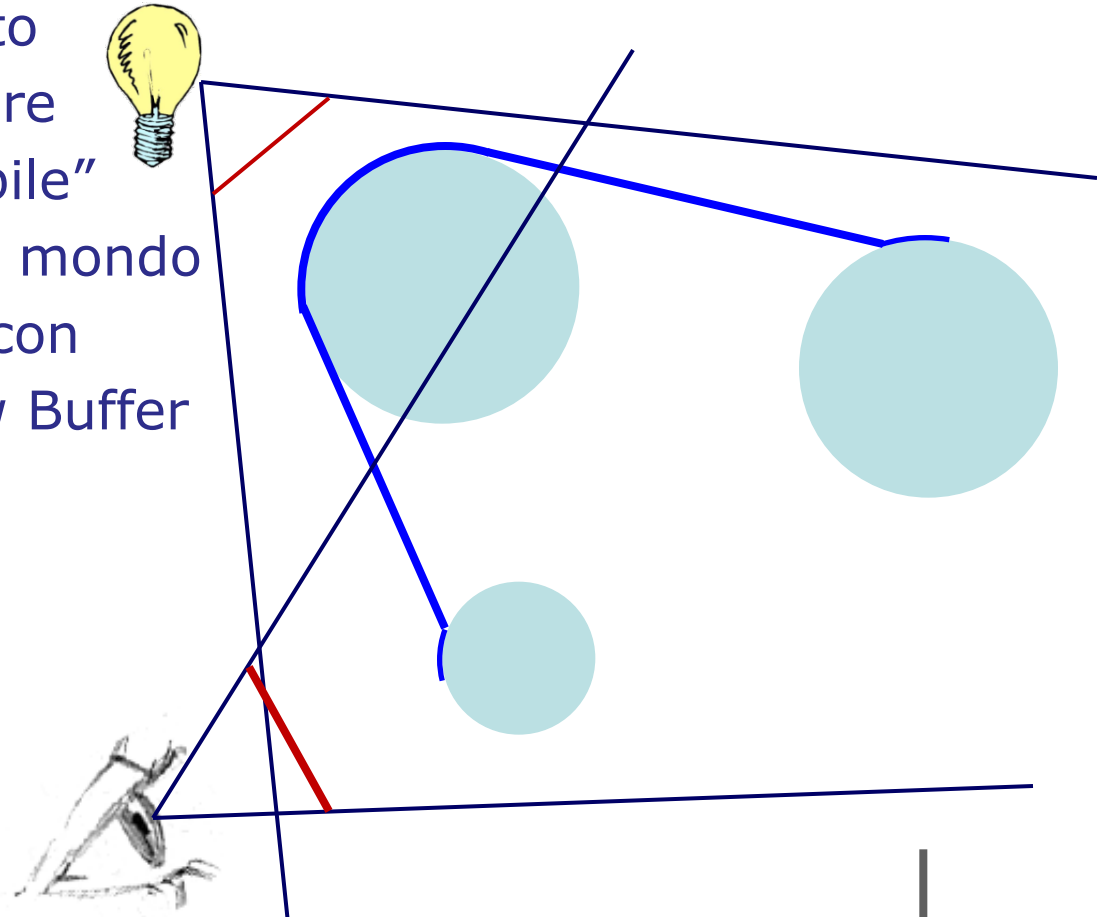
Shadow Buffer

- Rende la scena dal punto di vista della sorgente luminosa
- Rende la scena dal punto di vista dell'osservatore



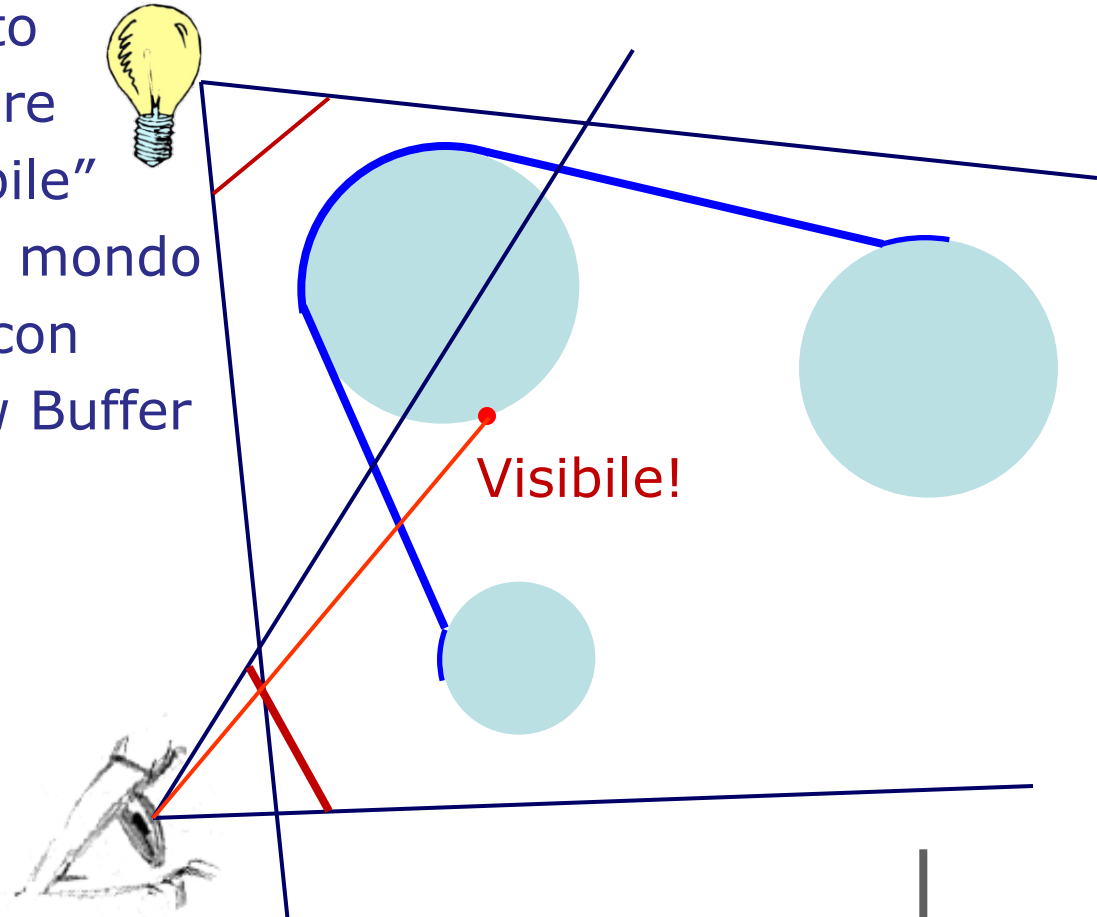
Shadow Buffer

- Rende la scena dal punto di vista della sorgente luminosa
- Rende la scena dal punto
  di vista dell'osservatore
- Per ogni fragment "visibile"
  - trasforma nello spazio mondo
  - confronta la distanza con
    il valore nello Shadow Buffer



