



Rendering della luce: Lighting, Shading, Ray tracing e Radiosity

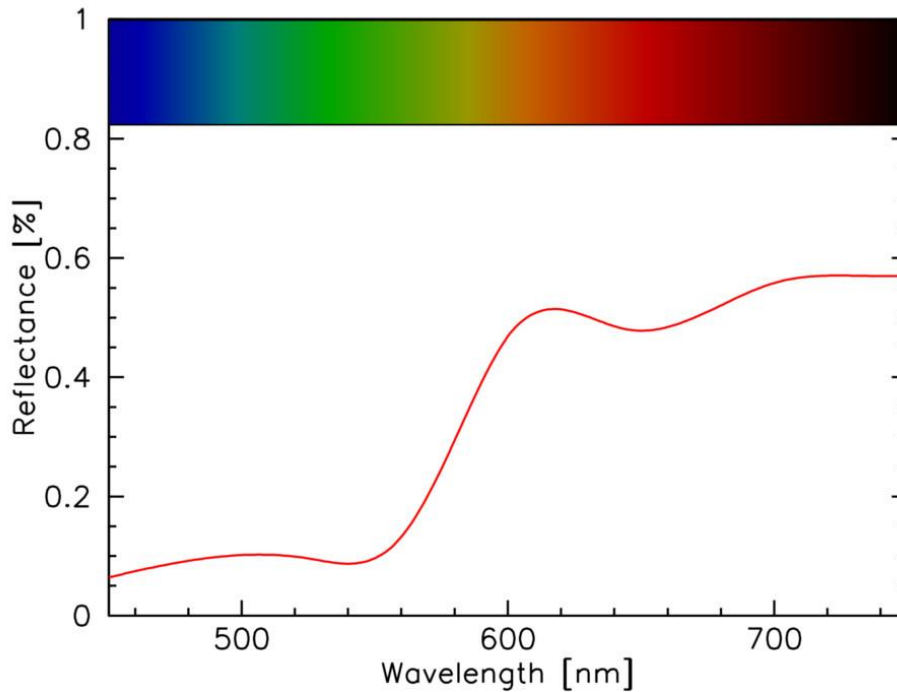
Texture mapping

Modello fisico dell'illuminazione

- La luce è una radiazione elettromagnetica che interagisce con la materia.
- Il termine luce si riferisce alla porzione dello spettro elettromagnetico visibile dall'occhio umano (tra 400 e 700 nanometri di lunghezza d'onda).
- La luce è caratterizzata da una distribuzione spettrale di potenza che determina il colore.
- La luce (senza entrare nei dettagli) trasporta una energia che ne determina l'intensità.
- La fotometria studia l'azione della luce visibile sull'occhio umano.

Modello fisico dell'illuminazione

- Il colore di un oggetto che noi osserviamo è dovuto alla radiazione riflessa dall'oggetto. Per esempio l'immagine qui riportata mostra lo spettro di una pallina rossa



Un'equazione di illuminazione globale

- La formulazione matematica dell'equazione del trasporto dell'energia luminosa è complessa (non vediamo come si ricava).
- Equazione integrale di rendering (Kajiya, 1987)

$$L(x, \omega) = L_e(x, \omega) + \int_{\Omega} \rho(x, \omega_i, \omega) L(x, \omega_i) (\omega_i \cdot \mathbf{n}) d\omega_i$$



luce emessa

luce riflessa

- La luce visibile in un punto della scena è data dalla somma della luce riflessa più la luce emessa
- La luce riflessa è un integrale che tiene conto di tutte le sorgenti di luce nell'ambiente e della loro orientazione rispetto al punto
- Risolvere in modo esatto l'equazione di rendering è molto oneroso
- E' una formulazione ricorsiva che dipende da tutte le luci e anche da tutte le superfici presenti nella scena

Un'equazione di illuminazione globale

$$L(x, \omega) = L_e(x, \omega) + \int_{\Omega} \rho(x, \omega_i, \omega) L(x, \omega_i) (\omega_i \cdot \mathbf{n}) d\omega_i$$

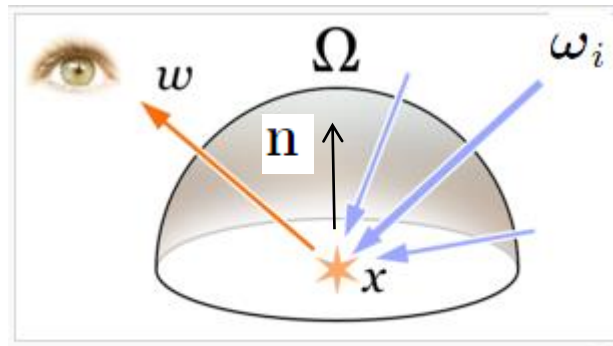
x punto sulla superficie in cui si calcola l'equazione

ω direzione che unisce il punto alla posizione dell'osservatore

ω_i direzione di provenienza del raggio incidente

$\rho(x, \omega_i, \omega)$ funzione che determina la frazione riflessa di luce incidente

$\omega_i \cdot \mathbf{n}$ coseno dell'angolo di incidenza rispetto alla normale alla superficie

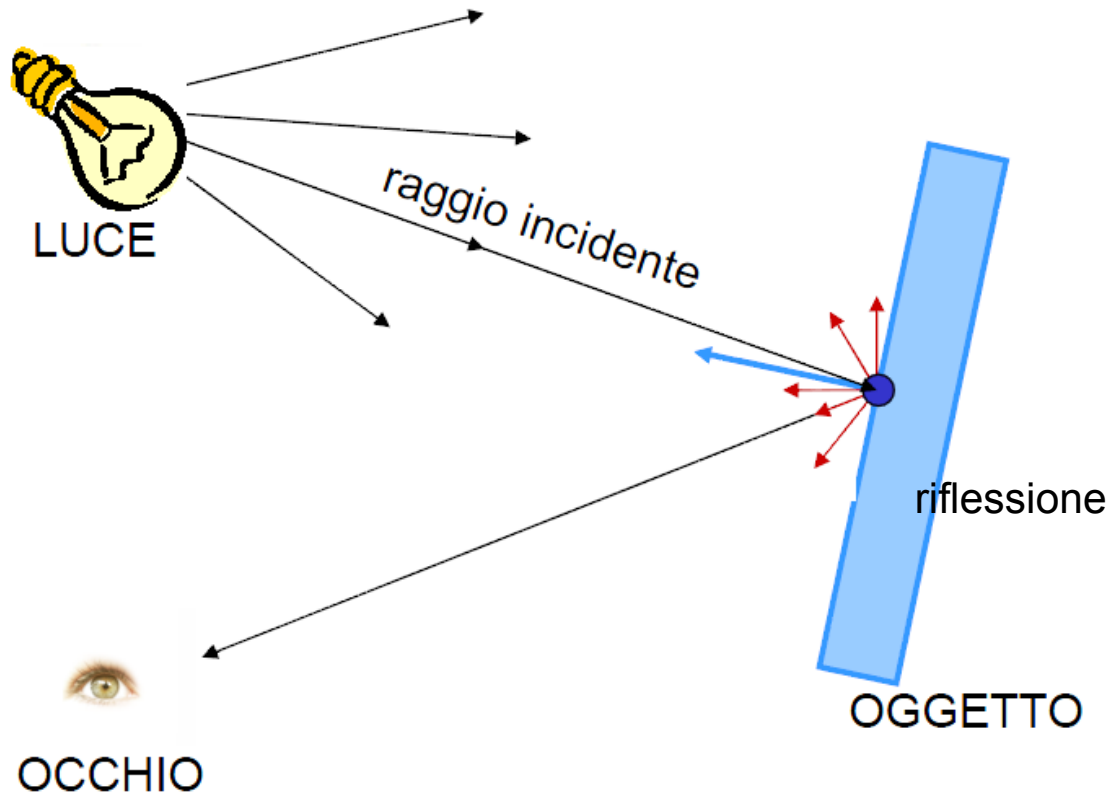


illuminazione in computer grafica

- In computer grafica sono stati elaborati dei modelli semplificati della equazione di rendering che consentono di ottenere risultati “realistici” per applicazioni real-time
- I metodi di illuminazione utilizzati in computer grafica si suddividono in:
 - **MODELLI LOCALI** (Phong, Blinn-Phong (usato in OpenGL)): tengono conto solo dell'effetto diretto delle sorgenti luminose trascurando le interriflessioni tra oggetti. Pertanto si elimina completamente la ricorsione nell'equazione della radianza. Ciascun punto viene trattato indipendentemente dal resto della scena
 - **MODELLI GLOBALI**: tengono conto della natura ricorsiva della equazione della radianza. Si trascurano alcuni fenomeni di interriflessione per rendere il problema trattabile.
 - **Ray tracing** che modella correttamente le riflessioni speculari.
 - **Radiosity** che modella solo le inter-riflessioni tra superfici diffusive.

