# Grafica al calcolatore
# -
# Computer Graphics

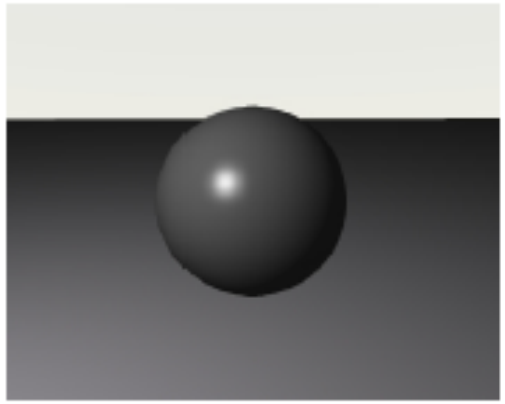
10 – Ombre

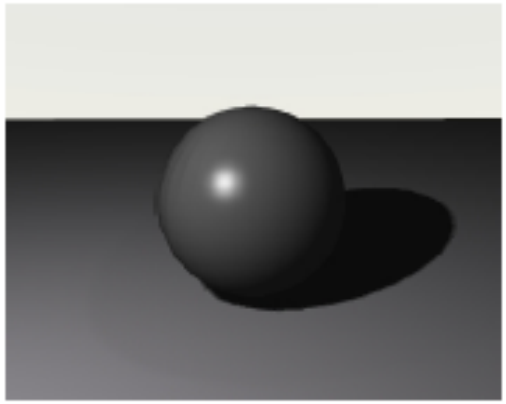
# Calendario

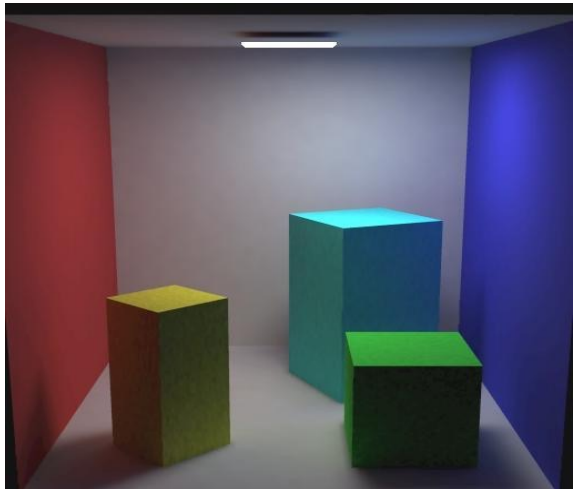
- Domani
- Giovedì 11/5: VISUAL COMPUTING DAY
- Venerdì 12/5 Laboratorio con tutor
- Giovedì 18 Lezione (anche correzione compiti)
- Venerdì 19 Lab
- Giovedì 25 Lezione (visualizzazione)
- Venerdì 26: Lab occupato.  Si può recuperare?
- Jun 1 Ultima lezione (visualizzazione/riepilogo)
- Jun 9 Ultimo lab e seconda consegna

# Effetti “globali”

- L'uso della pipeline di rasterizzazione con tecniche di illuminazione locale ha molti limiti.
- Simuliamo le interriflessioni con la componente ambientale
- Abbiamo usato le texture per generare effetti di riflessione speculare o illuminazione view independent
- Nulla abbiamo fatto per un importante effetto non locale: l'ombra
  - Un particolare che si nota immediatamente nelle immagini ottenute con modelli di illuminazione locale, è l’assenza di ombre proiettate, che sono fondamentali per dare profondità e realismo all’immagine.

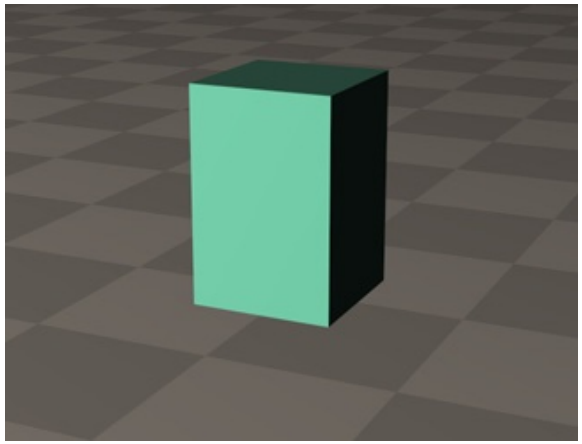


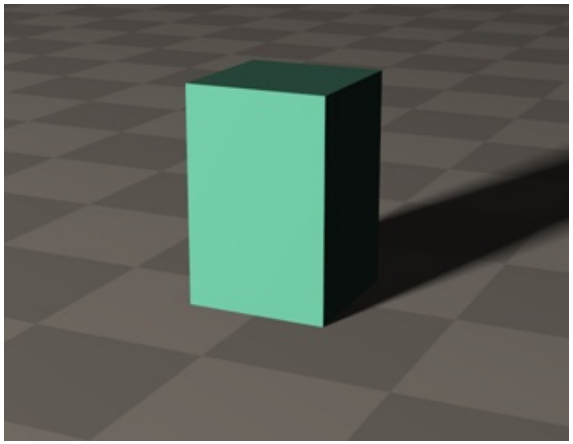




# Ombre e percezione

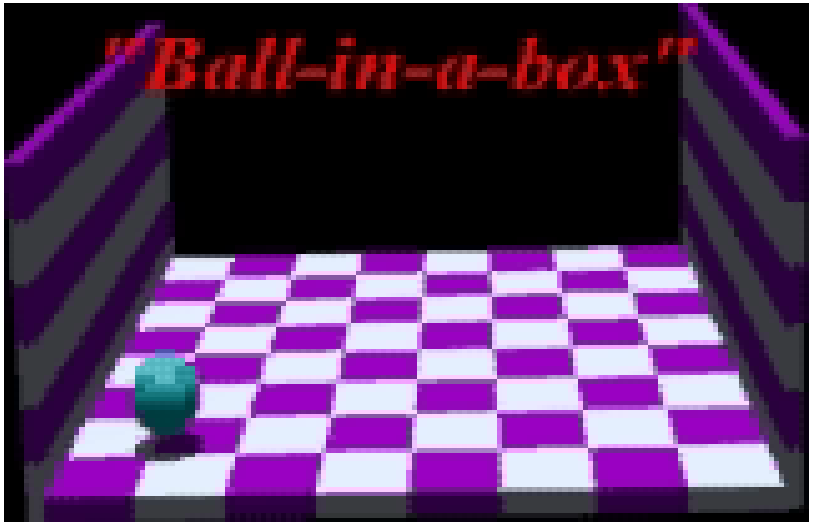
- Le ombre sono fondamentali per la percezione delle scene
- Chiariscono la posizione degli oggetti
- Attraverso di esse si riesce a interpretare il moto degli oggetti



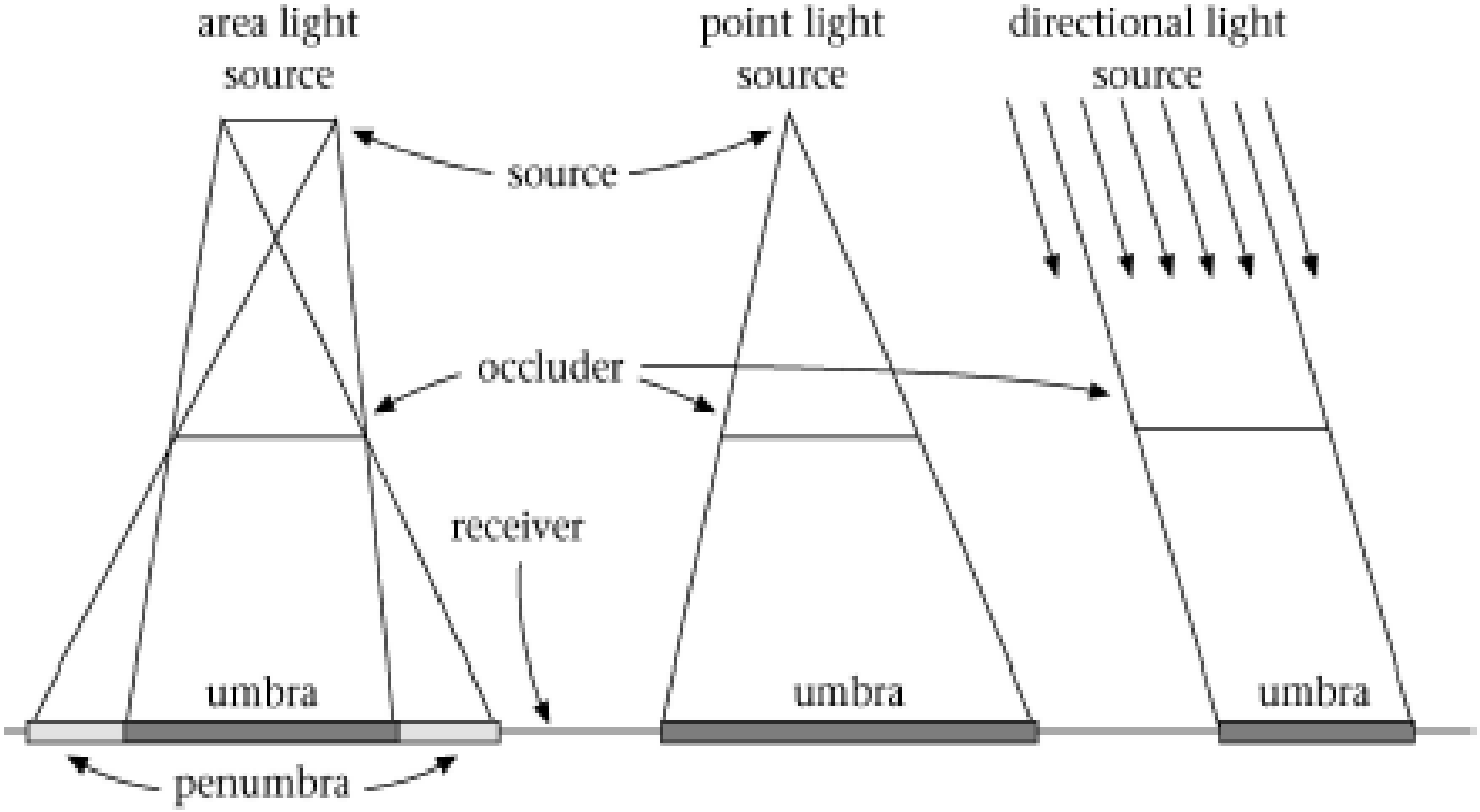


# Ombre e percezione

- Le ombre sono fondamentali per la percezione delle scene
- Chiariscono la posizione degli oggetti
- Attraverso di esse si riesce a interpretare il moto degli oggetti