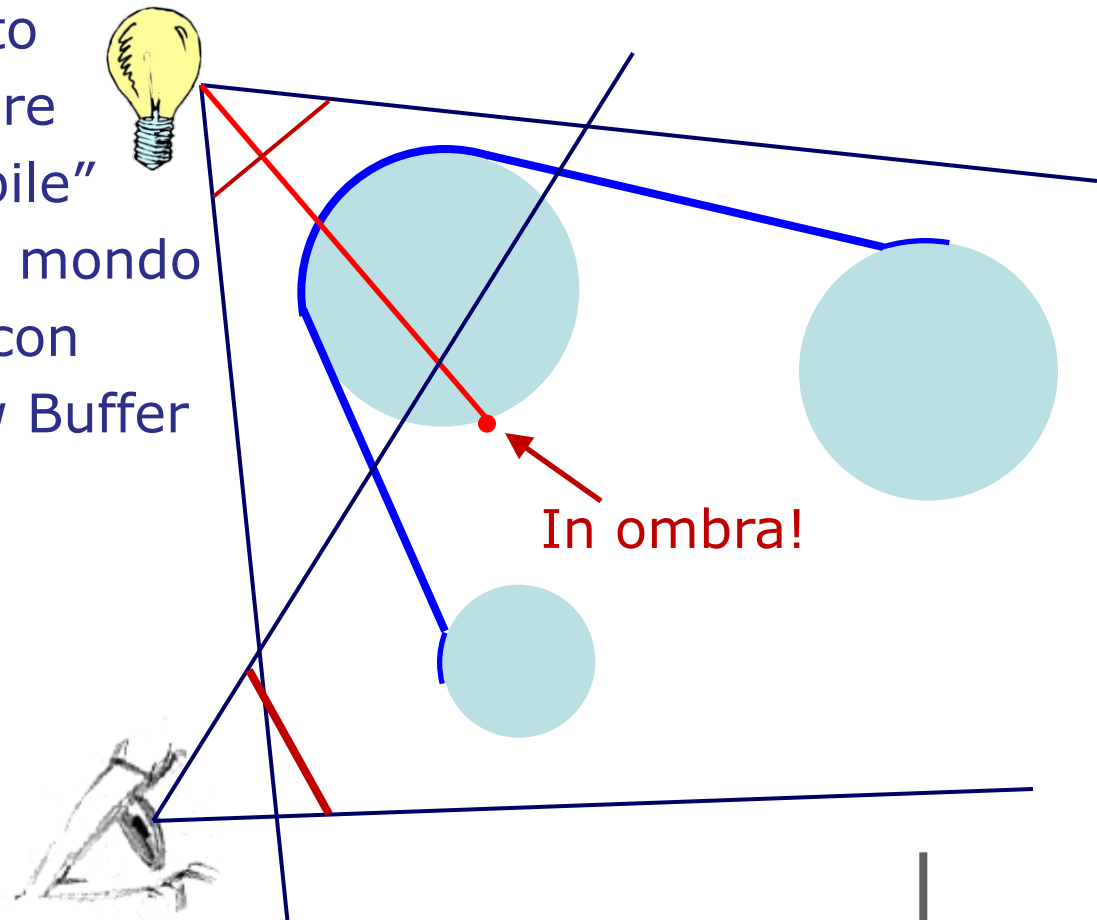
Shadow Buffer

- Rende la scena dal punto di vista della sorgente luminosa
- Rende la scena dal punto
  di vista dell'osservatore
- Per ogni fragment "visibile"
  - trasforma nello spazio mondo
  - confronta la distanza con
    il valore nello Shadow Buffer