illuminazione in computer grafica

Modelli locali



Tengono in considerazione:

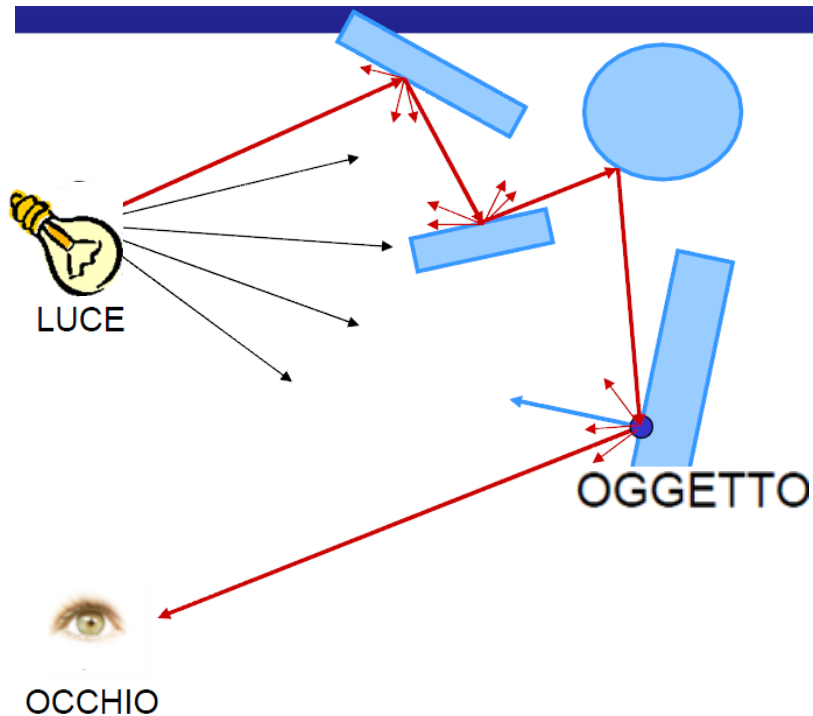
- condizioni di luce
 - sorgenti puntiformi
 - posizione sorgenti
 - colore sorgenti, intensità luce
- superficie da illuminare
 - orientamento (normale)
 - proprietà dei materiali

illuminazione in computer grafica

Modelli globali

Tengono in considerazione:

- riflessioni multiple
- ombre
- rifrazione
- ...



illuminazione, modelli locali

- **I colori degli oggetti sono il risultato del loro modo di reagire alla presenza luce**
- **L'effetto risultante del colore dipende:**
 - Presenza di luce di fondo ambientale
 - Presenza di sorgenti luminose che li illuminano
 - Reazione del materiale/superficie degli oggetti quando investiti dalla luce

Obiettivo:

valutazione semplificata dell'interazione tra sorgente luminosa ed entità illuminate

illuminazione, modelli locali

Modello di **Phong** (metà anni '70)

- La luce si considera composta da tre componenti cromatiche R,G,B. Si calcola l'intensità come sovrapposizione delle tre componenti cromatiche. Nel seguito faremo riferimento ad una singola componente.
- Sorgenti puntiformi, no inter-riflessioni.
- Modella la riflessione ma non la rifrazione.
- Solo materiali opachi, non trasparenti o semi-trasparenti, no ombre
- Se sono presenti più sorgenti luminose che illuminano un punto, si applica il principio di sovrapposizione degli effetti sommando i contributi delle singole sorgenti che si calcolano indipendentemente l'uno dall'altro.
- L'intensità dovuta alla singola sorgente I^{out} viene calcolata come somma di quattro componenti: luce ambientale, luce diffusa, luce riflessa e luce emessa:



$$I^{out} = I_a^{out} + I_d^{out} + I_s^{out} + I_e^{out}$$

illuminazione, modelli locali

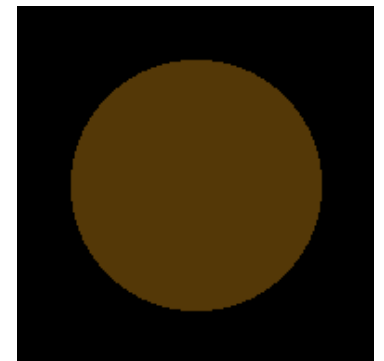
Componente **AMBIENTALE**

- i modelli locali non simulano riflessioni multiple della luce (inter-riflessioni). Pertanto se un punto di un oggetto non è rivolto verso la luce, allora risulta completamente nero.
- Per simulare in modo grossolano luce indiretta che arriva anche da sorgenti nascoste si introduce nel modello locale un termine che prende il nome di luce ambientale I_a .
- Alla superficie viene assegnato un coefficiente di riflessione ambientale k_a compreso tra 0 ed 1 (in generale uno per ogni componente RGBvedi esercitazione OpenGL).
- Il contributo all'intensità finale del punto da parte della luce ambientale è dato da:

$$I_a^{out} = I_a k_a$$

← coefficiente di riflessione ambiente
(proprietà del materiale), in $[0,1]$

Intensità luce ambientale (costante)



illuminazione, modelli locali

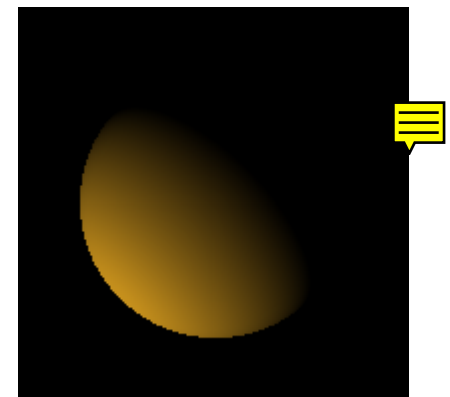
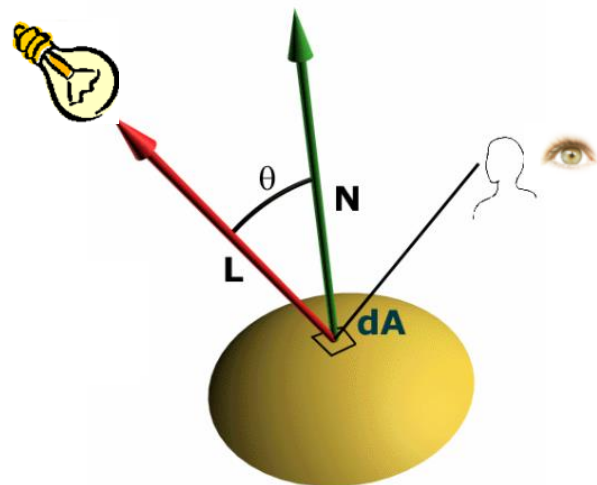
Componente **DIFFUSA** o di riflessione Lambertiana