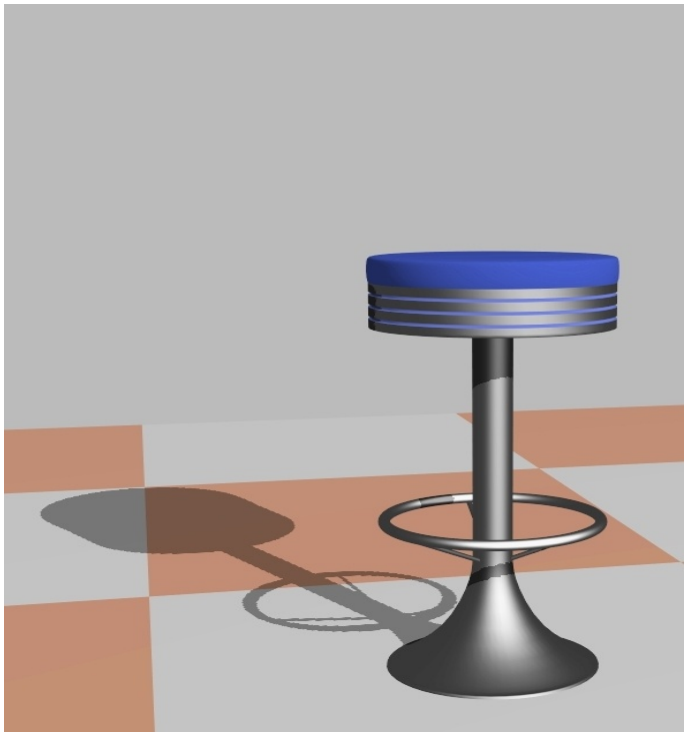


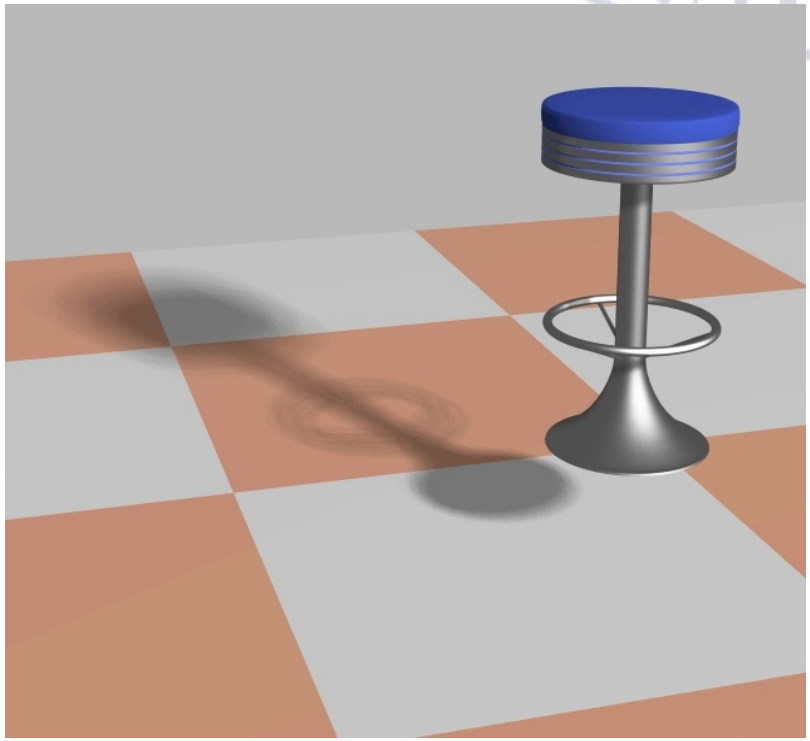
# Ombre proiettate



# Ombra e penombra

- Nella realtà essendoci illuminazione diffuse, le ombre sono sempre sfumate (penombra)
  - Ma non possiamo ottenerle da luci puntiformi

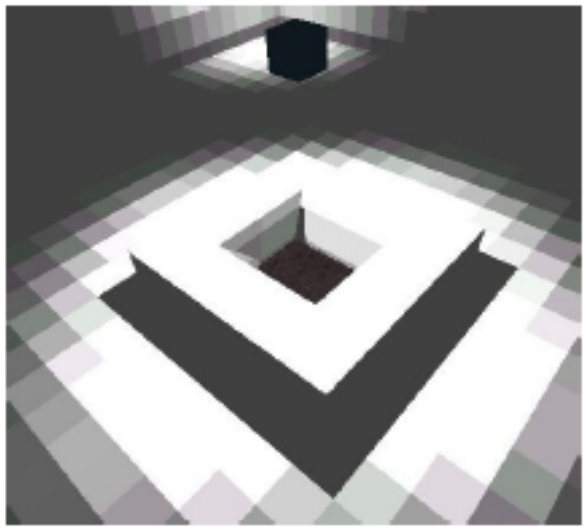




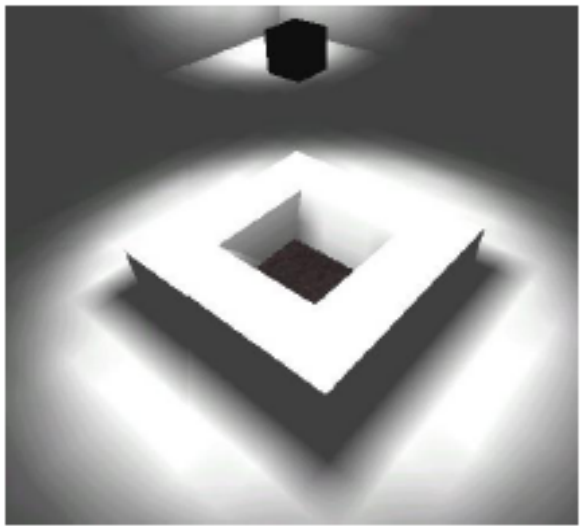
# Proprietà delle ombre proiettate

- L'ombra che il poligono A getta sul poligono B a causa di una sorgente luminosa puntiforme si può calcolare proiettando il poligono A sul piano che contiene il poligono B con centro di proiezione fissato in coincidenza della sorgente.
- Non si vede alcuna ombra se la sorgente luminosa coincide con il punto di vista. Ovvero, le ombre sono zone nascoste alla luce. Questo implica che si possono usare tecniche (modificate) di rimozione di superfici nascoste per calcolare le ombre.
- Per scene statiche le ombre sono fisse. Non dipendono dalla posizione dell'osservatore
- Se la sorgente (o le sorgenti) è puntiforme l'ombra ha un bordo netto, ovvero non c'è penombra (come nello spazio).

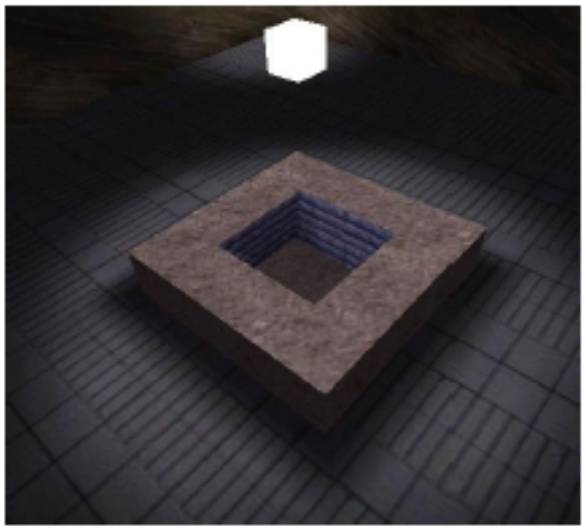
- Le ombre (geometriche) possono essere precalcolate per oggetti statici, ed aggiunte alla lightmap (molto usato nei videogiochi).



(7) Lightmap + geomeric shadows



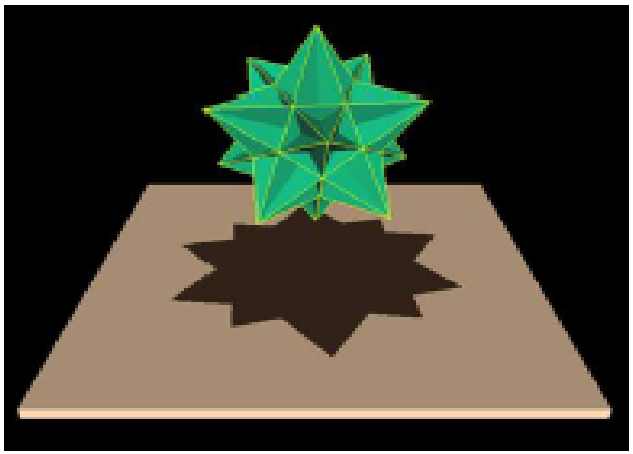
(8) Filtrata



(9) Applicata all'oggetto con texture

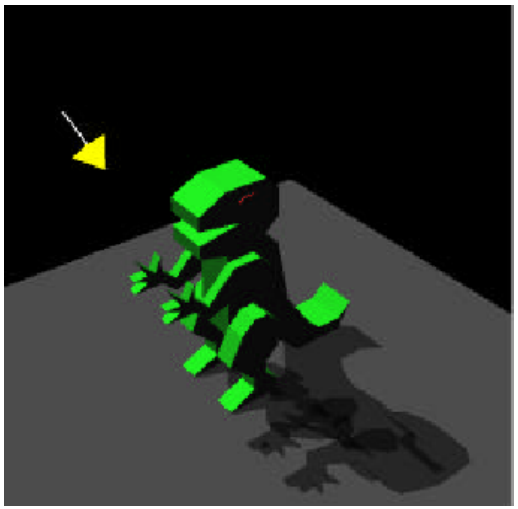
# Ombra sul piano (projective shadows)

- Si tratta di una tecnica per calcolare l'ombra portata da un oggetto su un piano $\pi$ (di solito il terreno) a causa di una luce posizionata in $\mathbf{P}_l$.
- L'idea è di disegnare l'ombra come un oggetto piatto –(tipicamente nero) sul piano.
- Per disegnare l'ombra si effettua il rendering della scena, con colore nero, con una telecamera centrata in $\mathbf{P}_l$ e che ha il piano $\pi$ come piano immagine.
- Se la luce è direzionale (distant light) si una una matrice di proiezione ortogonale
- La tecnica è semplice ed efficiente: si tratta semplicemente di fare il rendering della scena due volte.



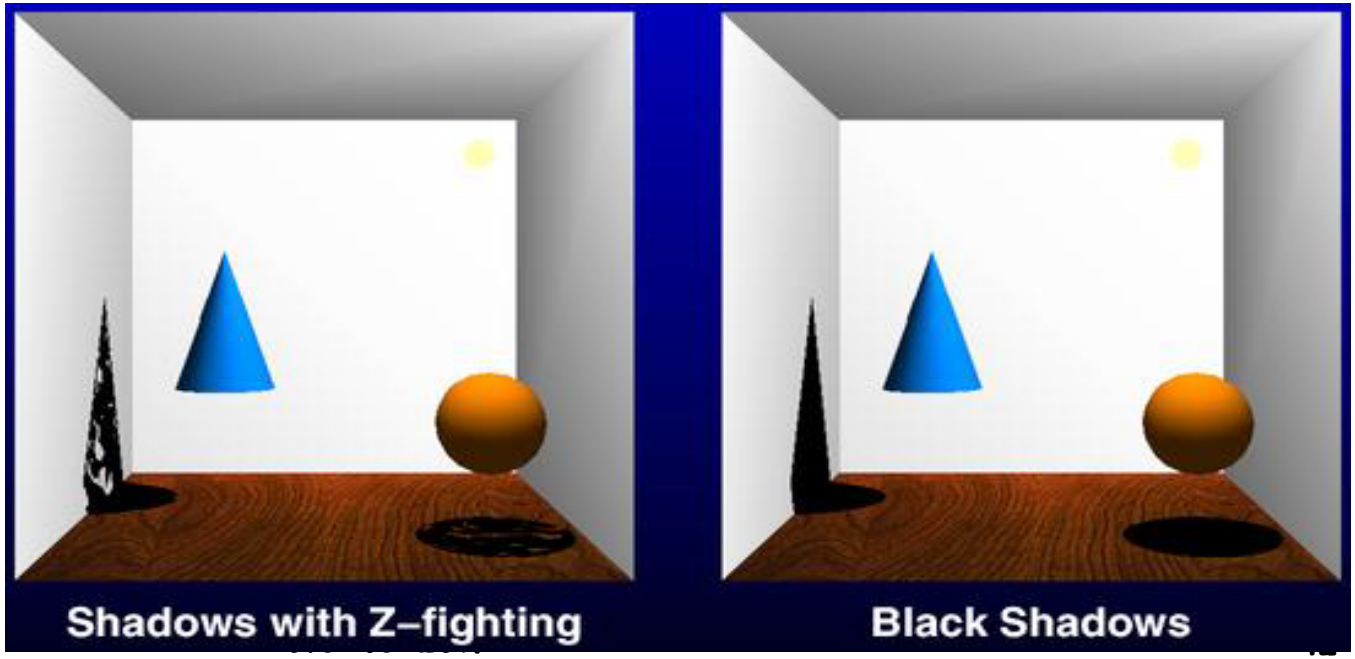
# Ombra sul piano

- In pratica, in termini di matrici, possiamo calcolare la matrice che mappa l'oggetto sul piano
  - Disegniamo il piano
  - Calcoliamo l'ombra dell'oggetto
  - Facciamo il rendering di questa con trasparenza sopra
- Il metodo gestisce oggetti arbitrariamente complicati,
  - ma la scena è vincolata ad essere molto semplice: un solo oggetto oppure più oggetti sufficientemente distanti da non farsi ombra.