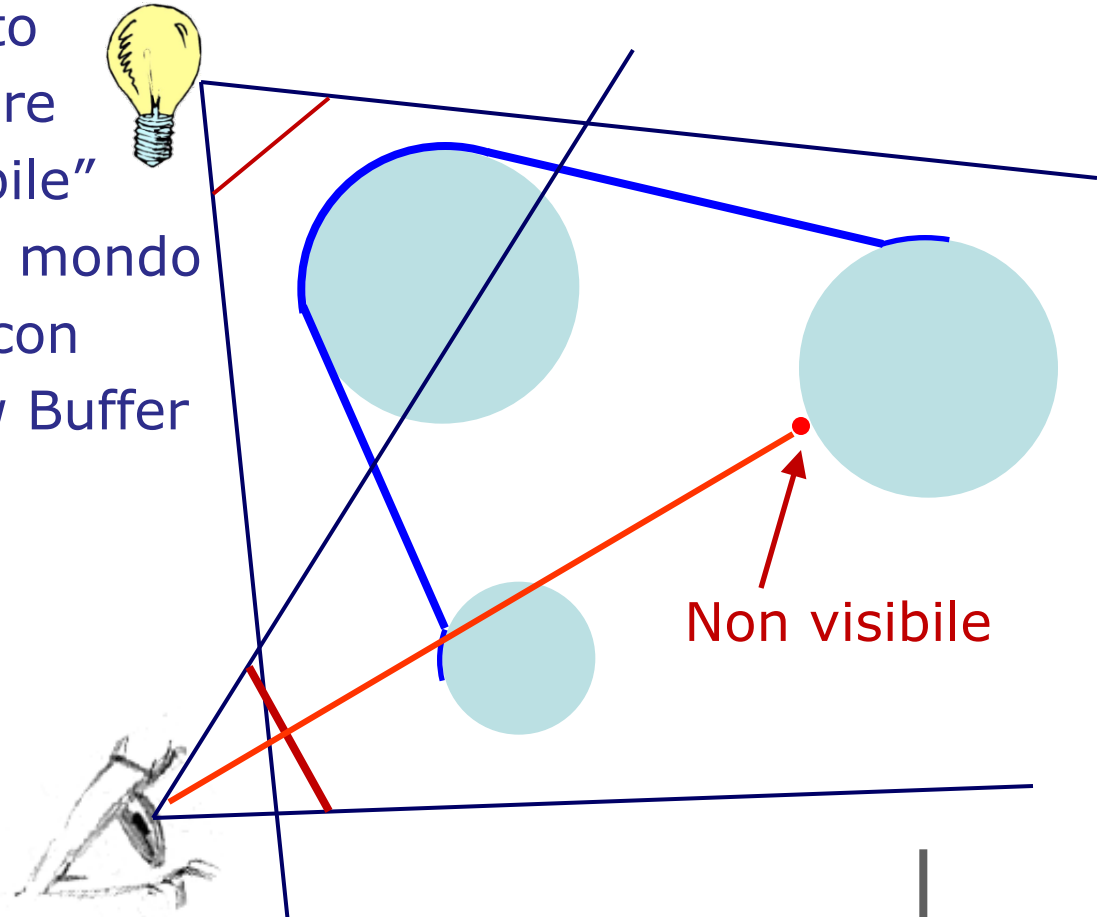
Shadow Buffer

- Rende la scena dal punto di vista della sorgente luminosa
- Rende la scena dal punto
  di vista dell'osservatore
- Per ogni fragment "visibile"
  - trasforma nello spazio mondo
  - confronta la distanza con
    il valore nello Shadow Buffer



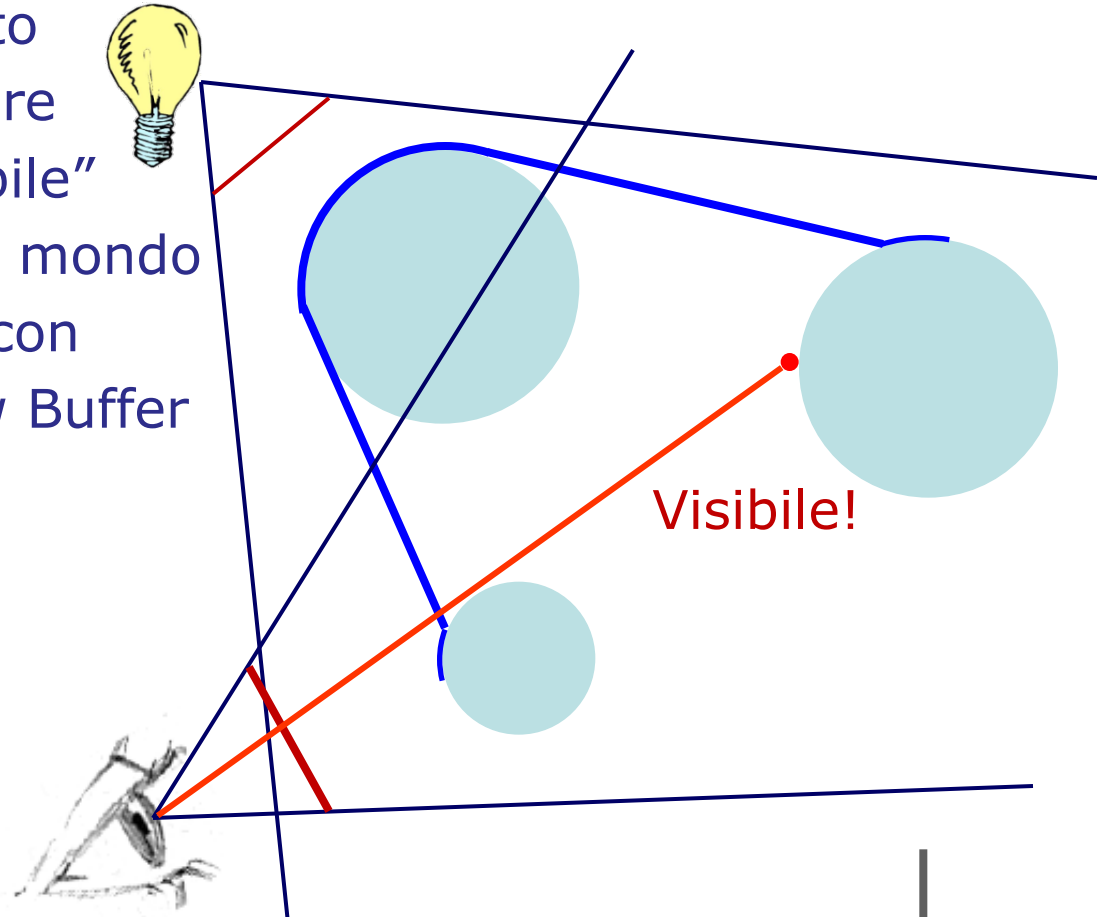
Shadow Buffer

- Rende la scena dal punto di vista della sorgente luminosa
- Rende la scena dal punto
  di vista dell'osservatore
- Per ogni fragment "visibile"
  - trasforma nello spazio mondo
  - confronta la distanza con
    il valore nello Shadow Buffer