- Non dipende dalla direzione di vista V , ma solo dalla normale N e da L (vettore normalizzato che congiunge il vertice della superficie con la sorgente di luce, "vertex to light")
- Simula il comportamento di alcuni materiali (per esempio il gesso, o il legno) che riflettono la luce che ricevono in tutte le direzioni

$$I_d^{out} = I k_d \max(0, \cos \theta) = I k_d \max(0, (\bar{N} \cdot \bar{L}))$$

intensità sorgente luminosa

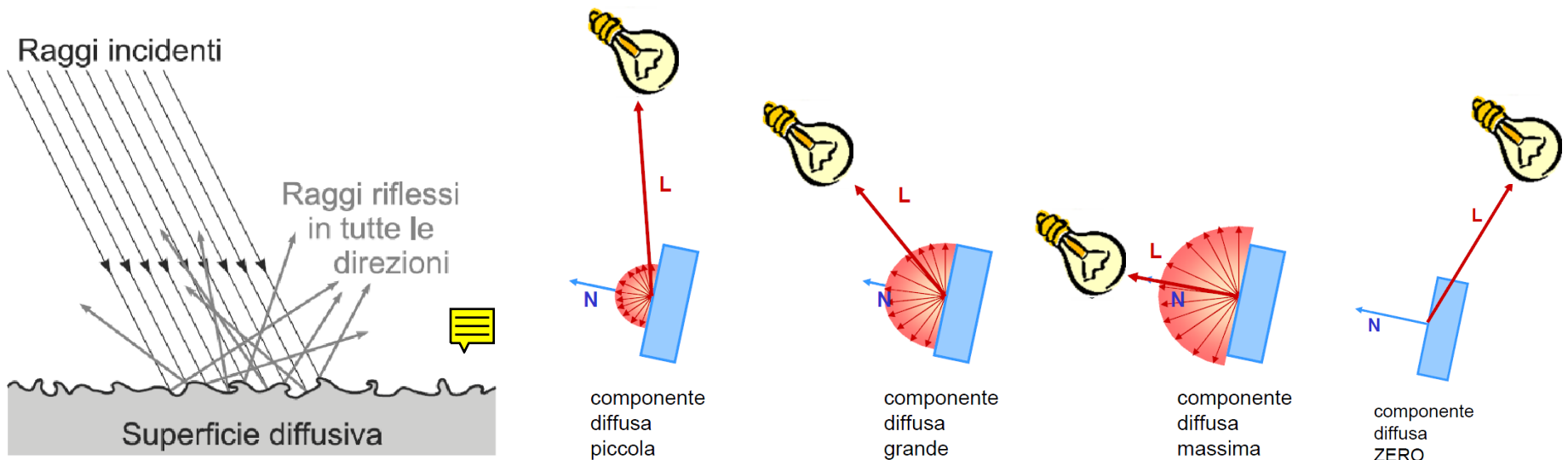
coefficiente di riflessione diffusa
(proprietà del materiale), in $[0,1]$



illuminazione, modelli locali

Componente **DIFFUSA** o di riflessione Lambertiana

- Si basa sul principio della legge di Lambert per materiali opachi **ideali**
- Tali materiali a livello microscopico hanno una rugosità che causa la riflessione della luce in tutte le direzioni
- L'intensità della luce diffusa è proporzionale all'angolo tra la direzione del raggio incidente e la normale alla superficie in quel punto.

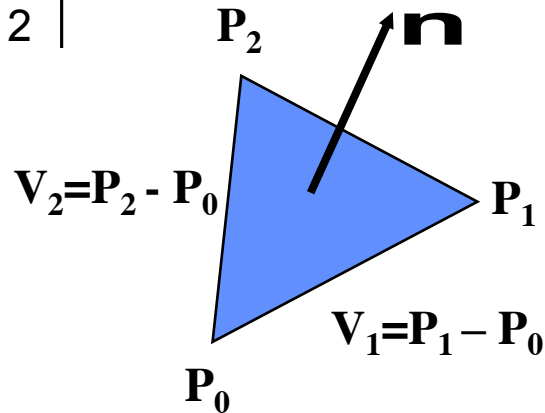


illuminazione, modelli locali

- Il vettore (versore) normale ad una superficie planare (racchiusa da poligono) è diretto come il prodotto vettoriale di due spigoli adiacenti del poligono



$$\mathbf{n} = \frac{\mathbf{v}_1 \times \mathbf{v}_2}{|\mathbf{v}_1 \times \mathbf{v}_2|}$$



illuminazione, modelli locali

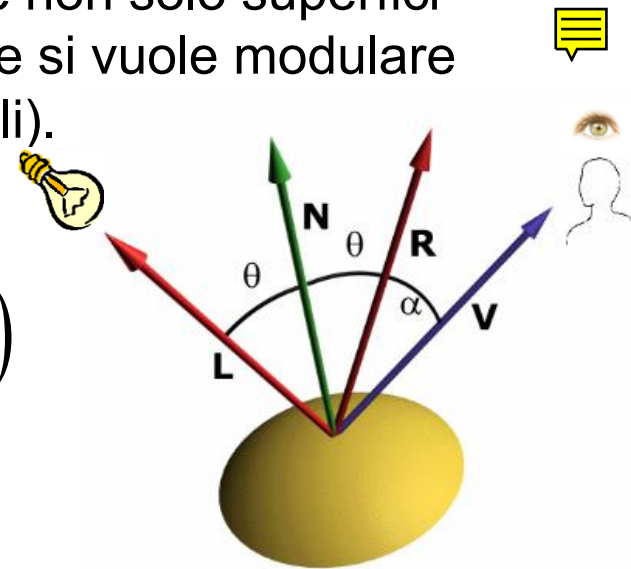
Componente **SPECULARE**

- Si basa sui principi dell'ottica geometrica che prevede che la luce venga riflessa in una direzione **R** che è ottenuta “specchiando” **L** rispetto a **N**
- Dipende dalla posizione dell'osservatore, l'osservatore percepisce una luce riflessa tanto più intensa quanto più α (angolo tra **V** e **R**) è piccolo (intensità massima quando $\alpha=0$ ovvero **V** ed **R** sono allineati, quindi $\mathbf{V} \cdot \mathbf{R} = 1$)
- **V** è la direzione di vista (versore dal vertice all'osservatore, “vertex to eye”)
- Con la componente speculare si vogliono modellare non solo superfici riflettenti ideali ma anche riflettori non perfetti. Inoltre si vuole modulare l'effetto di lucidità (*glossy*) della superficie (es:metalli).
Si introduce quindi la seguente formulazione:

$$I_s^{out} = I k_s \max(0, \cos^n \alpha) = I k_s \max(0, (\bar{\mathbf{R}} \cdot \bar{\mathbf{V}})^n)$$

↑
coefficiente di riflessione speculare
(proprietà del materiale), in $[0,1]$

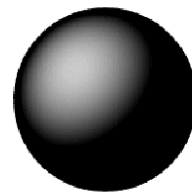
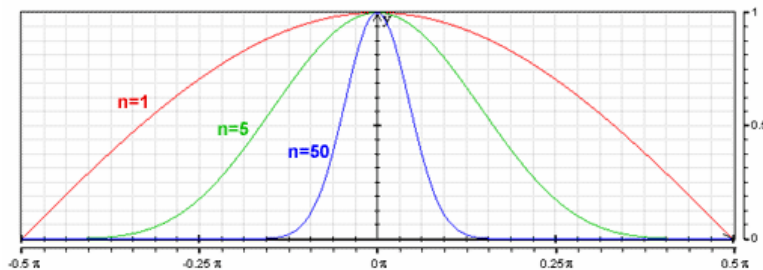
esponente di riflessione
speculare del materiale



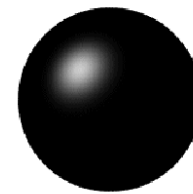
illuminazione, modelli locali

Componente SPECULARE

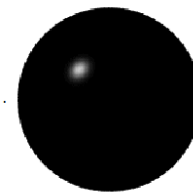
- L'esponente n modula la lucidità della superficie. Se n tende all'infinito si ottiene una riflessione speculare perfetta (più brillante)



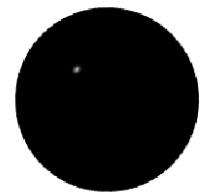
$n = 1$



$n = 5$



$n = 10$



$n = 100$

$$I_s^{out} = I k_s \max(0, \cos^n \alpha) = I k_s \max(0, (\bar{R} \cdot \bar{V})^n)$$

coefficiente di riflessione speculare
(proprietà del materiale), in $[0,1]$

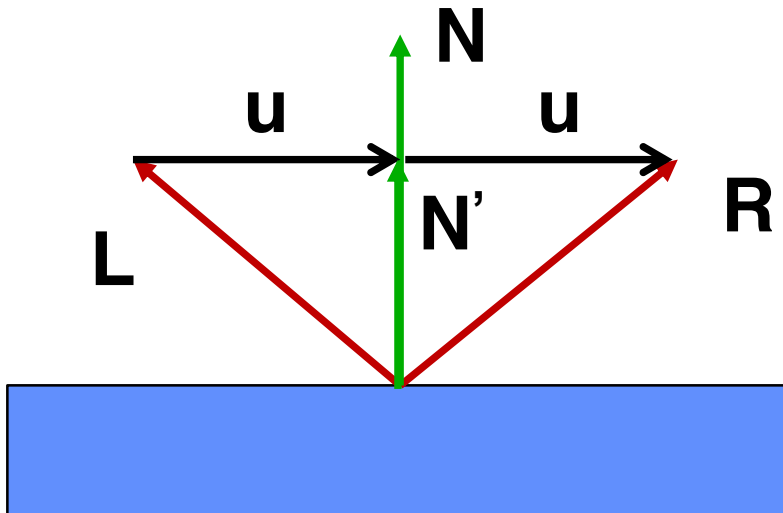
esponente di riflessione
speculare del materiale