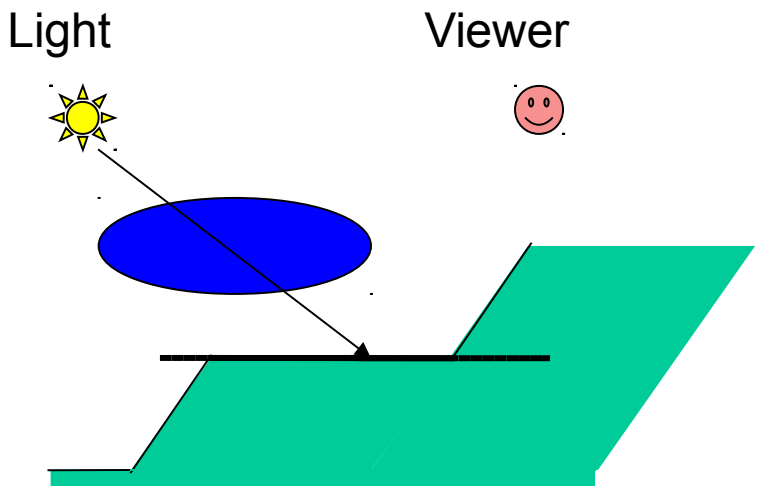
# Problema

- I triangoli d'ombra giacciono sullo stesso piano del piano $\pi$ visto sopra; ci possono essere problemi di risoluzione dello z-buffer per cui nel rendering alcuni di questi triangoli (o una loro parte) possono finire sotto il piano (z-fighting):
  - Trucco:alzare un poco il piano (può essere fatto cambiando un termine della matrice

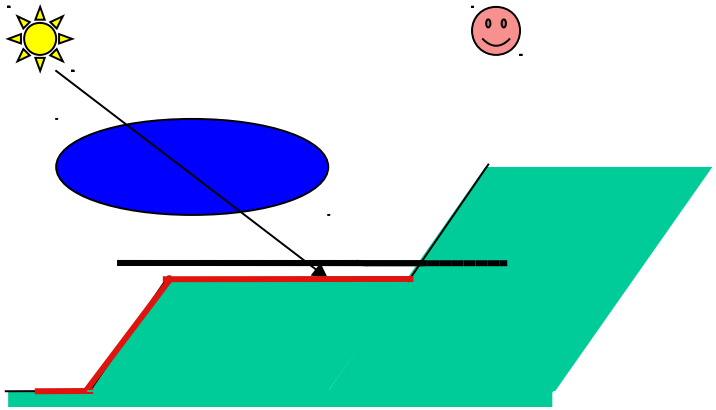


# Soluzione

- Alzare di poco il piano su cui si proietta l'ombra



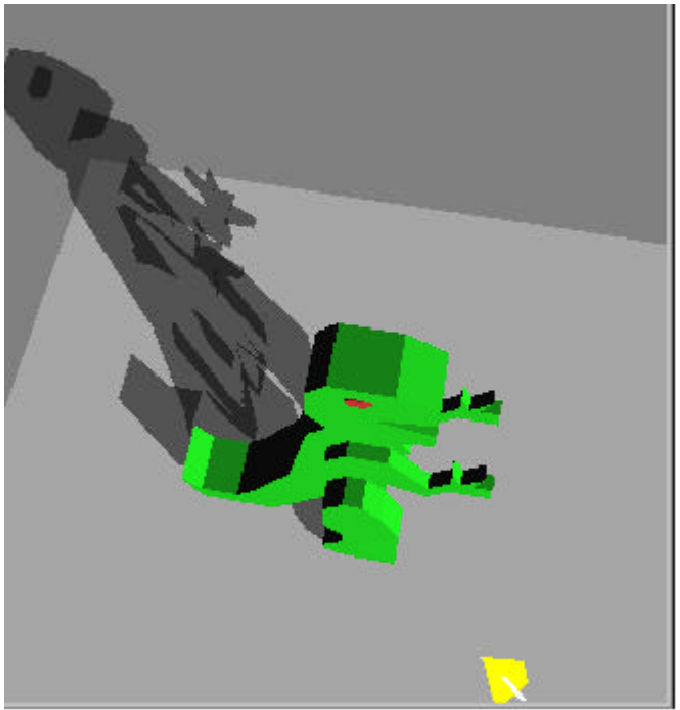

Shadow height equal with hit polygon
Z-buffer quantization errors

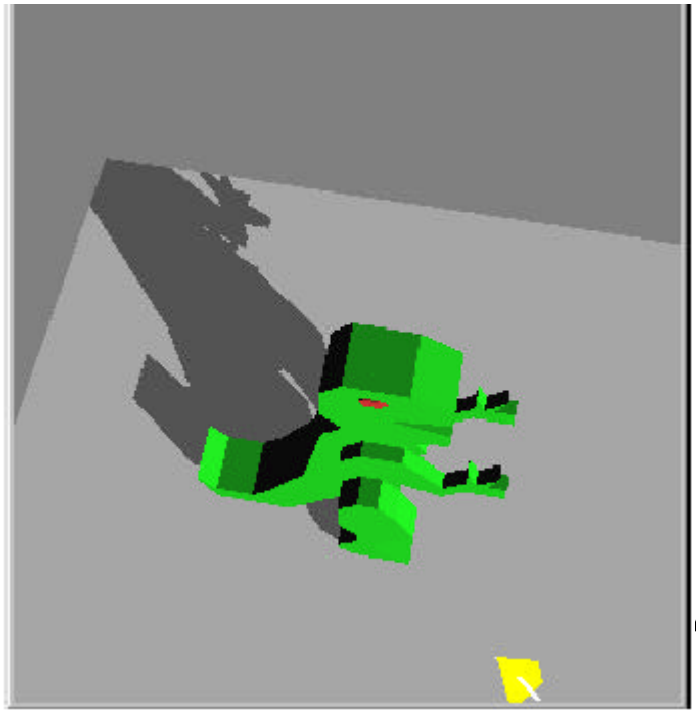


Shadow height above hit polygon

# Altri problemi

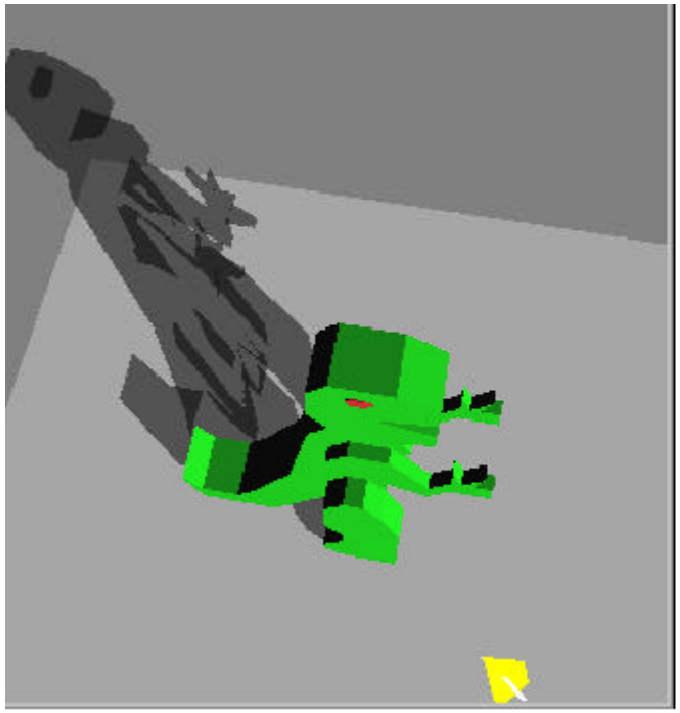
- Le ombre hanno contorni netti
- Se il piano su cui proietto ha una texture sopra il risultato è brutto
- Utilizzando il blending per sovrapporre l'ombra dove di ci sono poligoni multipli il risultato è più scuro
- Le ombre vanno ridisegnate anche se l'oggetto non si muove

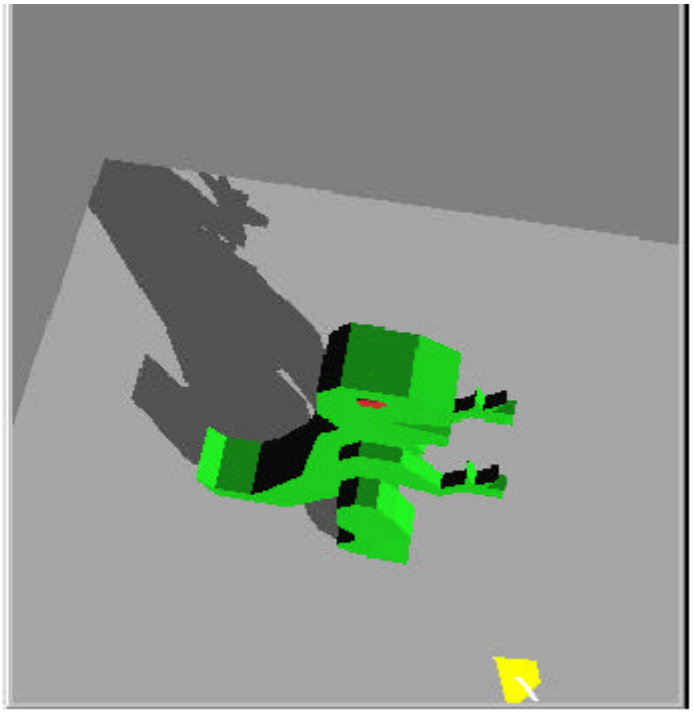




# Uso dello stencil buffer

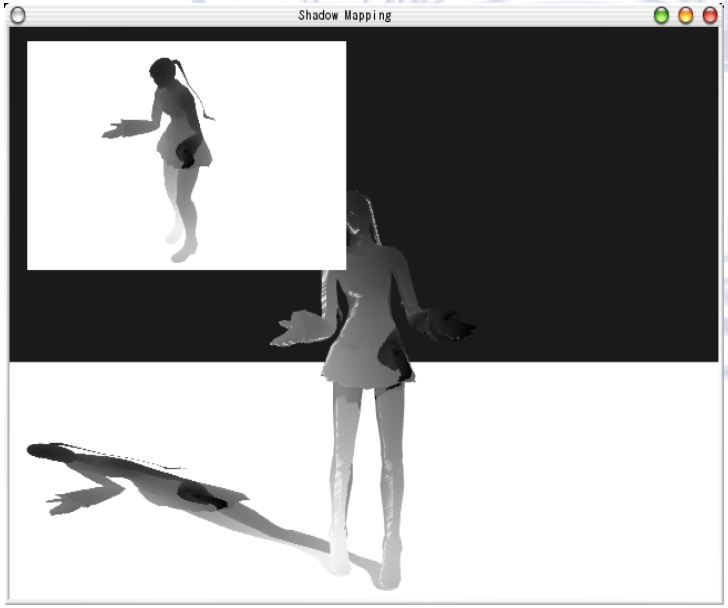
- Abbiamo nominato questo particolare buffer di openGL
  - Maschera per disabilitare rendering
  - Possiamo qui usarlo con due scopi
    - Proiettare solo sul piano
    - Inibire i frammenti dove è già stato proiettato: rimuove il problema delle parti più scure