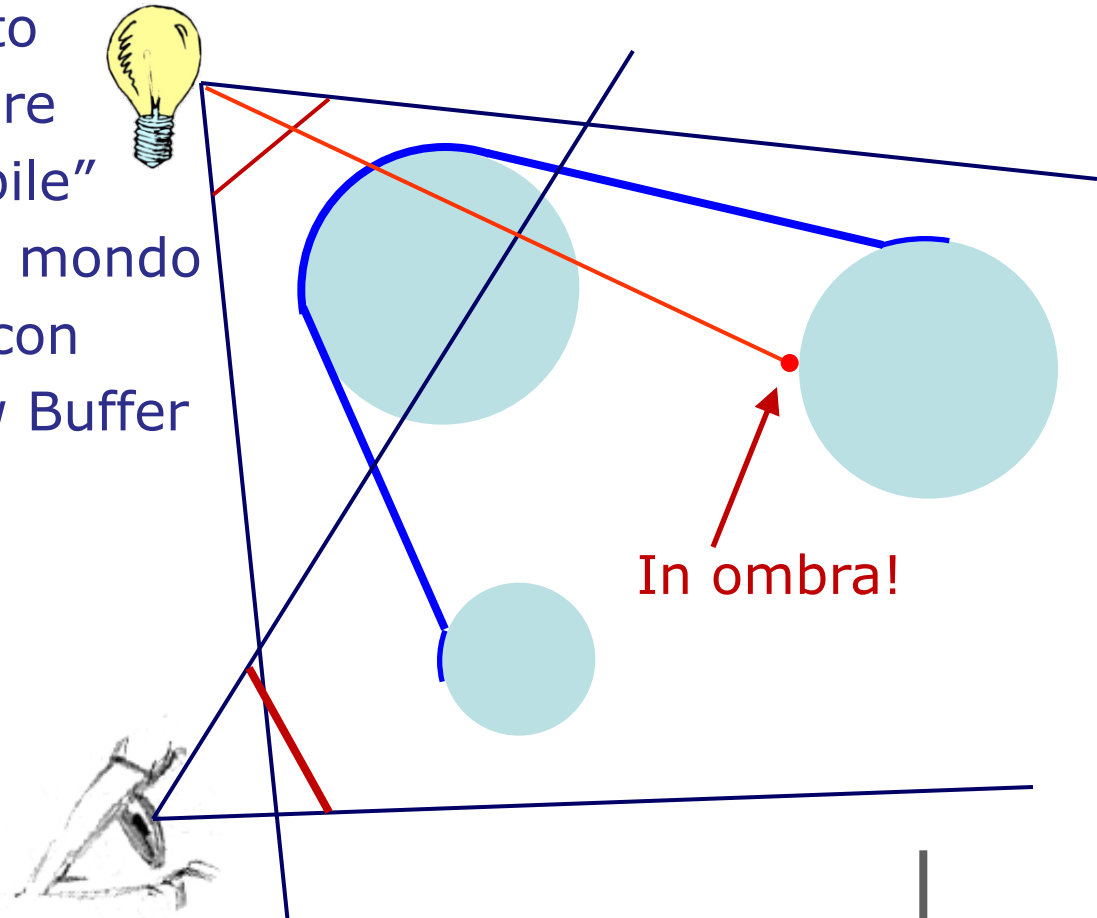
Shadow Buffer

- Rende la scena dal punto di vista della sorgente luminosa
- Rende la scena dal punto
  di vista dell'osservatore
- Per ogni fragment "visibile"
  - trasforma nello spazio mondo
  - confronta la distanza con
    il valore nello Shadow Buffer



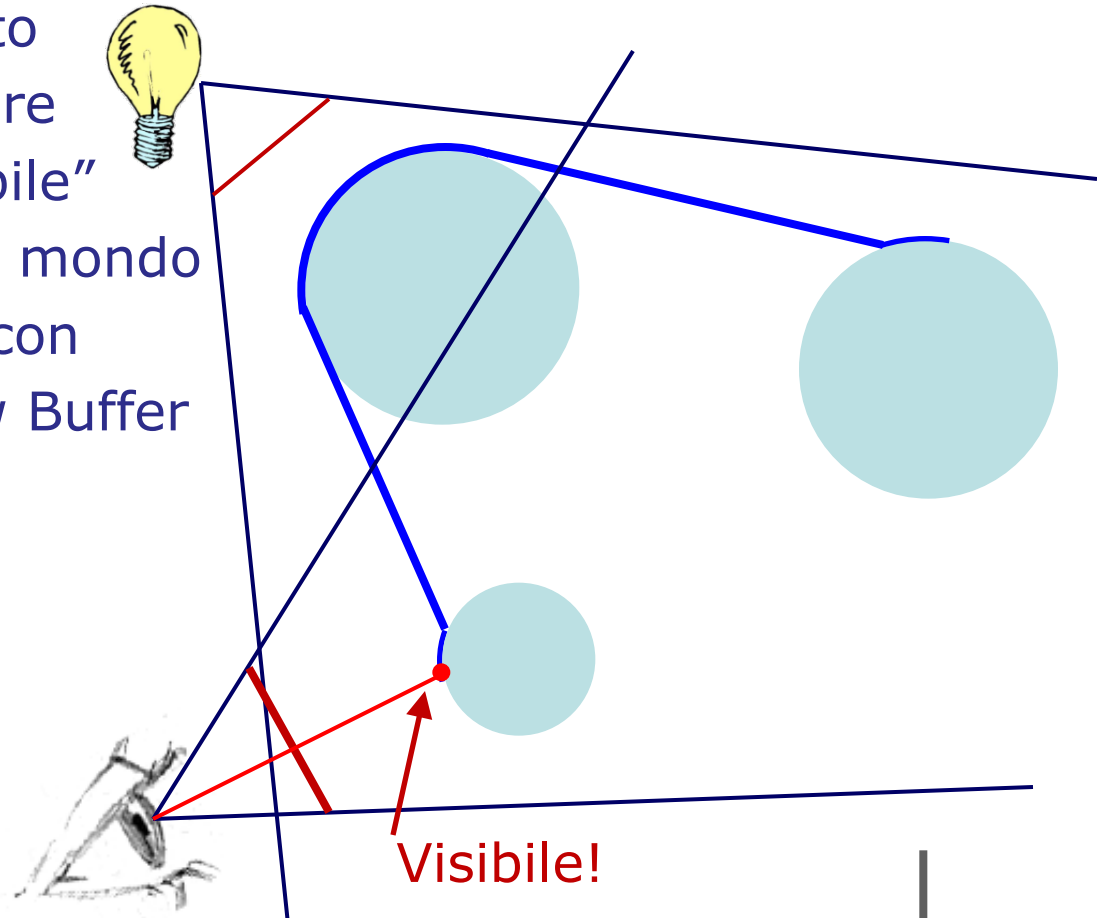
Shadow Buffer

- Rende la scena dal punto di vista della sorgente luminosa
- Rende la scena dal punto
  di vista dell'osservatore
- Per ogni fragment "visibile"
  - trasforma nello spazio mondo
  - confronta la distanza con
    il valore nello Shadow Buffer