illuminazione, modelli locali

Componente **SPECULARE**

- Calcolo del vettore normalizzato (versore) riflesso **R** di **L** rispetto a **N**



$$N' = (N \cdot L)N$$

$$u = N' - L$$

$$R = N' + u = 2N' - L = 2(N \cdot L)N - L$$

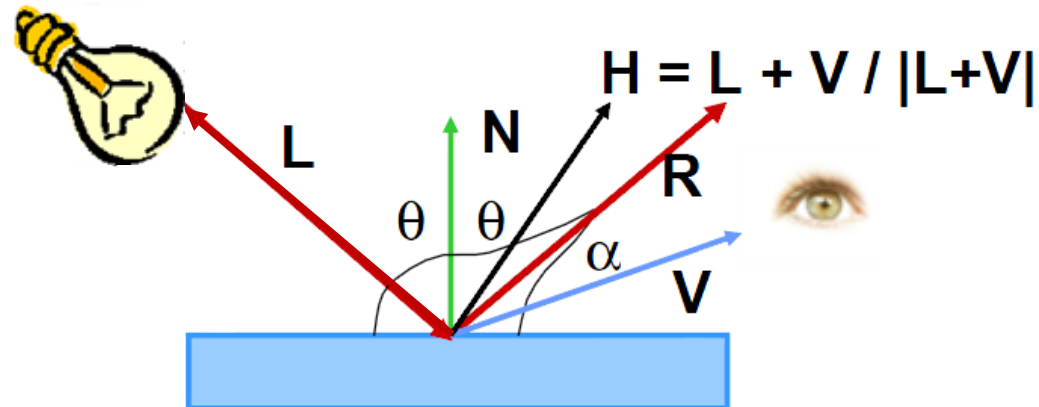
illuminazione, modelli locali

Componente **SPECULARE**, modello semplificato di **Blinn-Phong**

- Esiste una formulazione semplificata della riflessione speculare (utilizzata in OpenGL) che fa uso del *halfway vector* H , il vettore normalizzato che biseca l'angolo formato da L e V .

$$I_s^{out} = I k_s \max\left(0, (\bar{N} \cdot \bar{H})^n\right) \text{ dove } H = \frac{L + V}{\|L + V\|}$$

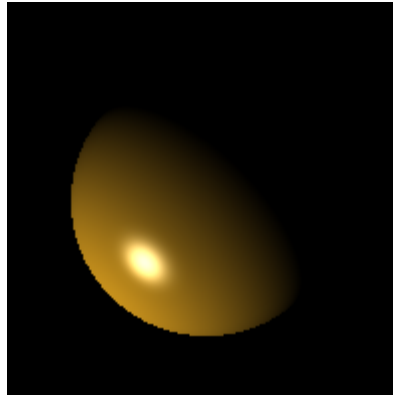
- vantaggio (efficienza): quando la luce e l'osservatore sono distanti dall'oggetto H si può considerare costante per tutti i punti di un oggetto.



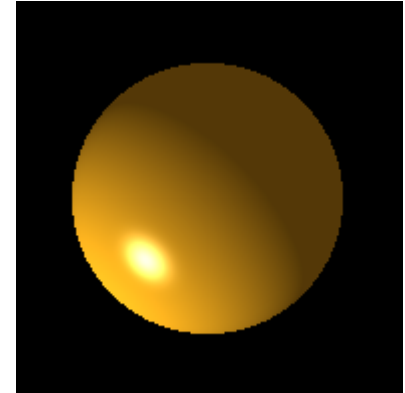
illuminazione, modelli locali



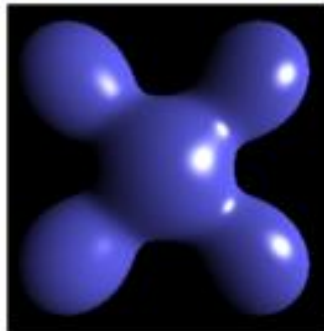
ambientale+diffusiva



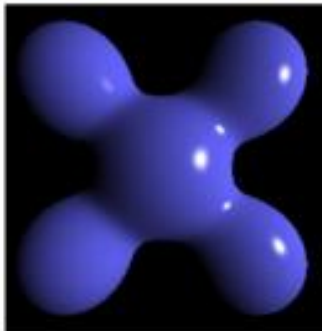
diffusiva+speculare



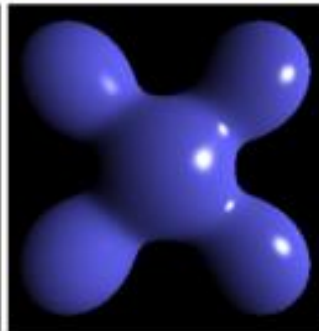
ambientale+diffusiva+speculare



Blinn-Phong



Phong



Blinn-Phong
(Lower Exponent)

illuminazione, modelli locali

Modello di **Phong** (metà anni '70)

- Sommando i contributi di luce ambientale diffusiva e speculare si ottiene l'espressione dell'intensità della luce delle tre componenti R,G,B di ciascuna sorgente luce (omesso il clamping)
$$I^{r,out} = I_a^r k_a^r + I^r (k_d^r (\bar{N} \cdot \bar{L}) + k_s^r (\bar{N} \cdot \bar{H})^n)$$
$$I^{g,out} = I_a^g k_a^g + I^g (k_d^g (\bar{N} \cdot \bar{L}) + k_s^g (\bar{N} \cdot \bar{H})^n)$$
$$I^{b,out} = I_a^b k_a^b + I^b (k_d^b (\bar{N} \cdot \bar{L}) + k_s^b (\bar{N} \cdot \bar{H})^n)$$
- Il colore di un vertice si determina assegnando le proprietà del materiale, ovvero i coefficienti k . Solitamente si pongono uguali i coefficienti ambientale e di diffusione di un materiale, mentre il coefficiente di riflessione speculare si pone dello stesso colore della luce incidente (solitamente bianco).
- il modello di Phong prevede che ogni sorgente di luce abbia tre componenti di intensità separate: luce speculare I_s , luce diffusa I_d e luce ambientale I_a che si utilizzano nelle equazioni precedenti. E' un modello flessibile che non ha significato fisico.
- Nel modello definito nelle slides precedenti abbiamo assunto $I = I_s = I_d$
- Per l'equazione di rendering completa usata in OpenGL si veda l'esercitazione sull'illuminazione

illuminazione, modelli locali

Componente **EMISSIVA**

- L'oggetto può possedere una componente di luce propria (es: lampadina, LED)
- E' una componente additiva, costante su tutto l'oggetto
- Non illumina gli altri oggetti della scena

$$I_e^{out} = k_e$$

illuminazione, modelli locali

Attenuazione della luce (solo per luci posizionali, vedi prossima slide)

- Si introduce un coefficiente di attenuazione della intensità della luce per ottenere risultati più realistici. Il coefficiente di attenuazione decresce con il quadrato della distanza sorgente oggetto.

$$f_{att} = \min \left(\frac{1}{c_1 + c_2 d_L + c_3 d_L^2}, 1 \right)$$

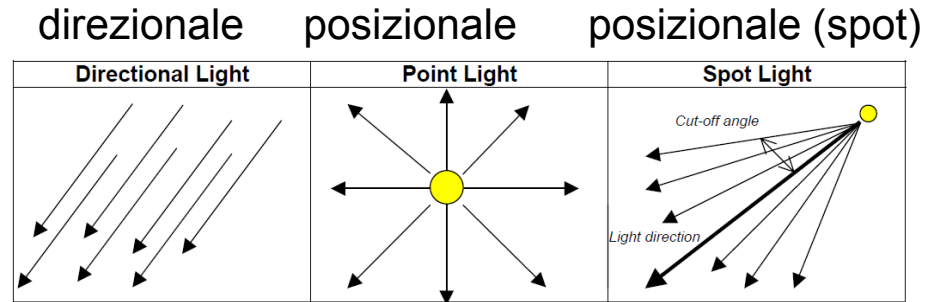
- L'espressione dell'intensità di un vertice dovuta ad una sorgente di luce diventa

$$I^{out} = I_e^{out} + f_{att} \left(I_a k_a + I_d k_d \max \left(0, (\bar{N} \cdot \bar{L}) \right) + I_s k_s \max \left(0, (\bar{N} \cdot \bar{H})^n \right) \right)$$