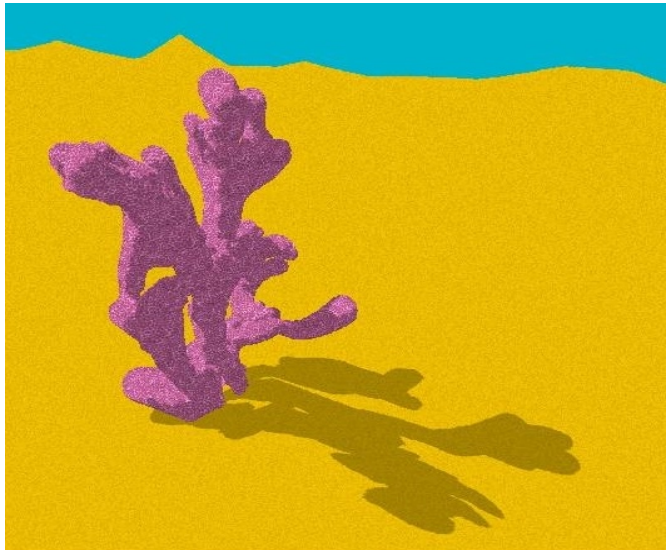




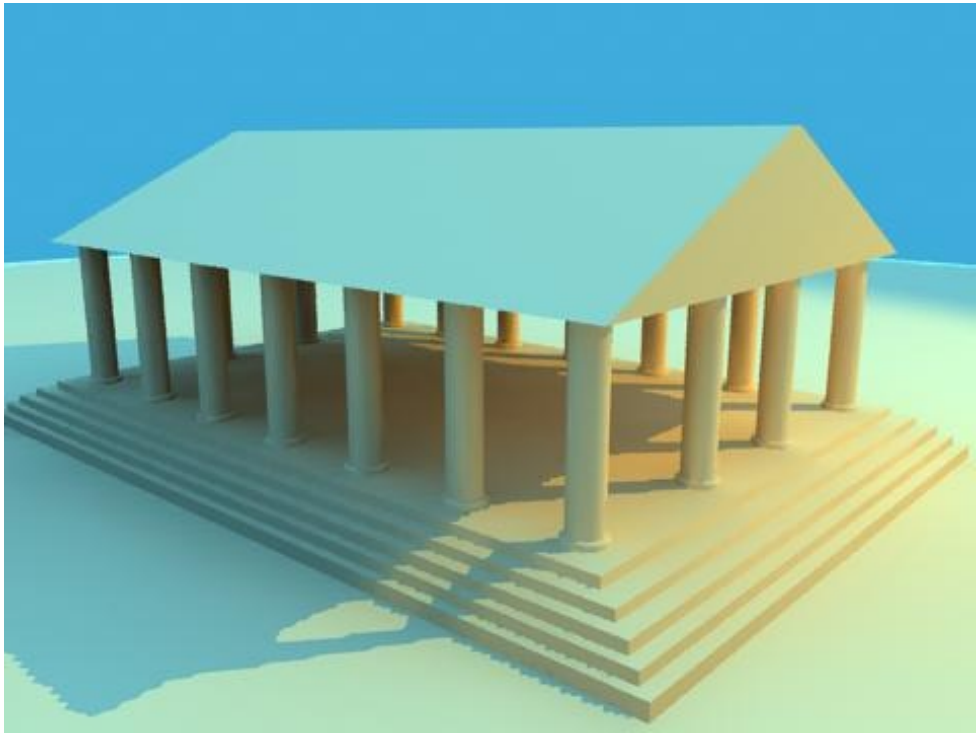
# Shadow Texture

- Altra idea usare l'immagine ombra e usarla come texture
- Generare un'immagine dell'occlusore dal punto di vista della luce e colorarla di grigio
- Proiettarla sullo sfondo e usarla per texture mapping
- Può generare ombre su oggetti non piani



# Problemi



- Occorre specificare geometria occlusore e "ricevitore"
- Gli oggetti non possono farsi ombra a vicenda
- Texture magnification
- Problemi con la risoluzione
  - Si potrebbe rendere adattativa e migliorare il risultato

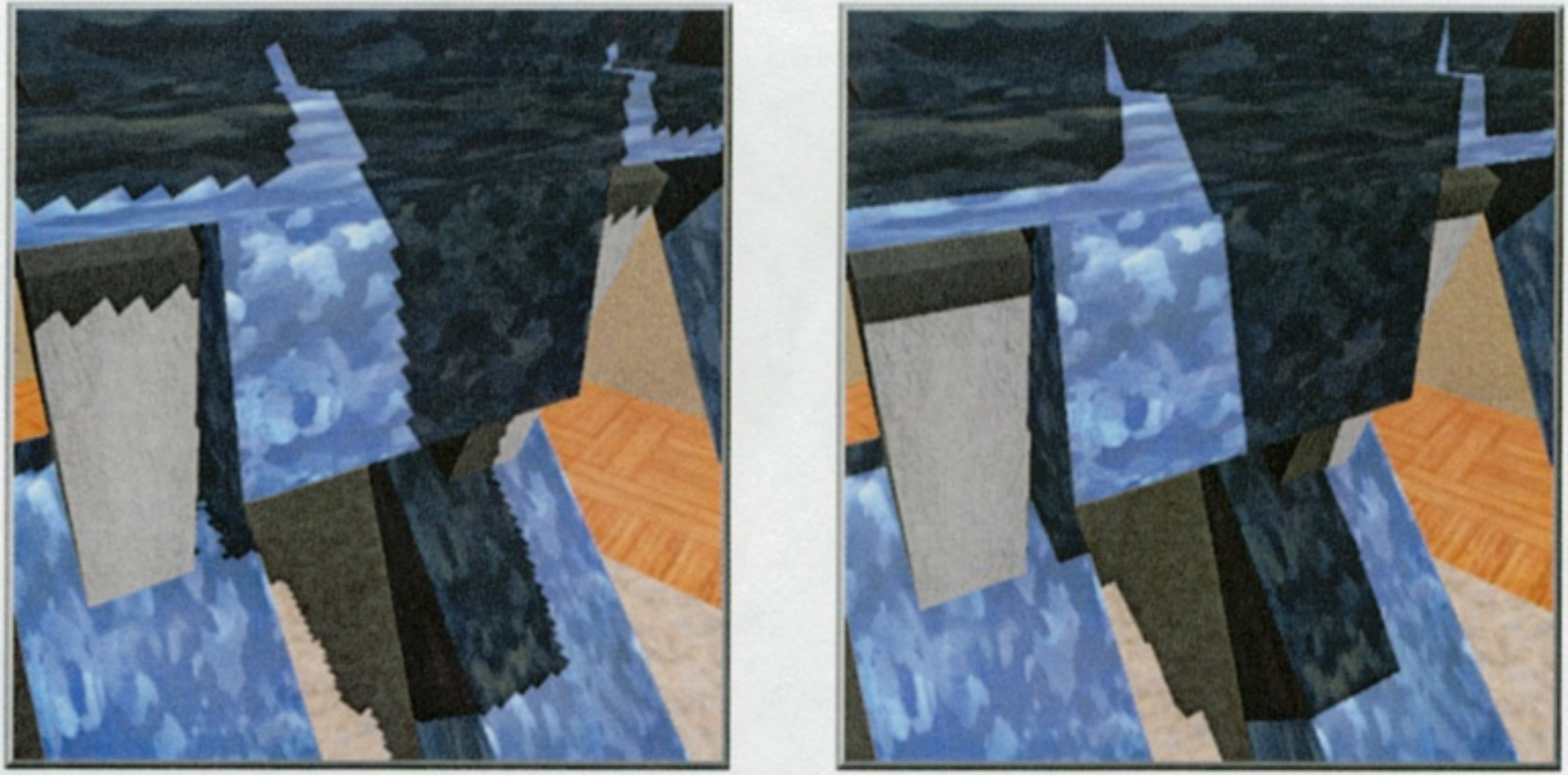
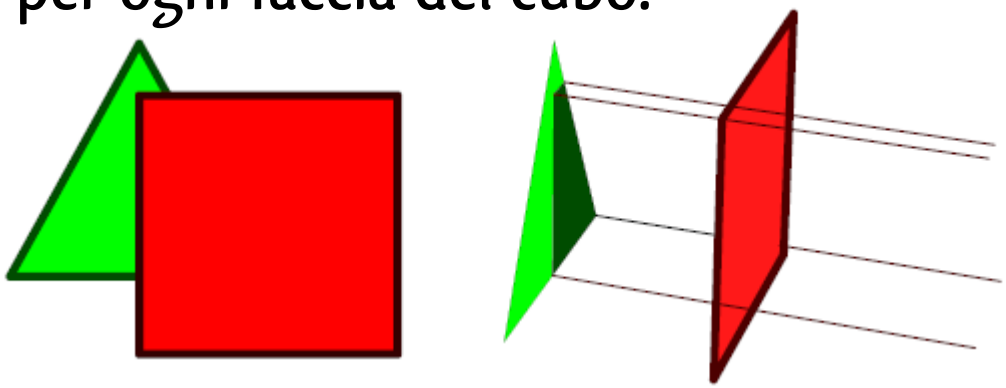


Figure 2: A conventional 2,048×2,048 pixel shadow map (left) compared to a 16 MB ASM (right). EFFECTIVE SHADOW MAP SIZE: 65,536×65,536 PIXELS.

# Shadow mapping/buffer

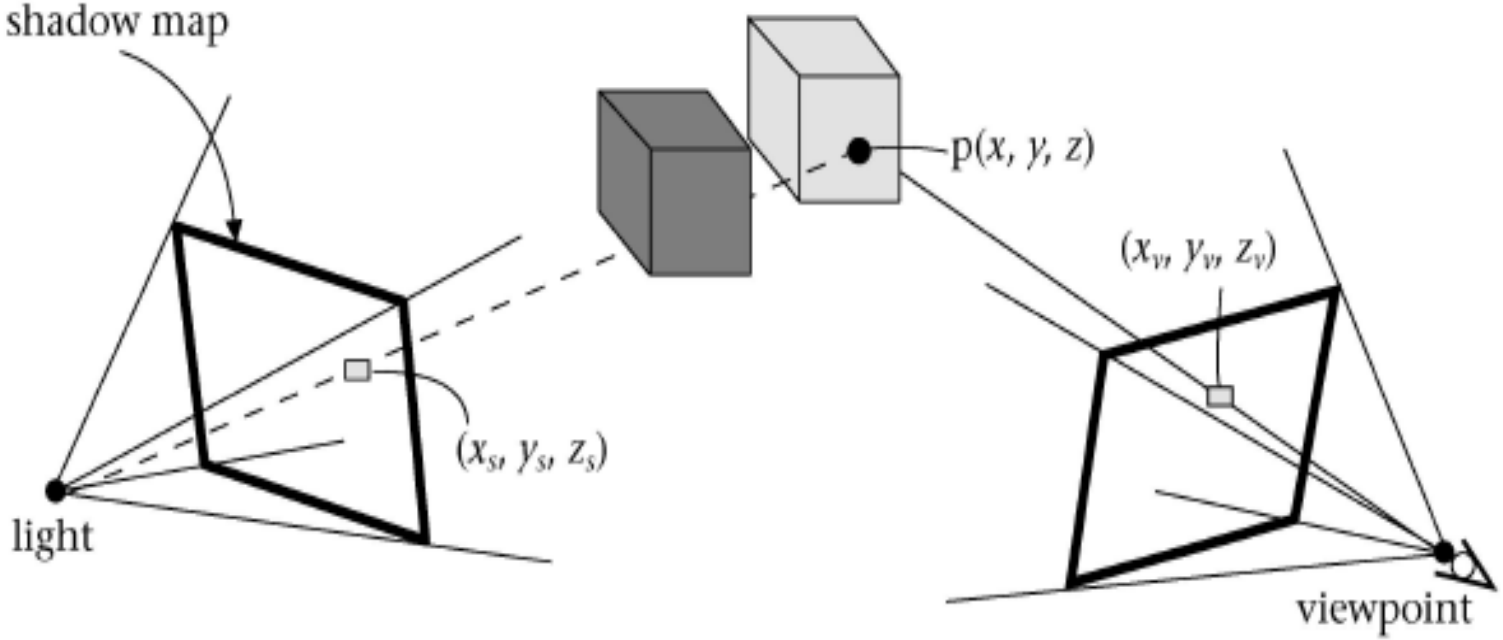
- (Two-pass Z-buffer shadow algorithm)
- Questa tecnica si rifà alla seconda osservazione sulle proprietà delle ombre, ovvero si basa su un algoritmo di rimozione delle superfici nascoste.
- Si calcola uno z-buffer dal punto di vista della luce (detto anche shadow buffer o shadow map):
  - Se la luce è spot-light allora basta calcolare uno z-buffer singolo
  - Se la luce è una point-light, la si immagina racchiusa in un cubo e si calcolano sei z-buffer, uno per ogni faccia del cubo.