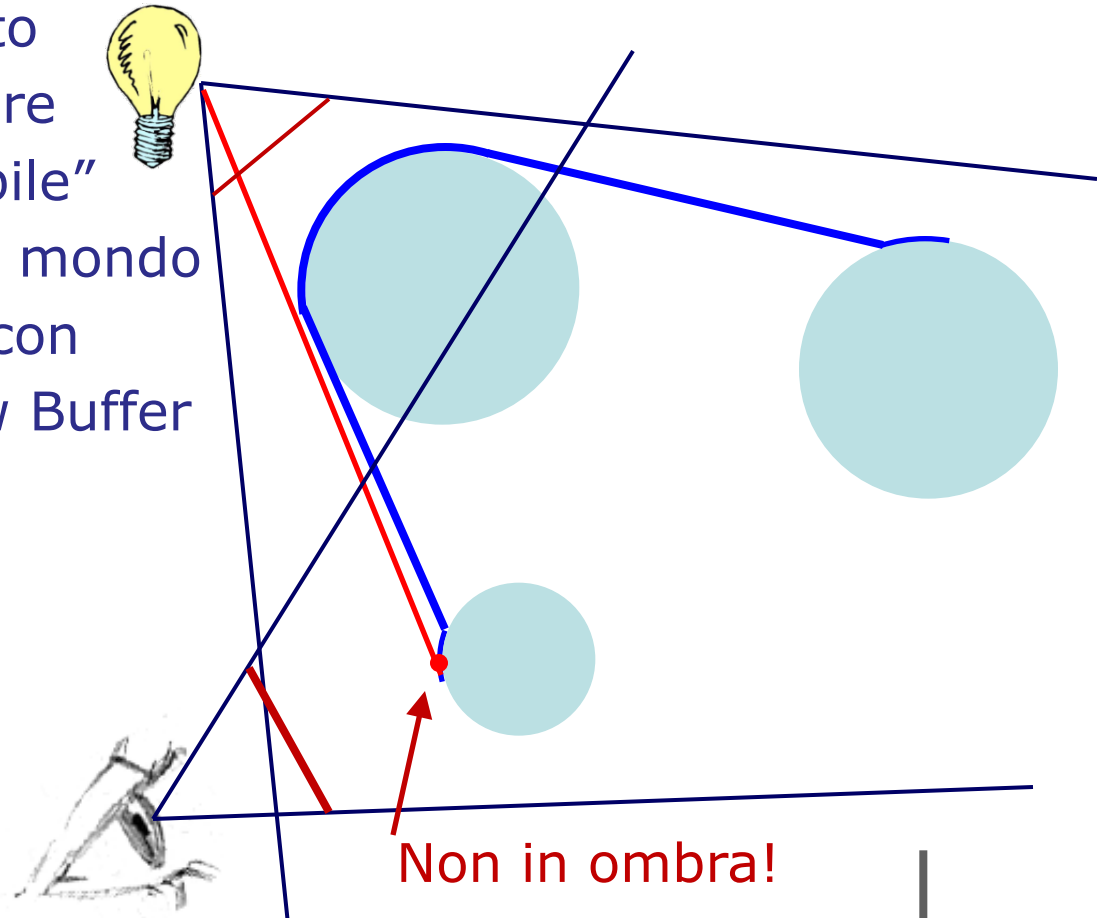
Shadow Buffer

- Rende la scena dal punto di vista della sorgente luminosa
- Rende la scena dal punto
  di vista dell'osservatore
- Per ogni fragment "visibile"
  - trasforma nello spazio mondo
  - confronta la distanza con
    il valore nello Shadow Buffer



Shadow Buffer

- Rende la scena dal punto di vista della sorgente luminosa
- Rende la scena dal punto
  di vista dell'osservatore
- Per ogni fragment "visibile"
  - trasforma nello spazio mondo
  - confronta la distanza con
    il valore nello Shadow Buffer



Shadow Buffer

1.Si considera come punto di vista la sorgente luminosa e si applica un classico depth-buffer, ma non si "renderizza" nulla e le profondità vengono memorizzate nello Shadow Buffer;

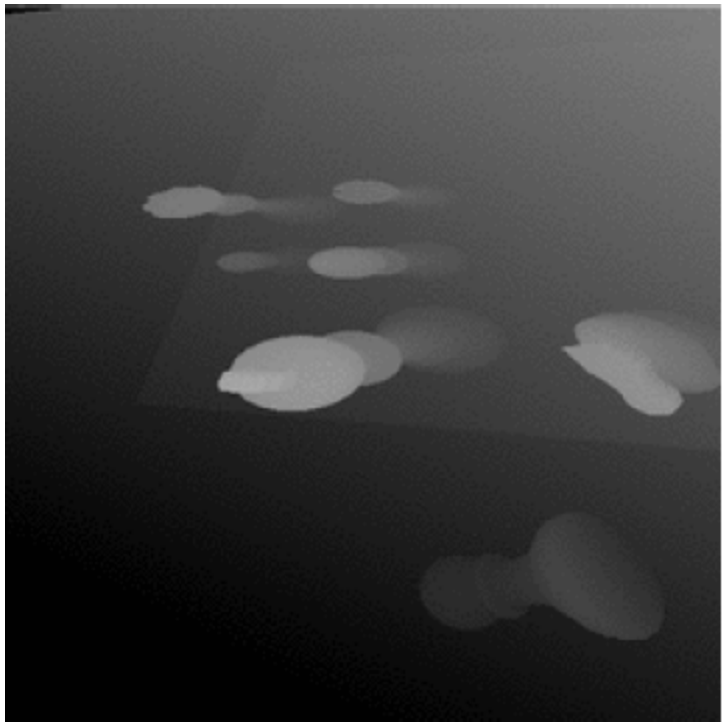
2.Si applica lo Z-Buffer per rendere la scena dal punto di vista dell'osservatore, modificato come segue:

- se un fragment è visibile, si applica una trasformazione per mappare le coordinate [x,y,z] del punto nello spazio schermo (dal punto di vista) in [x',y',z'] del punto nello spazio schermo (dalla sorgente luminosa).
- sia zs il valore nello Shadow Buffer nella posizione [x',y'] ;
- se z'>zs il punto sarà in ombra e sarà reso con intensità di ombra, altrimenti il punto verrà reso normalmente.