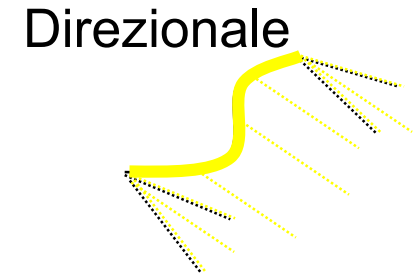
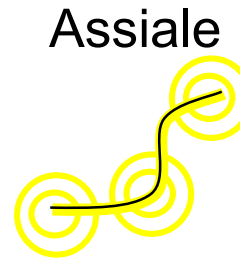
$$\text{dove } H = \frac{L + V}{\|L + V\|}$$

Tipi di sorgenti di luce

- **Sorgente puntiforme**
(OpenGL)

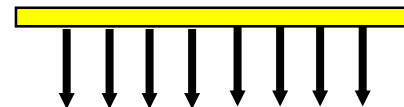


- **Sorgente filiforme**



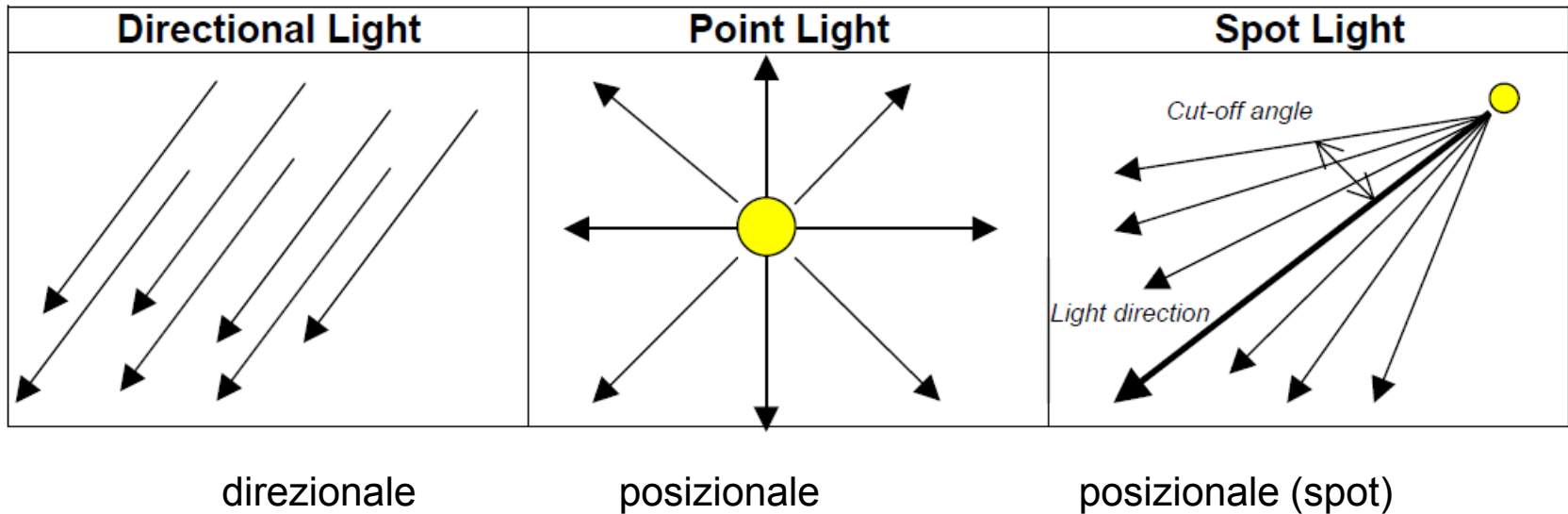
- **Sorgente superficiale**

Mono/Multidirezionale
(classificazione più complessa)



Tipi di sorgenti di luce

• Sorgente puntiforme (OpenGL)



- luce posta all'infinito
- raggi paralleli
- definita da una direzione e una intensità

- definita da una posizione e da una intensità

- luce tipo faretto direzionale
- definita da direzione, angolo di apertura e intensità

Verso l'illuminazione globale...

- **Nel mondo reale:**

- Oggetti che si interpongono tra luce e altri oggetti creano ombre su questi!
- Se un oggetto è riflettente, la sua luce riflessa illumina altri oggetti della scena

- **Limite del modello di illuminazione:**

- E' locale
- Non calcola le ombre
- Non calcola la riflessione diffusiva tra gli oggetti
- Non calcola la riflessione speculare di radiazioni tra gli oggetti

- **Necessità di tecniche aggiuntive per il rendering globale**

- **RAY TRACING**

- Per risolvere problemi di punto di vista e luci nella scena, trasparenza, riflessioni speculari tra oggetti, ombre (dipendente dal punto di vista)

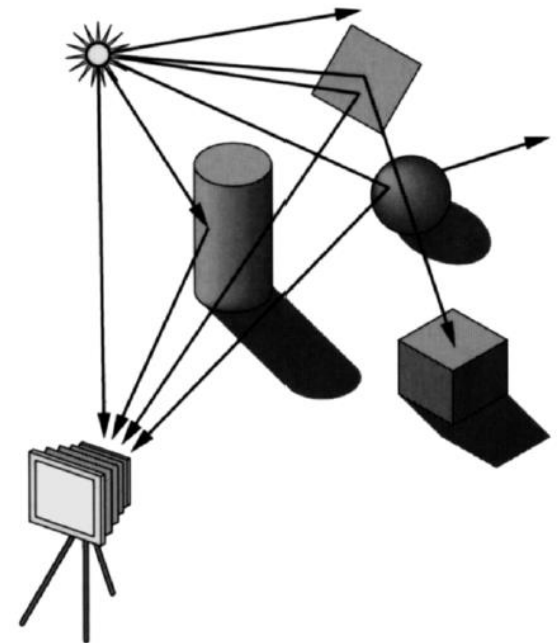
- **RADIOSITY**

- Per sorgenti luminose di forma qualsiasi
- Modello globale che tenga conto delle inter-riflessioni diffuse tra oggetti
- Indipendente dal punto di vista

illuminazione, modelli globali

Ray Tracing (Whitted, 1980)

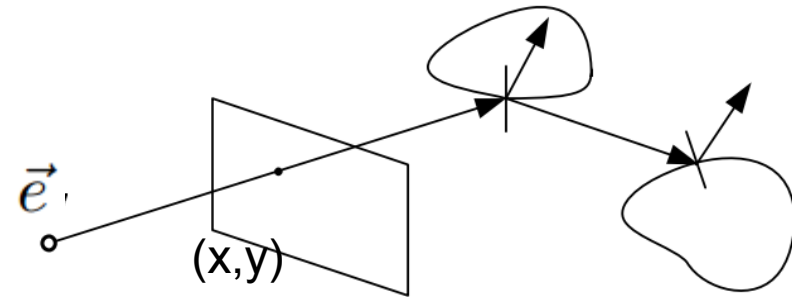
- tecnica ricorsiva, estensione del ray casting
- Si tracciano i raggi ottici uscenti dal centro di proiezione che colpiscono i pixel sul piano immagine
- Se si verifica una intersezione di un raggio con un oggetto si applica sul vertice dell'oggetto un modello di illuminazione locale (es: Phong) tenendo conto del contributo di tutte le sorgenti di luce visibili dal vertice.
- Si tracciano poi altri due raggi secondari del raggio ottico a partire dal vertice: il raggio riflesso e il raggio rifratto (se l'oggetto è trasparente) per raccogliere altri contributi di illuminazione sul vertice
- Si entra in un procedimento ricorsivo che termina quando il raggio si perde all'infinito, oppure quando si raggiunge un limite massimo di iterazioni, oppure quando l'intensità del raggio in esame scende sotto una soglia
- Modello adatto per superfici trasparenti e riflettenti
- Non tiene conto della componente diffusiva della illuminazione indiretta.



illuminazione, modelli globali

Ray Tracing (Whitted, 1980)

- un generico raggio (viewing ray) viene tracciato dal view-point al pixel (x,y)
- il raggio colpisce il primo oggetto sul percorso
- si calcolano il raggio riflesso e il raggio rifratto
- si continua ricorsivamente a tracciare raggi riflessi e rifratti
- Nel ray tracing esistono 4 tipi di raggi:



1. **raggio ottico (viewing ray):** dal view-point ad un pixel
 2. **raggi d'ombra (shadow ray):** dai vertici sugli oggetti alle sorgenti di luce
 3. **raggi di riflessione (reflection rays):** riflessione del viewing ray
 4. **raggi di rifrazione (refraction rays):** rifrazione del viewing ray
- si calcola il valore dell'intensità di illuminazione sul vertice considerando tre contributi:
 1. illuminazione locale causata dalle sole luci che illuminano direttamente il vertice: I_{local}
 2. contributo sul vertice portato dal raggio riflesso del viewing ray (calcolato ricorsivamente): $I_{riflessione}$ **La direzione del raggio riflesso rappresenta la direzione di osservazione da utilizzare nell'iterazione successiva**
 3. contributo sul vertice portato dal raggio rifratto del viewing ray (calcolato ricorsivamente): $I_{rifrazione}$ **La direzione del raggio rifratto rappresenta la direzione di osservazione da utilizzare nell'iterazione successiva**

illuminazione, modelli globali

Ray Tracing (Whitted, 1980)

PSEUDOCODICE

Per ogni pixel dell'immagine {

Crea un raggio che, dall'occhio, passa attraverso questo pixel

Inizializza "NearestT" a "INFINITO" e "NearestObject" a "NULL"

Per ogni oggetto della scena {

Se il raggio colpisce questo oggetto {

Se la distanza "t" è meno di "NearestT" {

Setta "NearestT" a "t"

Setta "NearestObject" a questo oggetto } }

Se "NearestObject" è "NULL" {

Colora questo pixel con il colore di sfondo

} Altrimenti {

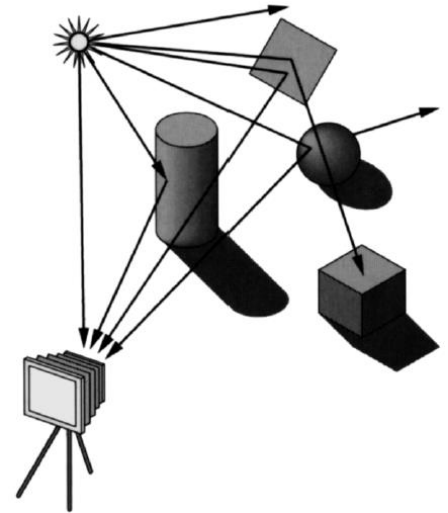
Spedisci un raggio ad ogni sorgente di luce per testare se è in ombra

Se la superficie è riflettente, genera il raggio riflesso: (ricorsione)

Se la superficie è trasparente, genera il raggio rifratto: (ricorsione)

Colora questo pixel dall'accumulazione dei tre contributi (come? vedi prossima slide)

}}



illuminazione, modelli globali

Ray Tracing (Whitted, 1980)

modello di illuminazione di Whitted per il ray-tracing

$$I^{out} = I_{local} + I_{riflessione} + I_{rifrazione}$$

$$= \sum_{i=0}^k S_i f_{att_i} I_i \left(k_d \max\left(0, (\bar{N} \cdot \bar{L}_i)\right) + k_s \max\left(0, (\bar{N} \cdot \bar{H}_i)^n\right) \right) + k_r I_r + k_t I_t$$

- S_i =1 se il vertice non è in ombra rispetto alla sorgente di luce i -ma,
=0 se è in ombra,
[0..1] un valore intermedio se tra la luce e il vertice ci sono oggetti trasparenti
- I_r contributo di intensità portato dal raggio riflesso (calcolato ricorsivamente tramite la stessa equazione facendo partire il raggio riflesso del viewing ray dal vertice)
- I_t contributo di intensità portato dal raggio rifratto (calcolato ricorsivamente tramite la stessa equazione facendo partire il raggio rifratto del viewing ray dal vertice)
- k_r, k_t coefficienti <1

illuminazione, modelli globali

Ray Tracing (Whitted, 1980)

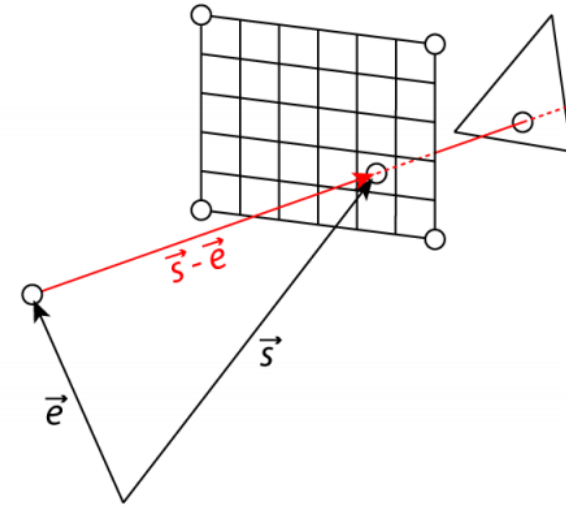
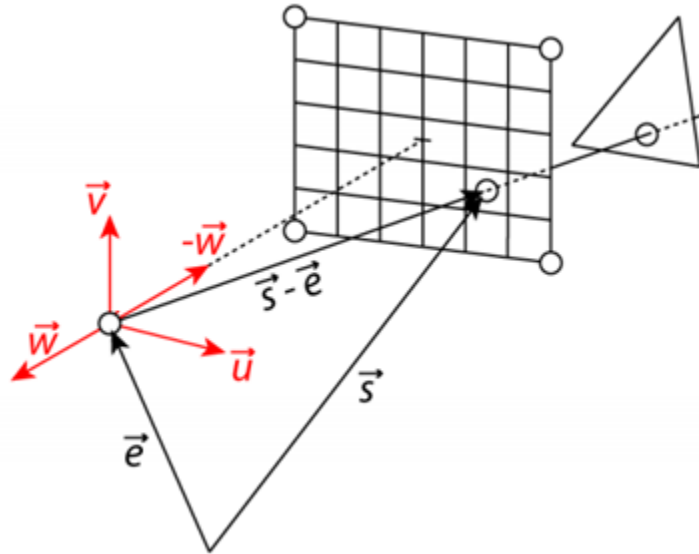
calcolo dei raggi ottici (viewing ray)

equazione parametrica del raggio dal viewpoint $\vec{e}$

$$\vec{p}(t) = \vec{e} + t(\vec{s} - \vec{e})$$

si definisce il sistema di riferimento $(\vec{u}, \vec{v}, \vec{w})$ del viewpoint:

- direzione di vista $-\vec{w}$



illuminazione, modelli globali

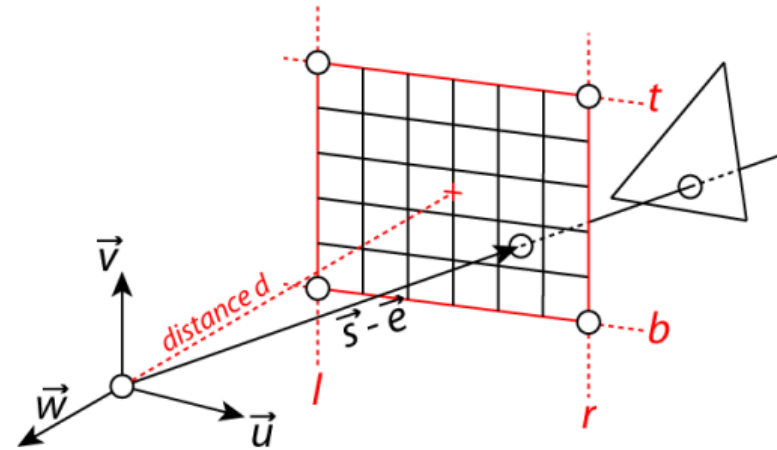
Ray Tracing (Whitted, 1980)

calcolo dei raggi ottici (viewing ray)

- si definiscono gli estremi della window:
 l, r, t, b
- la terna del viewpoint è a distanza $-w = d$ dal piano immagine
- assumendo una finestra di $n_x \times n_y$ pixel, un pixel (i, j) assume coordinate (u, v) pari a:

$$u = \frac{l + (r - l)(i + 0.5)}{n_x}$$

$$v = \frac{b + (t - b)(j + 0.5)}{n_y}$$



illuminazione, modelli globali

Ray Tracing (Whitted, 1980)

calcolo dei raggi ottici (viewing ray)