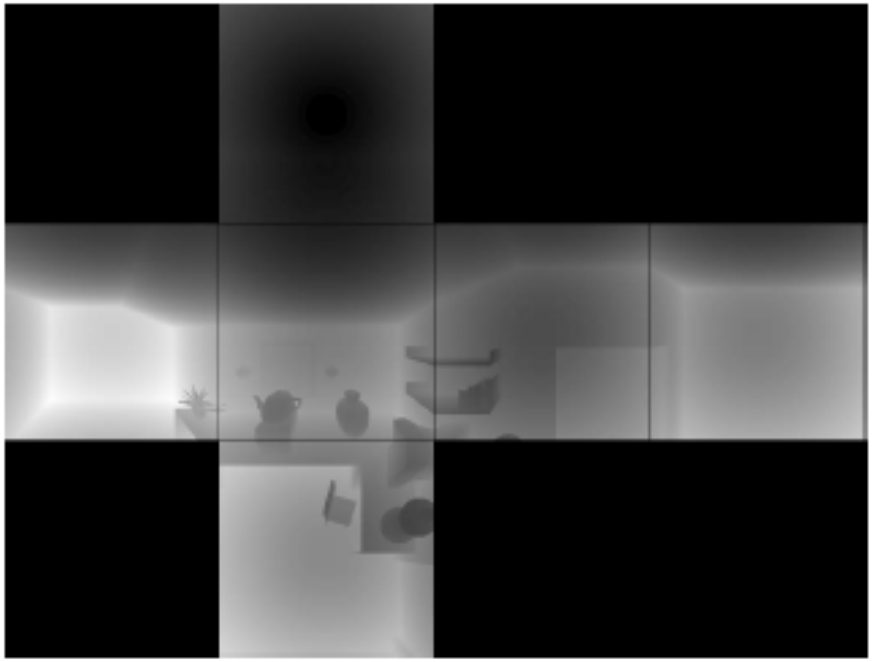
Scene from light source

Scene from eye

- In fase di rendering, se un punto di un poligono deve essere disegnato, lo si trasforma nel sistema di riferimento della luce e si trova il valore $z_l$ nello shadow buffer
- Se lo $z$ del punto (nel riferimento della luce, non della camera) è più grande di $z_l$, significa che vi è un oggetto che blocca la luce per quel punto, quindi è in ombra e lo si può colorare di conseguenza. Altrimenti è colpito dalla luce e si opera lo shading normalmente
- Se si hanno più luci bisognerà avere uno shadow buffer per ognuna.
- La tecnica eredita le inefficienze dello z-buffer: richiede molta memoria e compie calcoli che poi possono venire sovrascritti.



# Shadow buffer



(10) Shadow buffers

(11) Image

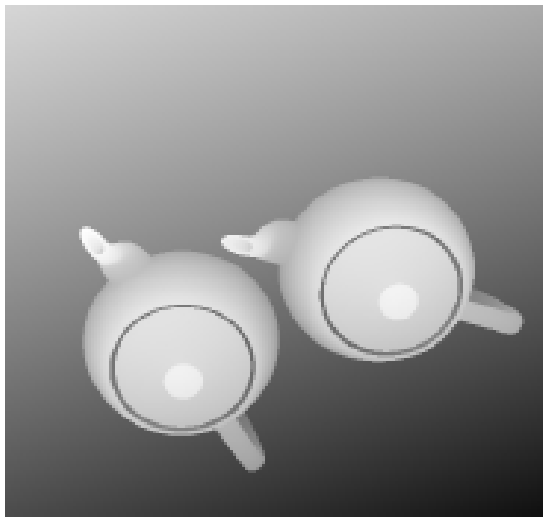
© Alan Watt

# Esempio: environment map e shadow buffer

(12) Scena prima

(13) Scena dopo

(14) Envmap

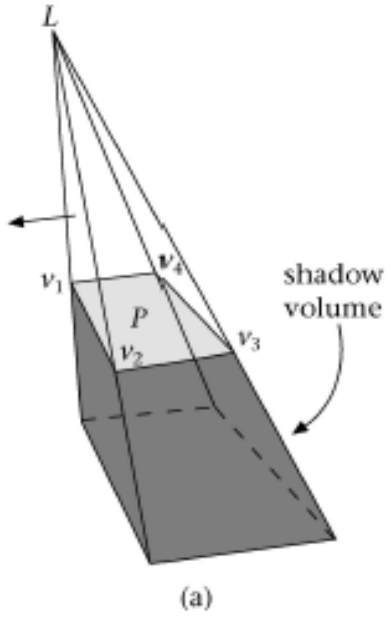


(15) Shadow buffer

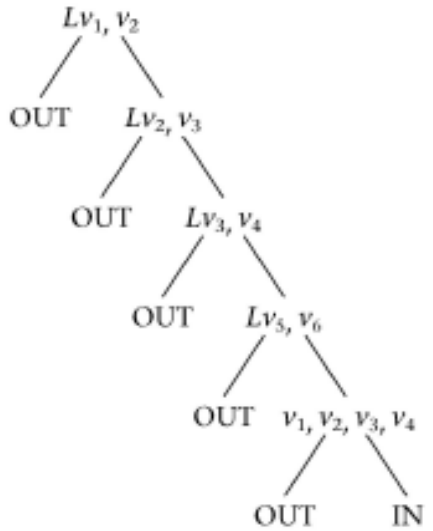
# Shadow volume

- Metodo introdotto da Crow (1977) e Bergeron (1986).
- Il volume d'ombra (shadow volume) di un poligono rispetto ad una sorgente luminosa puntiforme è la parte di spazio che il poligono occlude alla vista della luce

- E' un tronco di piramide semi-infinita (senza base) che ha il vertice nella sorgente luminosa ed è limitata da una parte dal poligono stesso.
- Le facce della piramide prendono il nome di shadow planes. Vengono rappresentate in modo che il "fronte" sia verso la luce ed il "dietro" verso l'ombra.

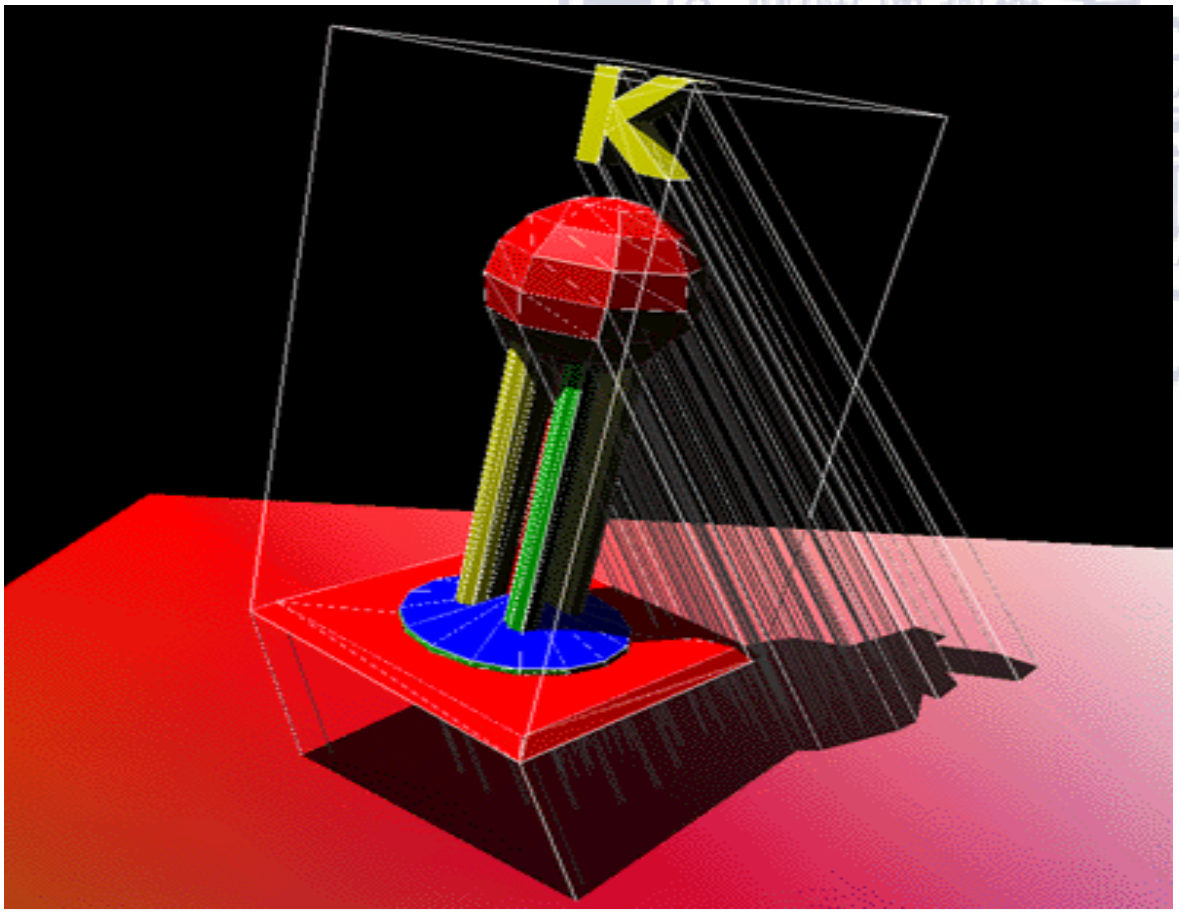


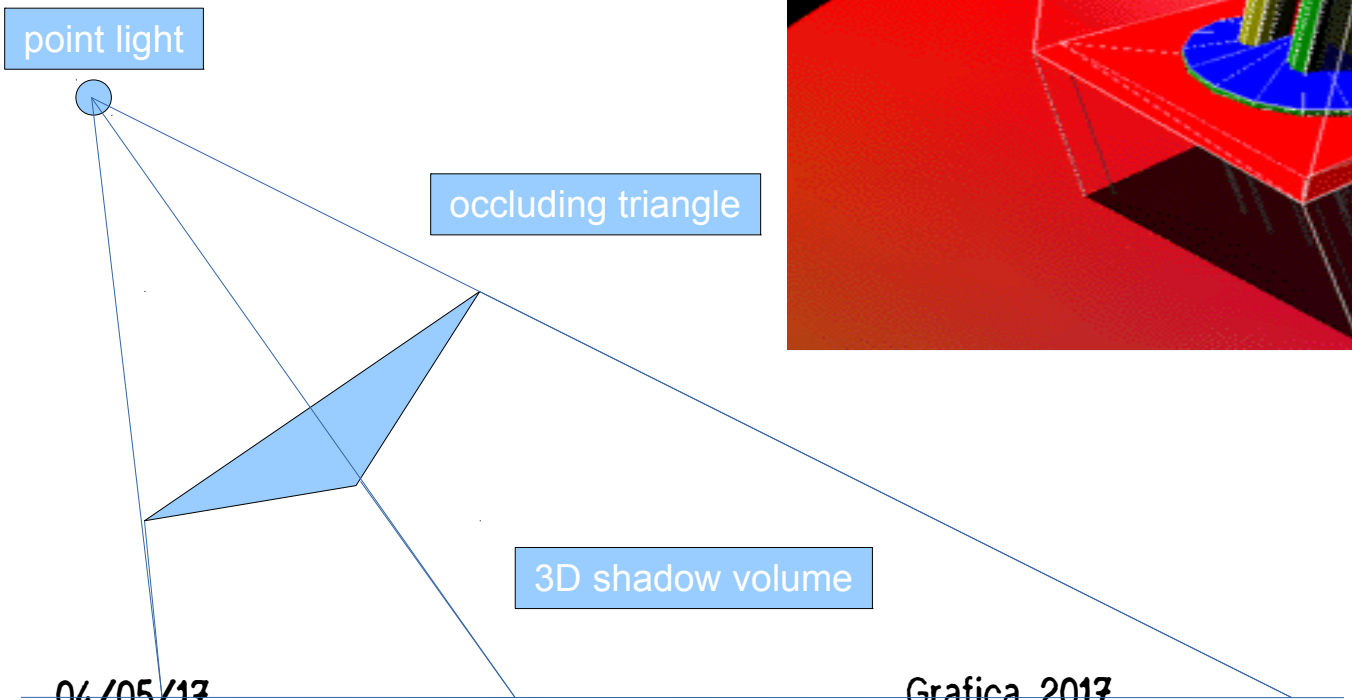
(a)