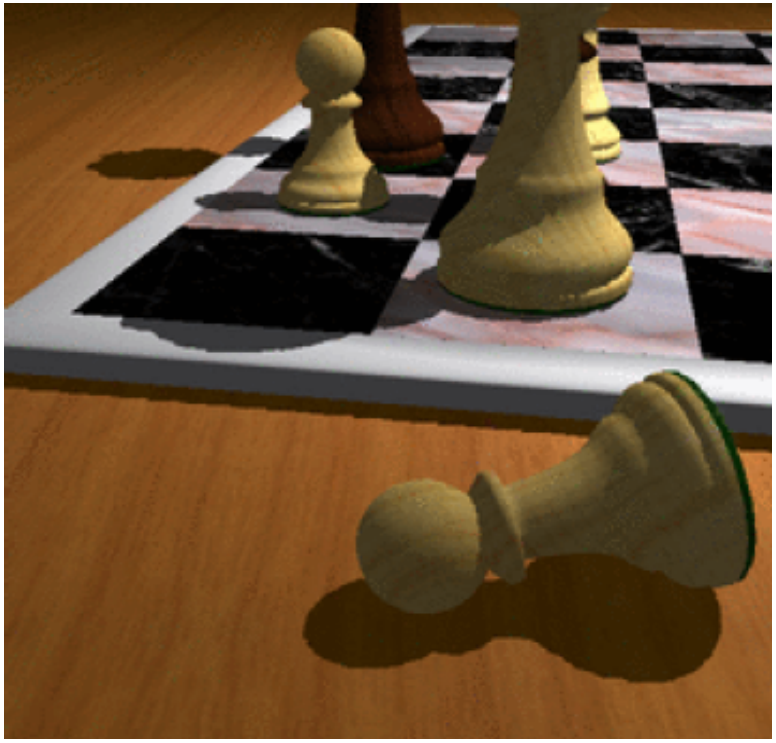
Shadow Buffer: Algoritmo

- Rendere la scena come se fosse vista dalla sorgente luminosa
- Salvare il depth dei pixel (2D Shadow Buffer)
- Rendere la scena dalla posizione dell'osservatore
  - Trasformare le coordinate del pixel nel sistema di coordinate della luce
  - Confrontare z' con il valore zs memorizzato nella corrispondente posizione nello Shadow Buffer
    - Il pixel è in ombra se $z' > zs$