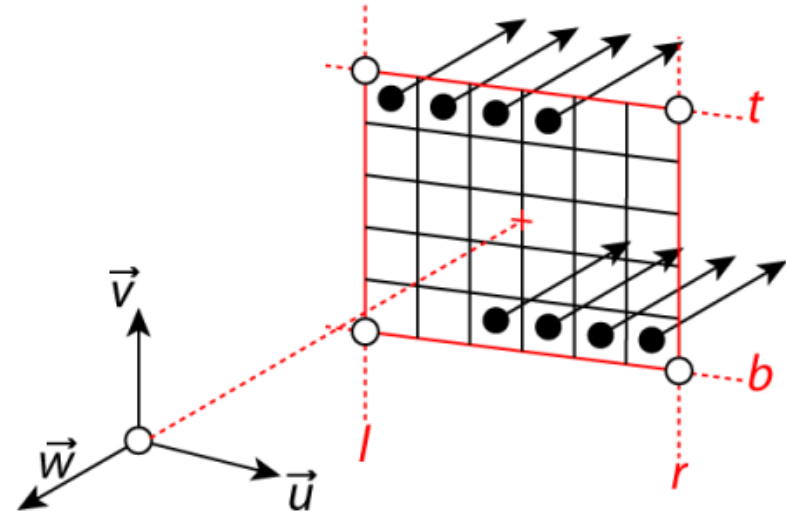
proiezioni ortogonali:

- i raggi hanno tutti la stessa direzione $-\vec{w}$
- ma diversa origine:

$$\vec{e} + u\vec{u} + v\vec{v}$$

il generico viewing ray
si può scrivere come:

$$\vec{p}(\alpha) = \vec{e} + u\vec{u} + v\vec{v} - \alpha\vec{w}$$



illuminazione, modelli globali

Ray Tracing (Whitted, 1980)

calcolo dei raggi ottici (viewing ray)

proiezioni prospettiche:

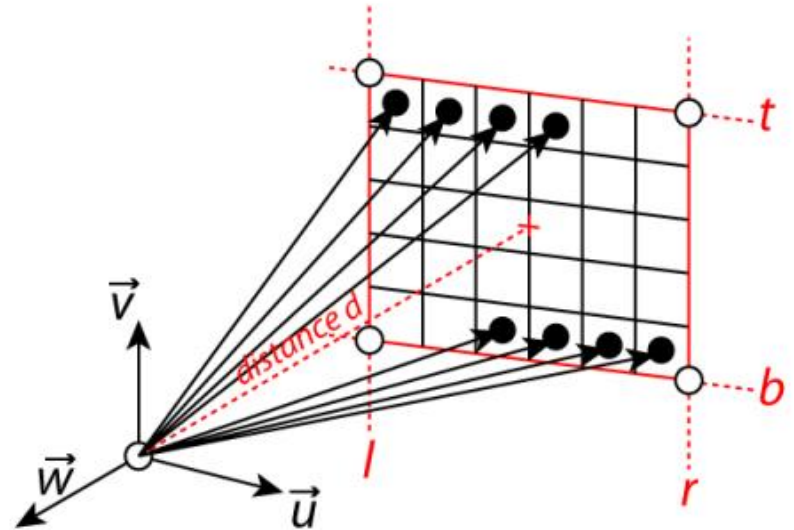
- i raggi hanno tutti la stessa origine $\vec{e}$
- ma diversa direzione:

$$u\vec{u} + v\vec{v} - d\vec{w}$$

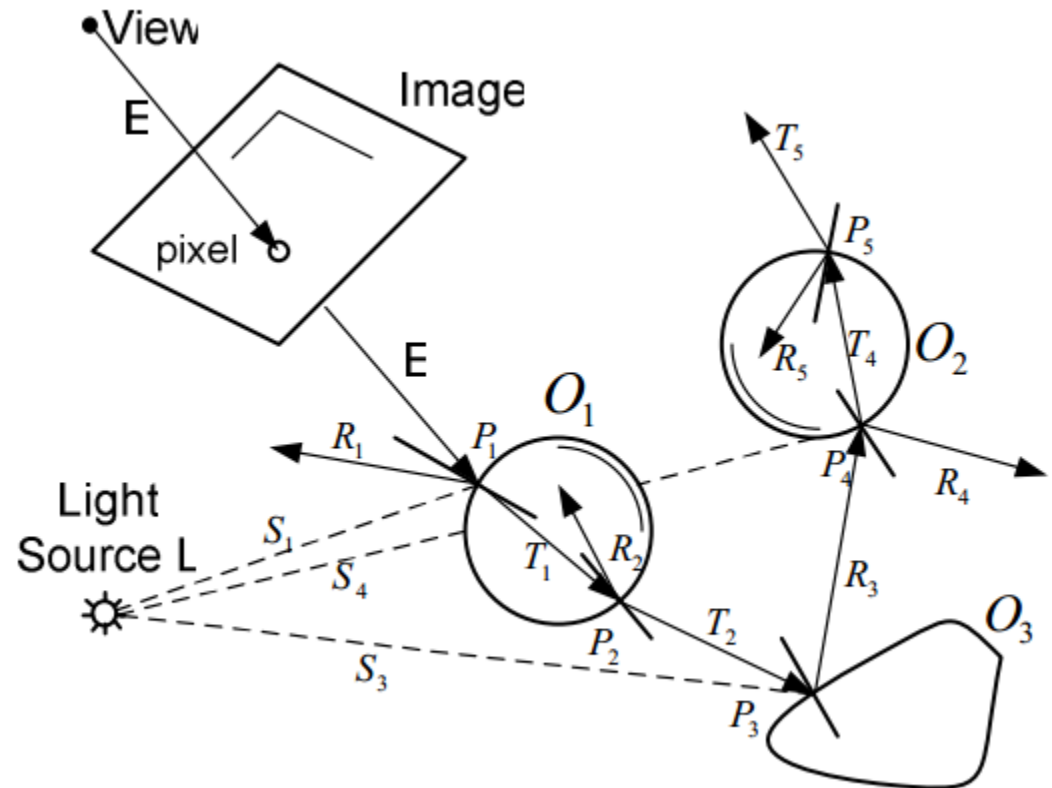
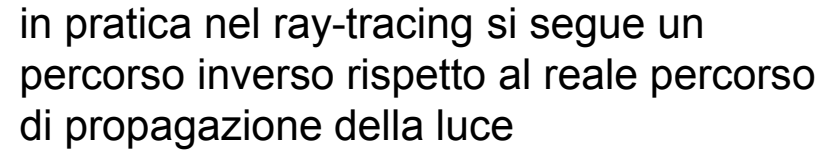
il generico viewing ray
si può scrivere come:

$$\vec{p}(\alpha) = \vec{e} + \alpha(u\vec{u} + v\vec{v} - d\vec{w})$$

Queste espressioni dei raggi ottici sono utili per calcolare le intersezioni dei raggi con gli oggetti della scena (vedi slides ray casting)