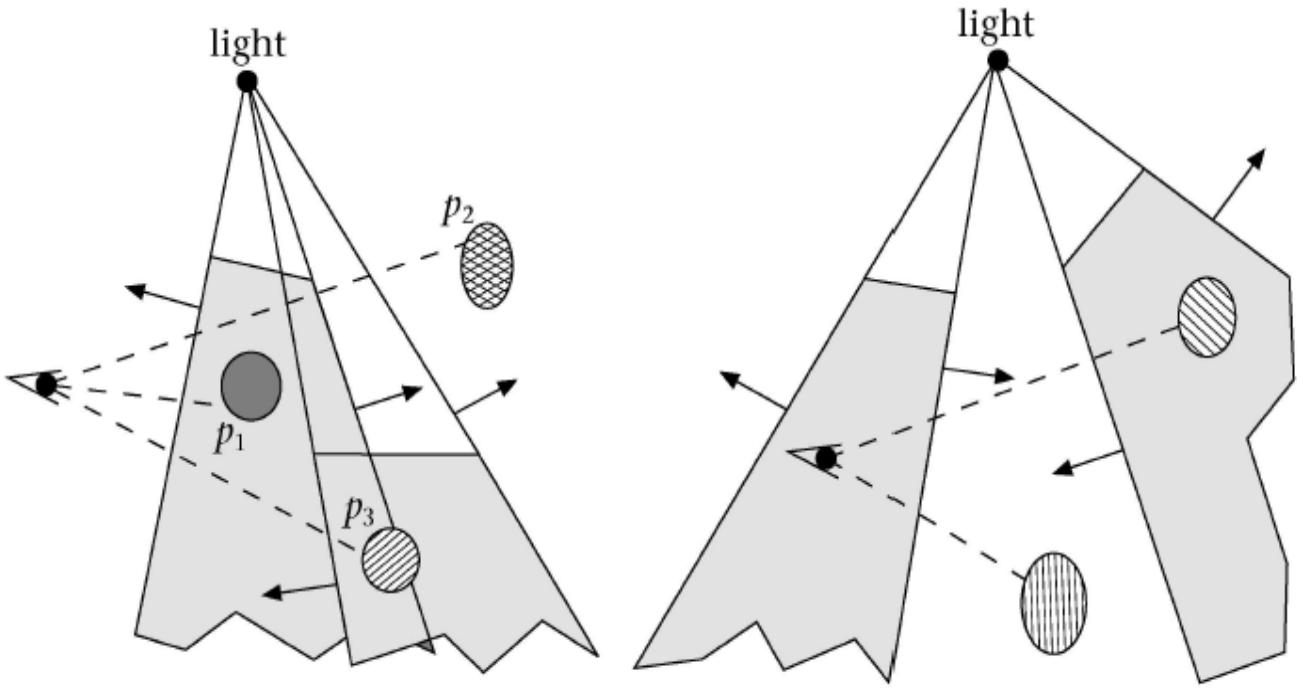
(b)

# Shadow volumes





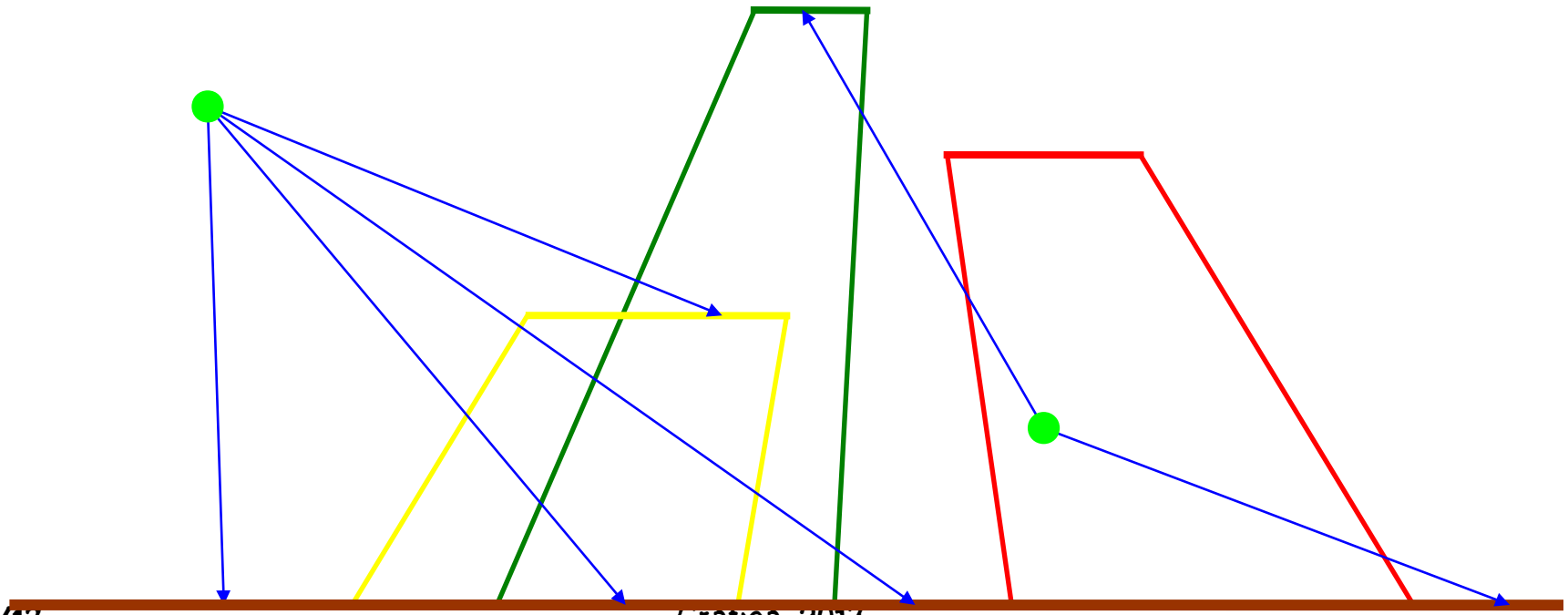
- Dato un punto di vista non in ombra, e dato un punto P della scena, tracciamo il segmento di retta che congiunge il punto di vista e P e contiamo gli attraversamenti degli shadow planes, incrementando un contatore quando l'attraversamento è in entrata (dal fronte) e decrementandolo quando è in uscita (dal retro). Se la differenza è zero il punto è nella luce, altrimenti è nell'ombra (di uno o più poligoni).
- Se il punto di vista è nell'ombra bisogna partire dal numero di shadow volumes nei quali è contenuto. Si calcola contando le intersezioni con gli shadow planes lungo una semiretta arbitraria con origine nel punto di vista.