Shadow Buffer: Esempio

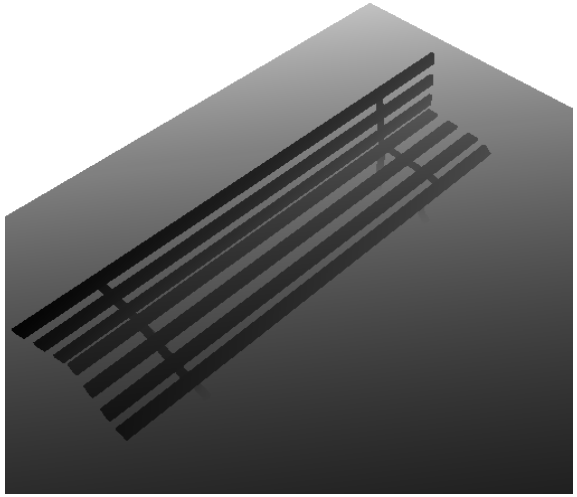


Shadow map

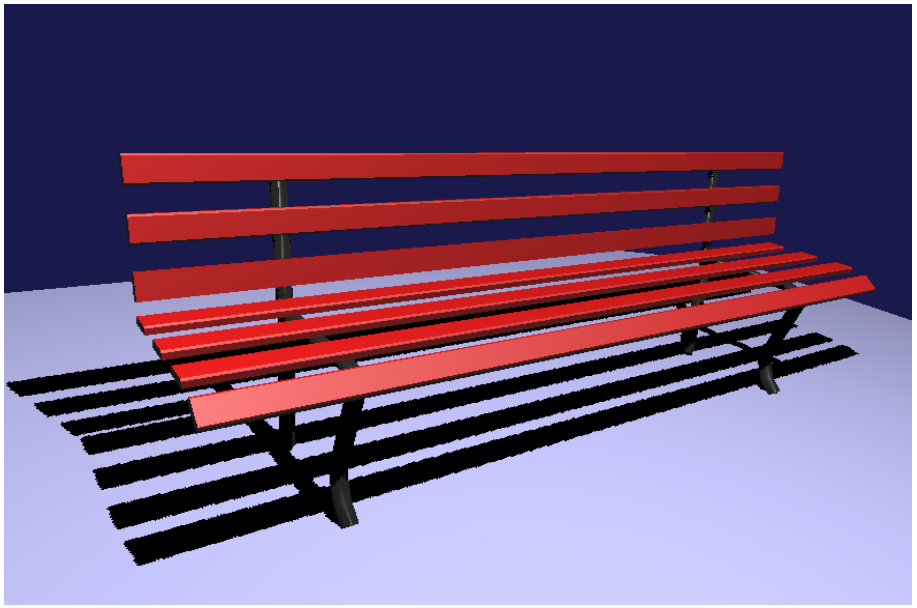


Scena con una luce

Shadow Buffer: Esempio

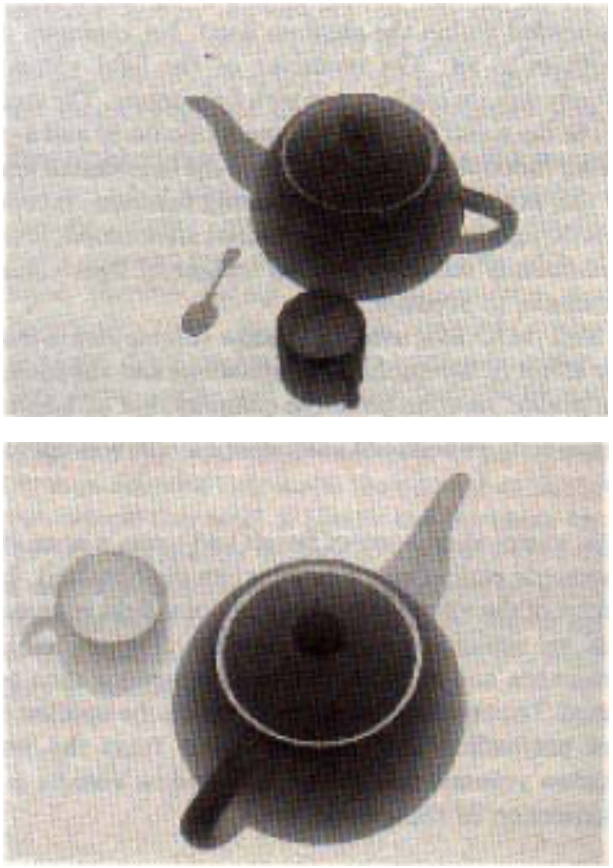


Shadow map



Scena con una luce

Shadow Buffer: Esempio



Shadow map

Scena con due luci

Shadow Buffer

- Vantaggi
  - Semplice da implementare
  - Non dipende dalla complessità della scena (tranne che per rendere lo Shadow Buffer)
- Svantaggi
  - Immagini a risoluzione fissa portano ad artefatti

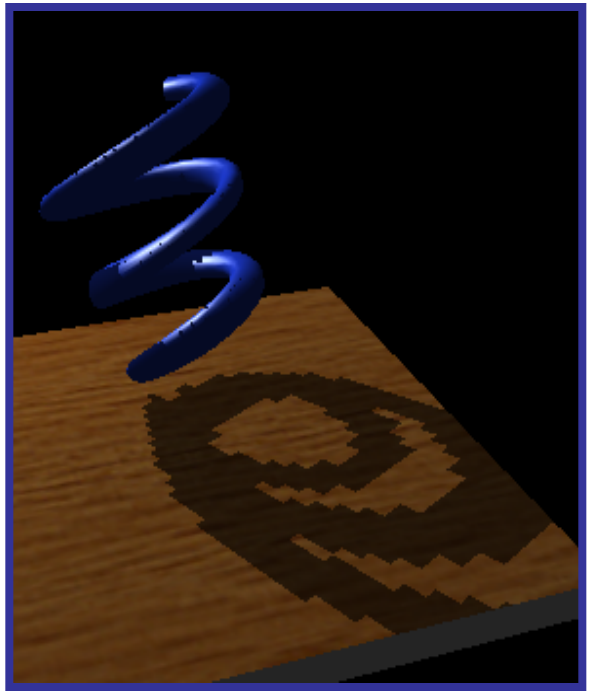
Aliasing nello Z-Buffer con ombre mediante Shadow-Buffer

Con aliasing ci si riferisce ad un effetto che comporta che differenti segnali, quando vengono campionati, siano indistinguibili (da alias uno al posto dell'altro).

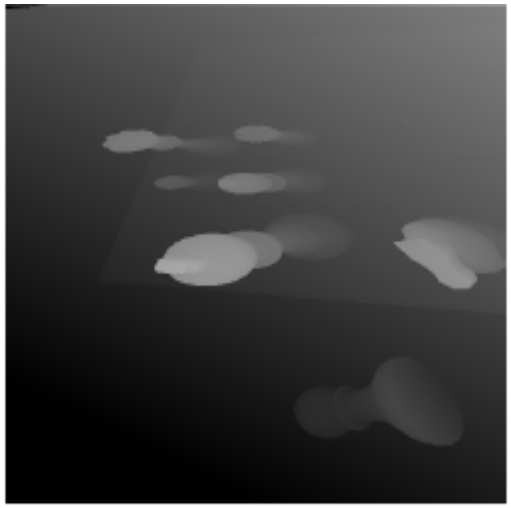
Ci sono due tipi di aliasing:

- lungo i lati delle ombre (che come detto sono ben nette a causa di luci puntiforme) si notano le tipiche scalettature;

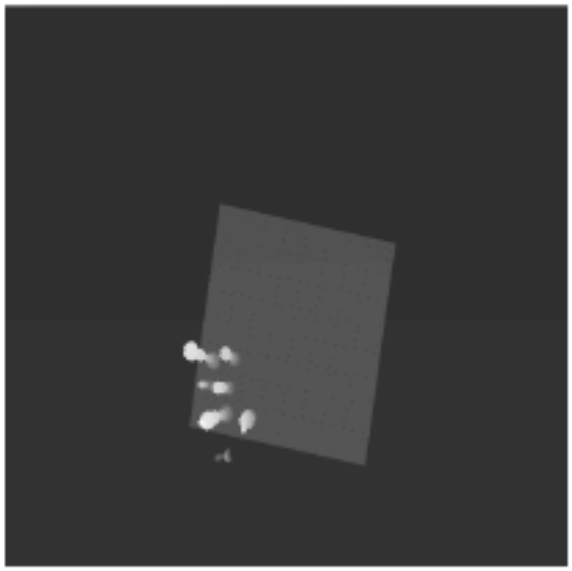
- quando si proietta un pixel nel buffer delle ombre, cioè su un discreto, si deve effettuare una decisione netta (analogo a quanto visto per le texture).