# Esempio

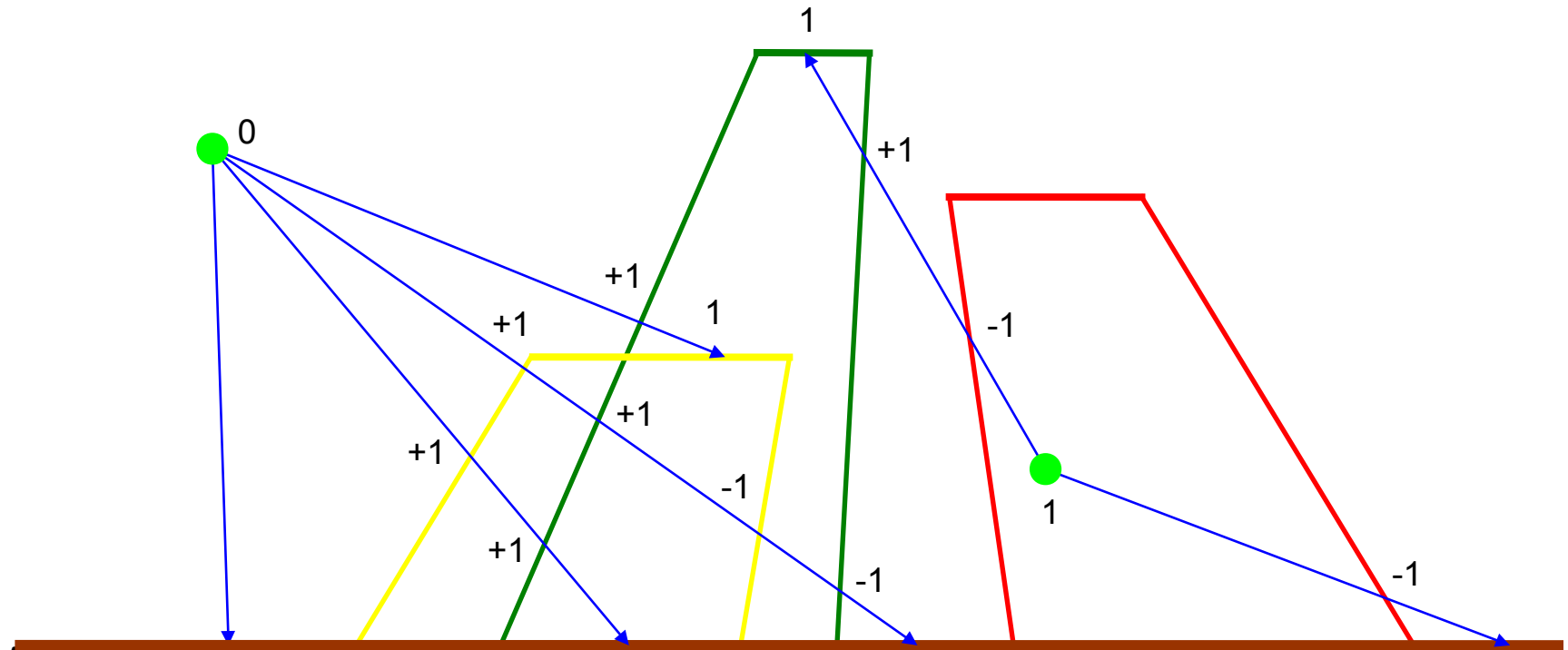


- Calcolare il contatore per i punti visibili



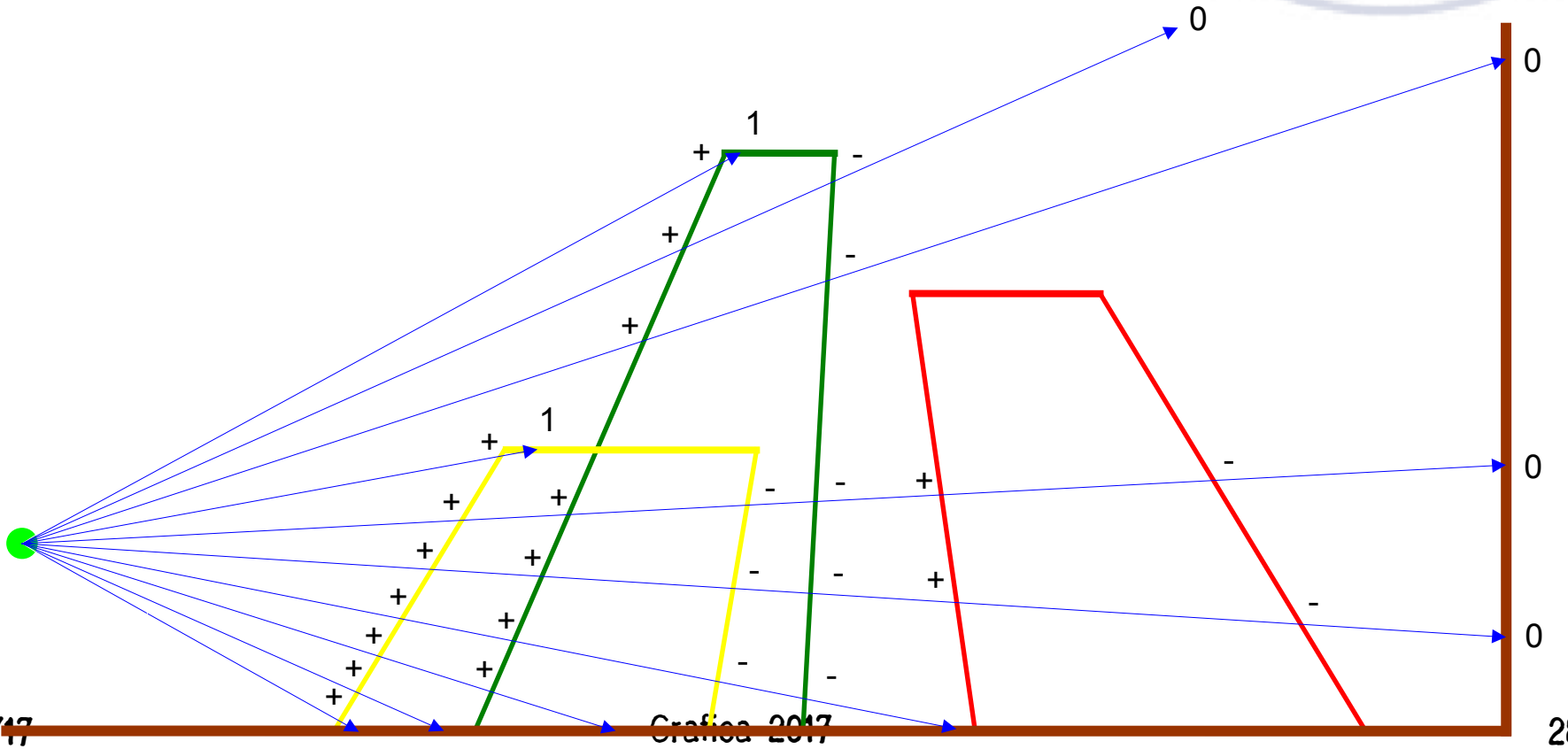
# Esempio

- Calcolare il contatore per i punti visibili



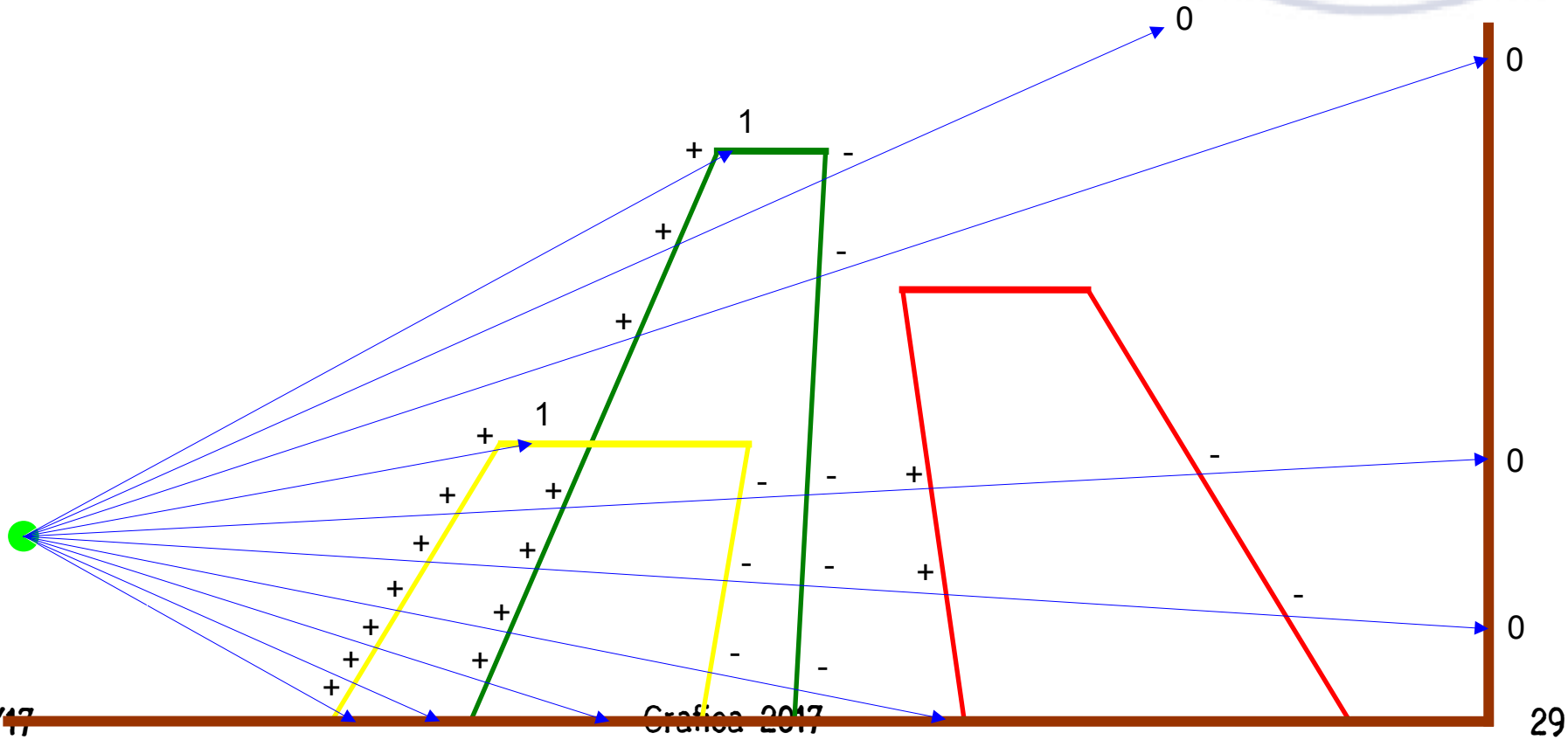
# Come si calcola in pratica?

- Ray casting? Troppo complesso



# Come si calcola in pratica?

- Osservazioni:
  - Se l'occlusore è convesso lo è anche l'ombra
  - La linea di vista entra dalle facce front facing e esce dalle back facing

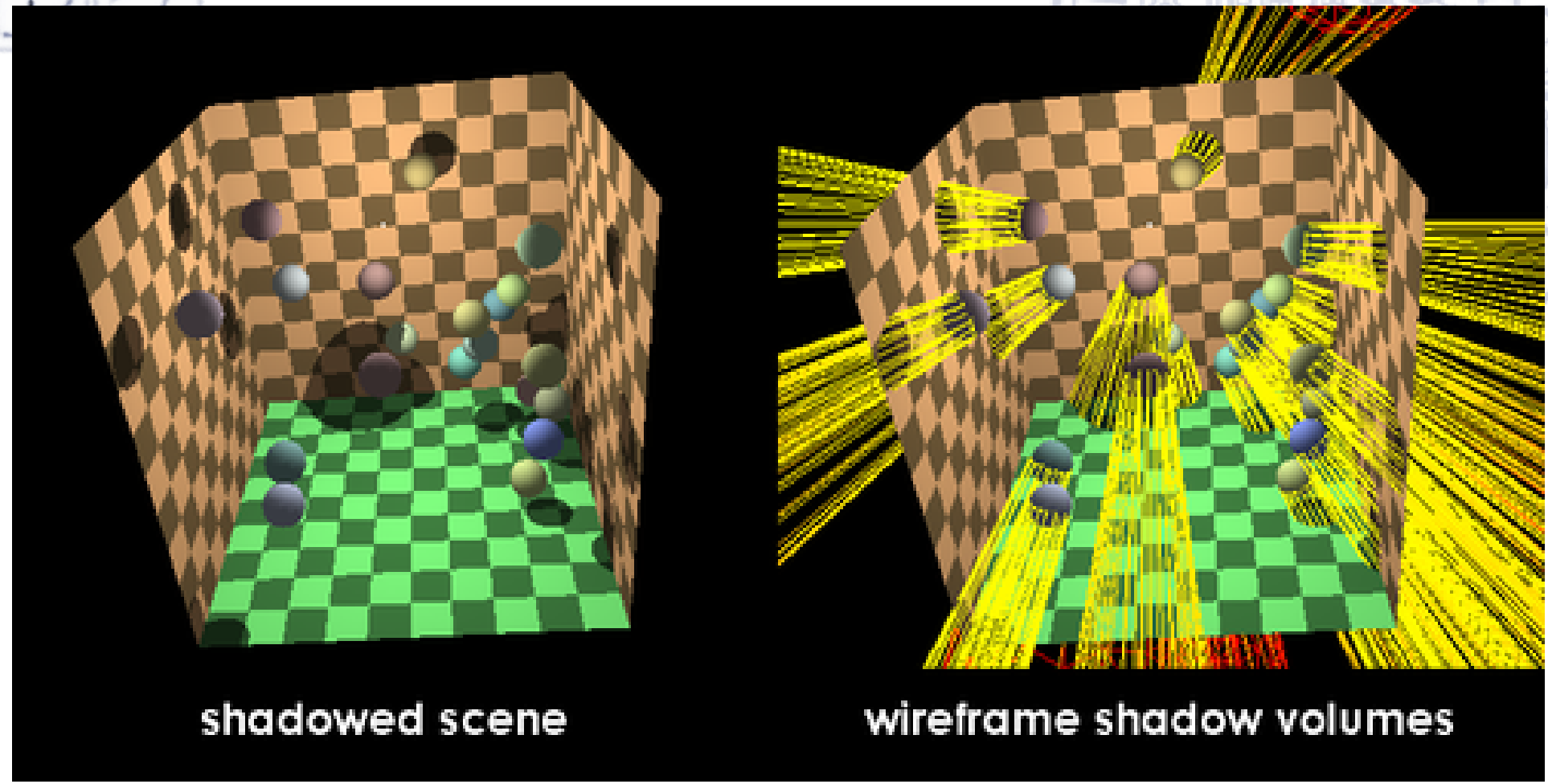


# Soluzione

- Usare la pipeline hardware di rasterizzazione per calcolare il contatore
  - Creare shadow volumes per ogni occlusore (convesso)
  - Fare rendering con componente ambientale tenere la depth
  - Per ogni sorgente
    - Inizializzare buffer al numero di volumi contenenti il punto di vista
    - Usando z buffer test disabilitando il suo aggiornamento
      - Fare rendering solo front facing dei poligoni degli occlusori incrementando buffer (stencil) per tutti I frammenti che superano il depth test
      - Fare rendering con solo back facing decrementando il buffer/contatore
    - Solo sui pixel col buffer a 0 si rifà il rendering con l'illuminazione accesa (sorgenti puntiformi)