Shadow Buffer: Esempio



Shadow map



Luxo Jr. in Real-time using Shadow Mapping

Luxo Jr. Demo Details

Steve Jobs at 2001 MacWorld Japan shows this on a Mac with OpenGL using hardware shadow mapping

- **Luxo Jr. has two animated lights and one overhead light**
  - **Three shadow maps *dynamically* generated per frame**
- **Complex geometry (cords and lamp arms) all correctly shadowed**
- **User controls the view, shadowing just works**
- **Real-time Luxo Jr. is technical triumph for OpenGL**
- **Only available in OpenGL.**

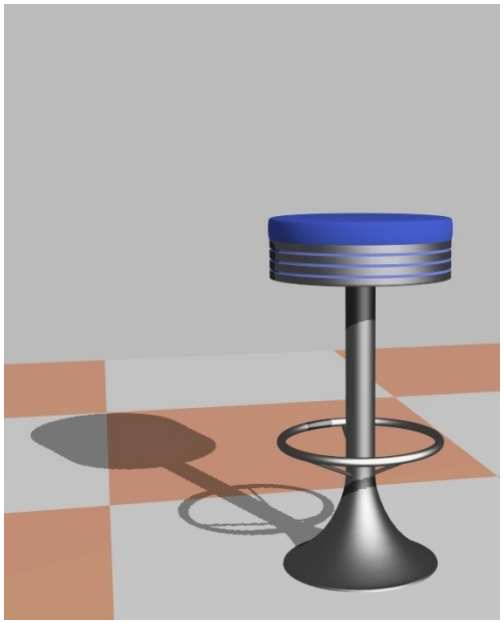
see youtube: Apple MacWorld Japan 2001 Luxo Jr. Demo
https://www.youtube.com/watch?v=nOmgUR4L8I0

Shadow Mapping

HTML5_webgl_2/sphere_shadows_spot_light.html e .js

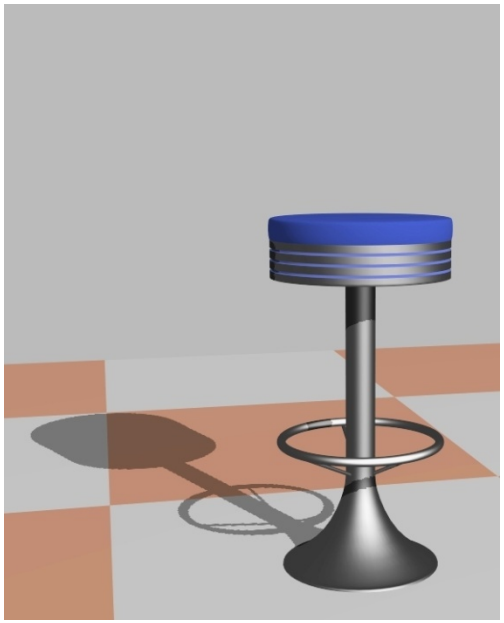
Soft Shadow

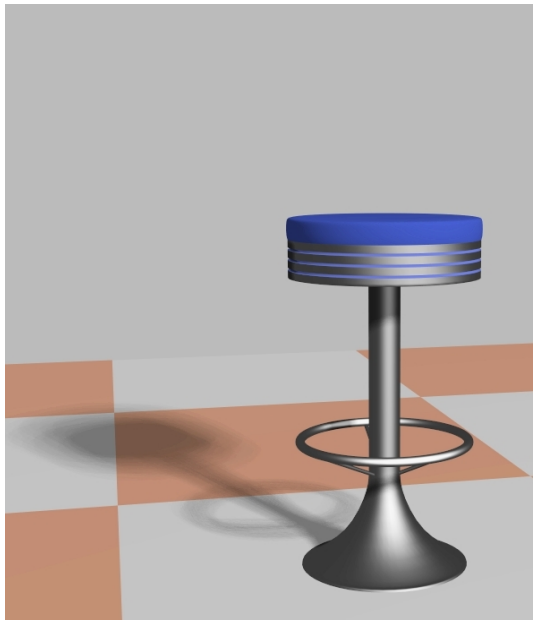
- La luce puntiforme causa, come già detto, hard shadow
- Le sorgenti puntiformi, in realtà, non sono infinitamente piccole
- Le sorgenti di luce hanno una certa area o volume e queste producono soft shadow



Soft Shadow

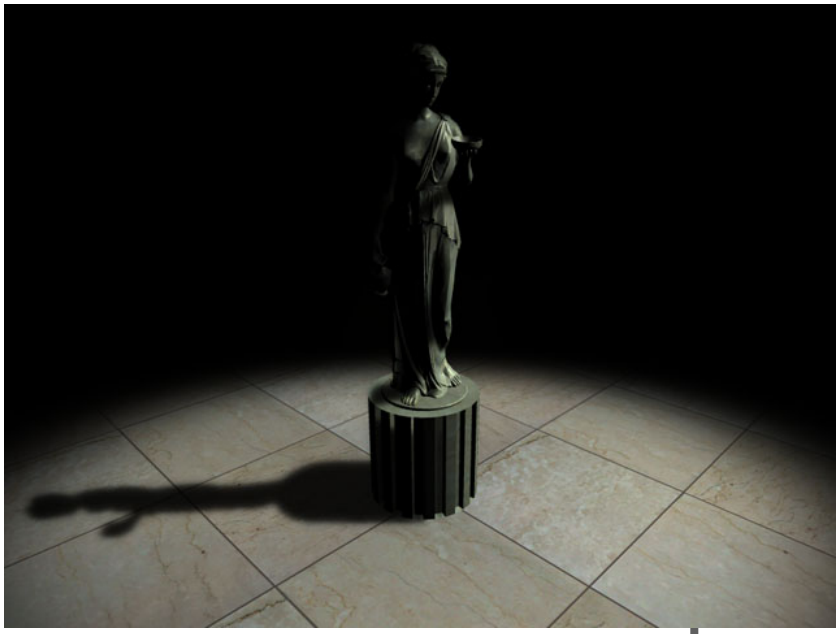
- La luce puntiforme causa, come già detto, hard shadow
- Le sorgenti puntiformi, in realtà, non sono infinitamente piccole
- Le sorgenti di luce hanno una certa area o volume e queste producono soft shadow





Soft Shadow

- Simulare un'area di luce con più sorgenti puntiforme (costoso in tempo)
- Nello spazio immagine effettuare un blur delle ombre
- Utilizzare una tecnica di Antialisaing per le ombre



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Giulio Casciola

Dip. di Matematica

giulio.casciola at unibo.it