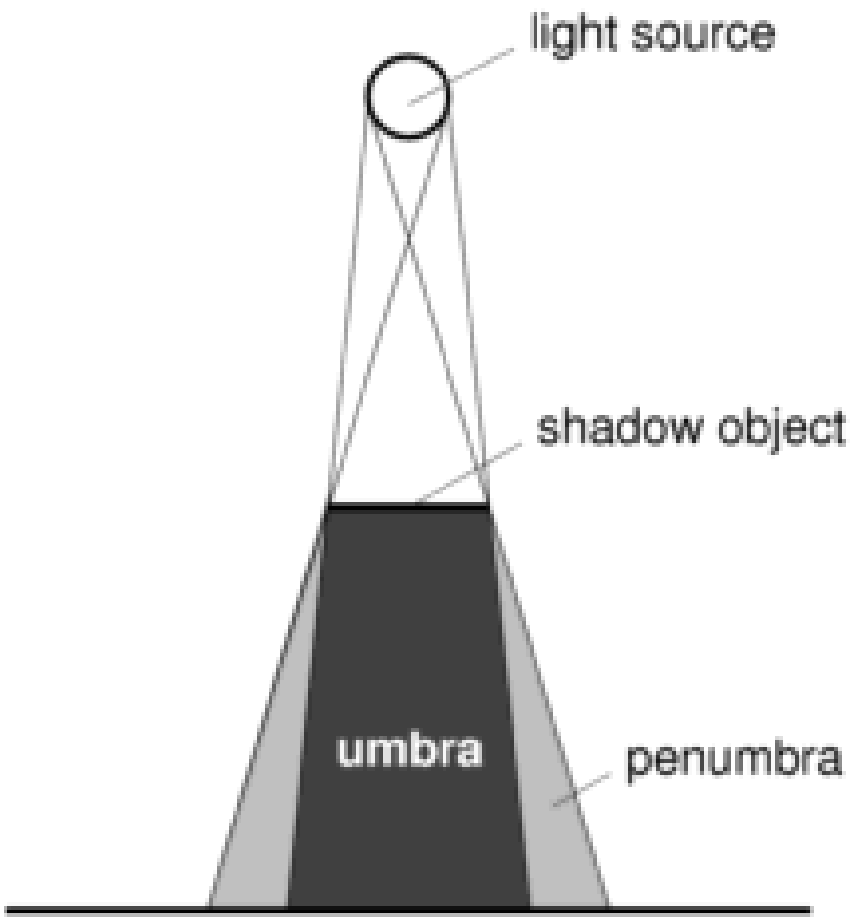
# Shadow volume

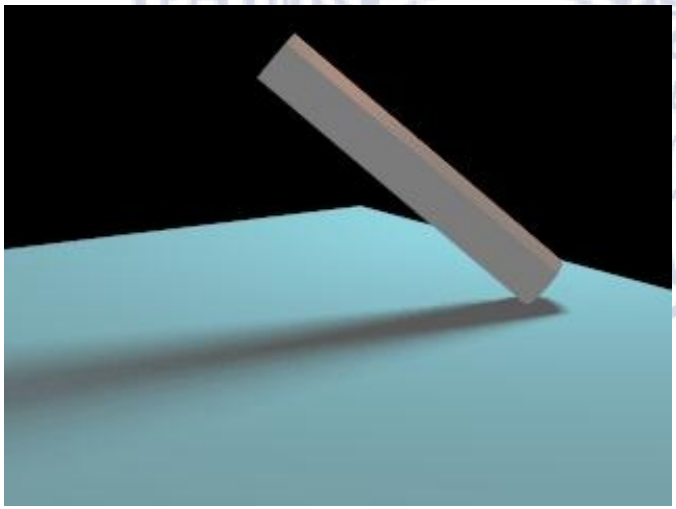
- Riassumendo: la generazione degli shadow planes avviene off-line, vengono aggiunti alla scena come poligoni (vengono tagliati ad una certa distanza dalla luce) e vengono processati come tutti gli altri poligoni, eccetto che sono invisibili. Durante la scan conversion (con un algoritmo scan-line), quando viene incontrato un tale poligono invece che colorare il pixel corrispondente, si incrementa/decrementa un contatore per pixel. Il risultato è una mappa che per ogni pixel dice se è in luce (= 0) oppure in ombra (> 0)
- In OpenGL si fa in tre passate, con un trucco che usa z-buffer e stencil buffer.
- Varianti per velocizzare:
  - calcolare le silhouette delgli oggetti ed usarle per gli shadow volume (invece dei singoli poligoni).
  - creare gli shadow volumes solo degli oggetti “importanti”.




From wikipedia

# Ombre sfumate

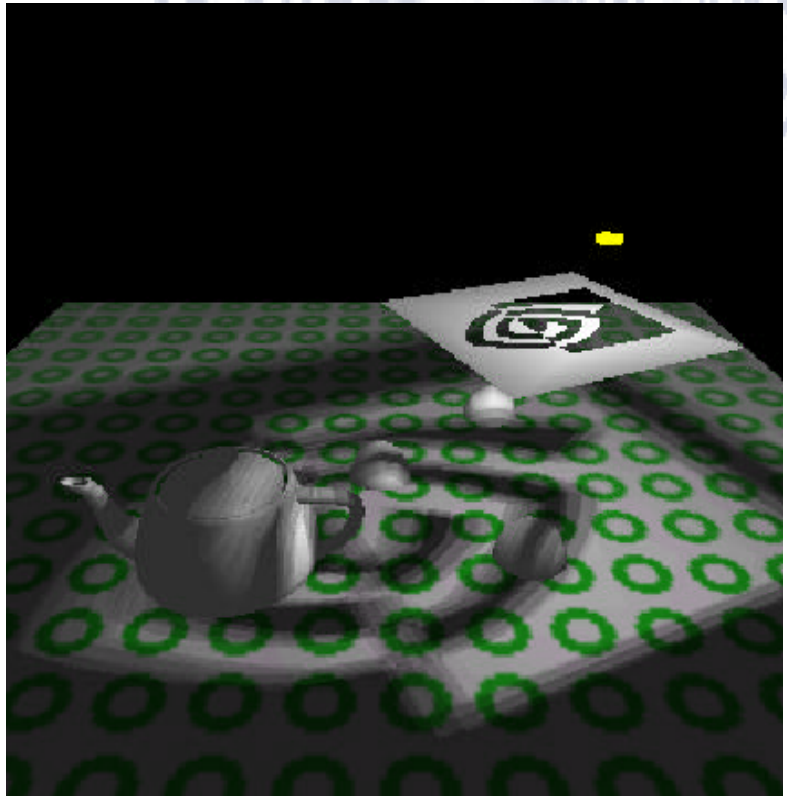


- Si possono generare con più sorgenti puntiformi

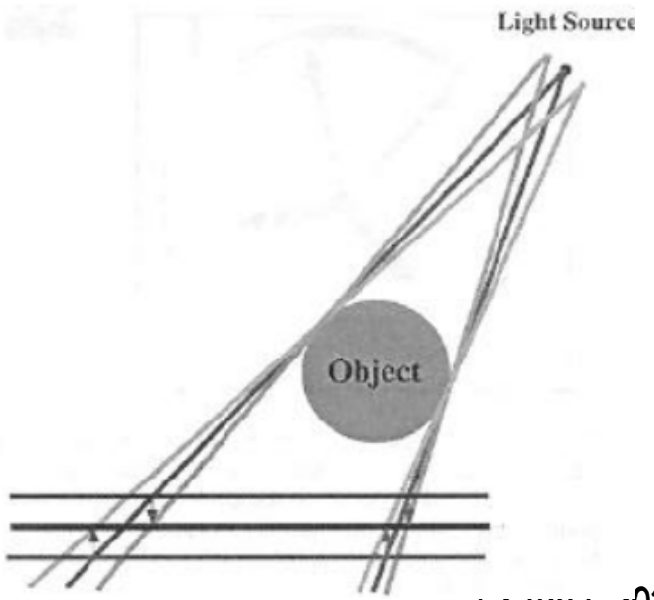
# Soft shadows

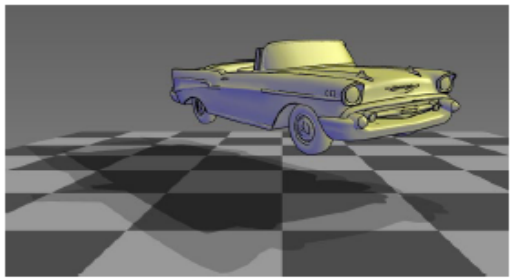
- Uso di blending additivo per accumulare i risultati di varie passate.
- Ci voglioni varie sorgenti per simulare bene la sfumatura
- La complessità dipende dal numero di campioni (luci puntiformi) usate per approssimare la luce allargata
- Si può sfumare per post processing di immagine (filtro convolutivo)
- Si può fare con tecniche differenti



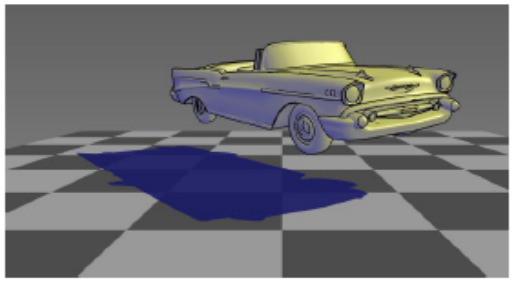
# Altro metodo (Gooch et al.)

- Idea: spostare il piano anziché la luce
- Si mediano i risultati
- Non occorrono più matrici di proiezione
- "Interactive technical illustration"
  Gooch et al. I3D 1998

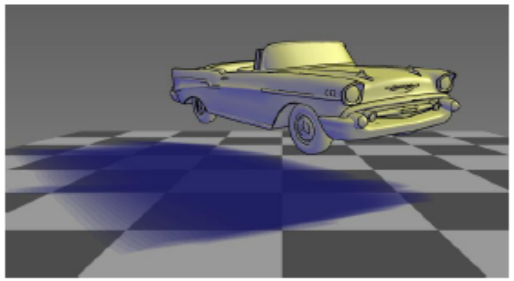


(a) Hard penumbra and hard umbra.



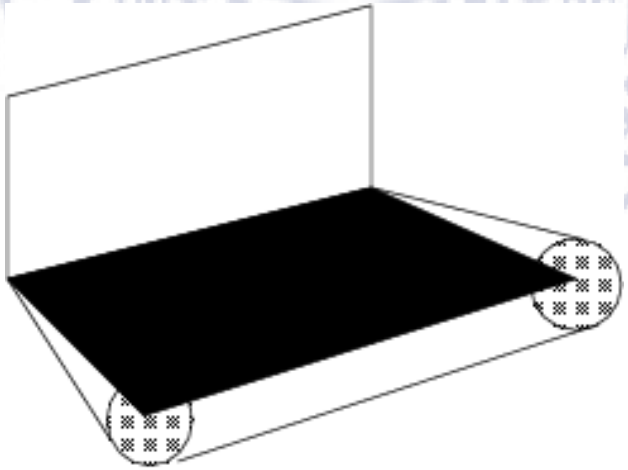
(b) Single hard, colored shadow.



(c) Colored soft shadow.

# Altra idea

- Sfumatura dei bordi con larghezza proporzionale all'altezza della silouhette (Haines 2001)



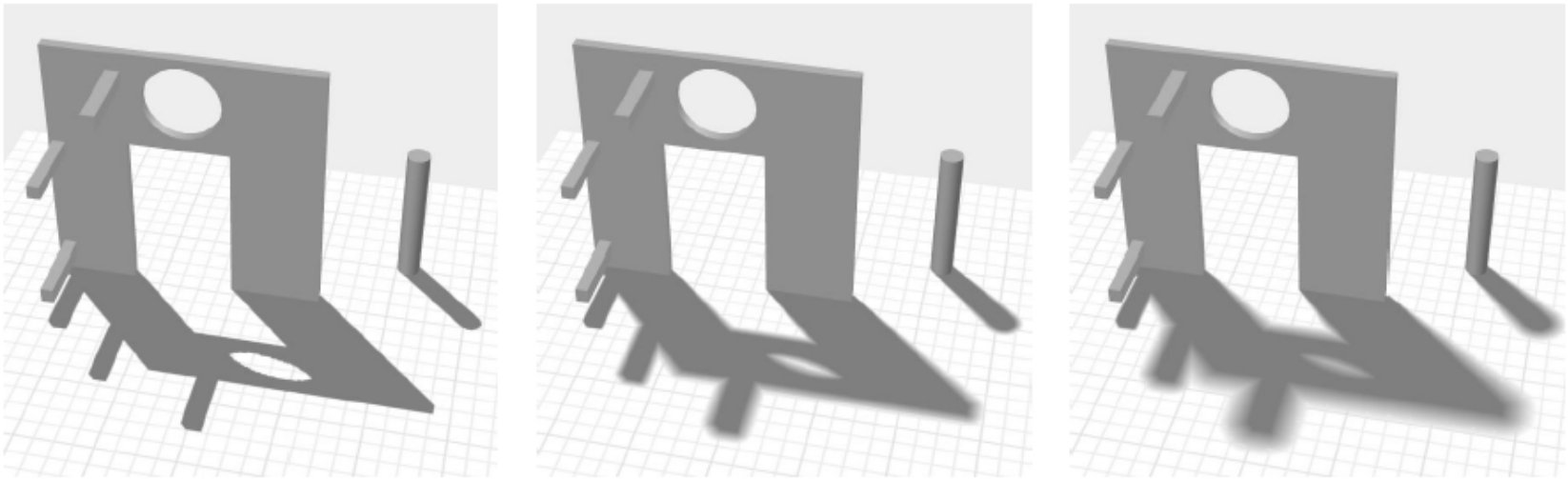


Figure 3: On the left is a hard shadow, the middle shows the effect of a small area light source, the right a larger light source.

# Domande di verifica

- Quali sono le proprietà delle ombre proiettate?
- Come si può implementare con la pipeline grafica la tecnica dello shadow volume?
- Quali sono i limiti delle varie tecniche di generazione delle ombre?