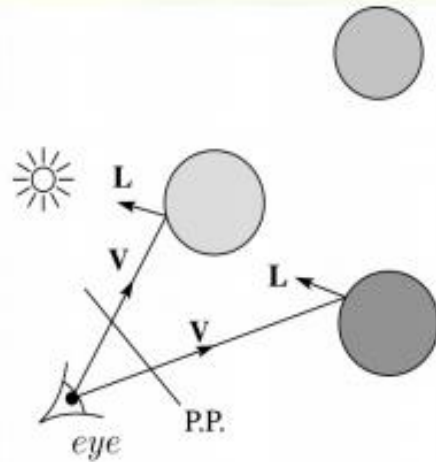


Ray Tracing

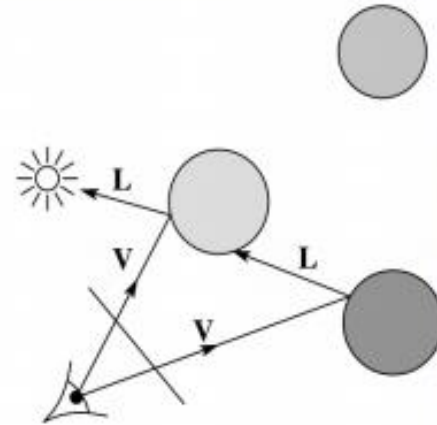


illuminazione, modelli globali

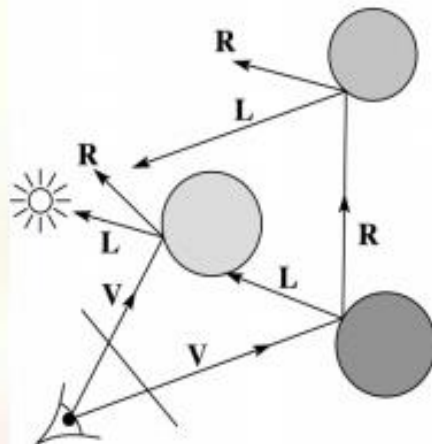
Ray Tracing



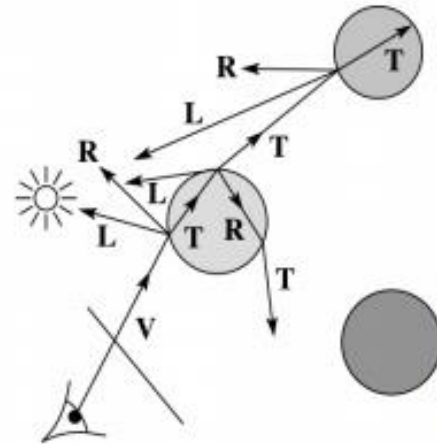
Primary rays



Shadow rays



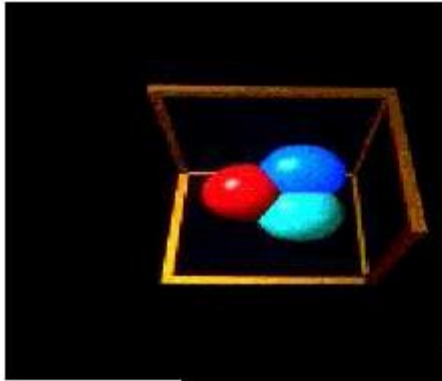
Reflection rays



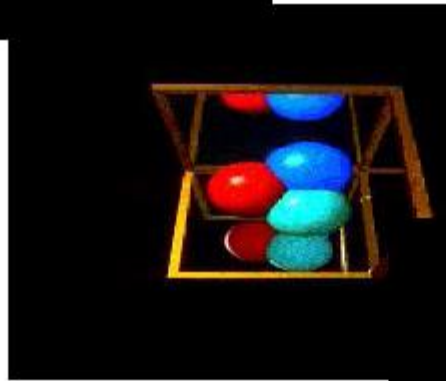
Refracted rays

illuminazione, modelli globali

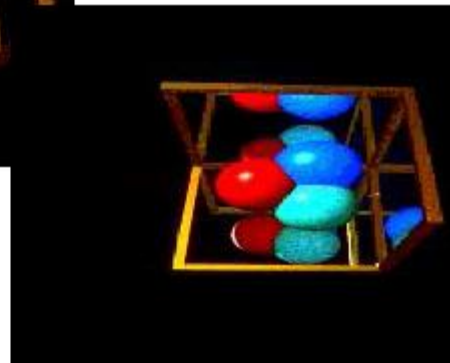
Ray Tracing



solo raggio primario



un solo rimbalzo



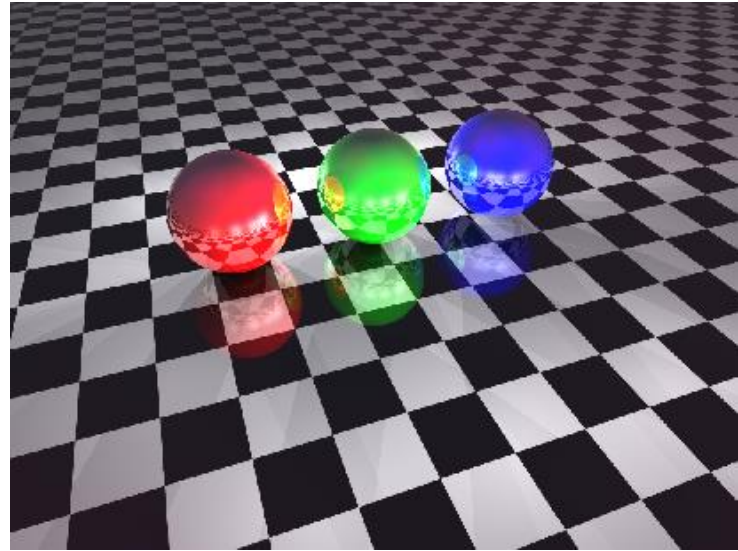
due rimbalzi

illuminazione, modelli globali

Ray Tracing



(Advanced Rendering Toolkit - Alexander Wilke - 1999)



illuminazione, modelli globali

Ray Tracing

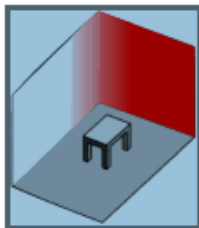
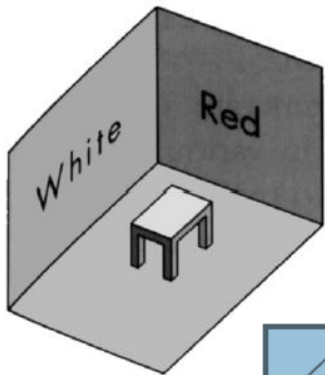
- computazionalmente costoso
- pesantemente basato su calcoli di intersezioni
- utilizzato principalmente per applicazioni non real-time
- non supportato in OpenGL

illuminazione, modelli globali



Radiosity (Goral et al. 1984)

- Tecnica per rendering di superfici perfettamente diffusive (lambertiane)
- Riflessione di radiazioni in tutte le direzioni
- Causa della diffusione di luce: rugosità “microscopica” della superficie
- Complementare rispetto al ray tracing
- Radiosità di una superficie: somma della radiosità emessa e di quella dovuta a riflessione diffusa
- Sorgenti ed oggetti trattati allo stesso modo
- Scena suddivisa in patches, pesante computazionalmente



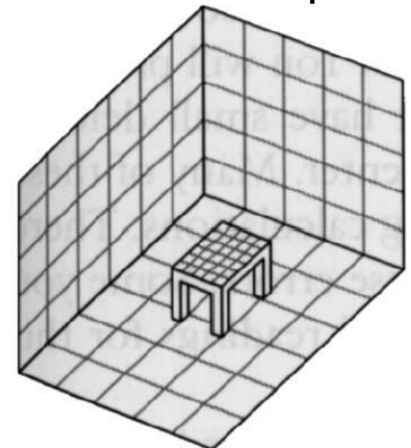
ESEMPIO

Stanza con pareti diffusive, una bianca ed una rossa:

La riflessione diffusa della parete rossa colpisce la parete bianca col risultato che luce di colore rosso si sovrappone alla luce bianca.

Idem per la parete bianca.

Suddivisione in patches



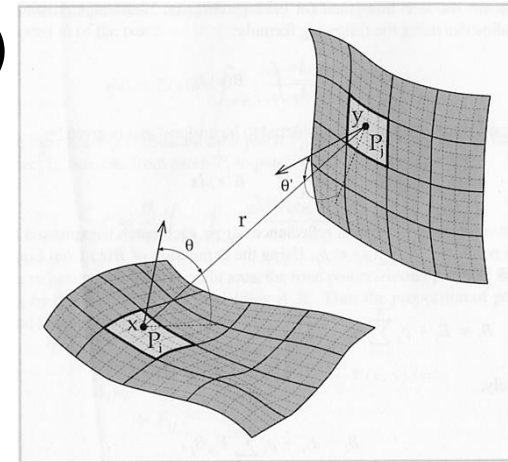
illuminazione, modelli globali

Radiosity

Procedimento

- Scena suddivisa in patches
- Determinazione dei fattori di forma per ogni coppia di superfici
- I fattori di forma definiscono quanta parte dell'energia che esce da una patch arriva su un'altra patch (tengono conto di occlusioni, orientamento reciproco e distanza)

equazione integrale $\rightarrow$ sistema lineare



- Risoluzione numerica: individuate le radiosità
il risultato è indipendente dalla posizione dell'osservatore

Quale distribuzione di superfici dá il maggiore trasferimento di energia luminosa e perché?



illuminazione, modelli globali

Equazione della radiosità discretizzata

(è una versione semplificata della equazione del rendering)

Per i-ma superficie

$$B_i = E_i + \rho_i \sum_{1 \leq j \leq n} B_j F_{ij}$$

n = numero di superfici

B_i = Radiosity della superficie i

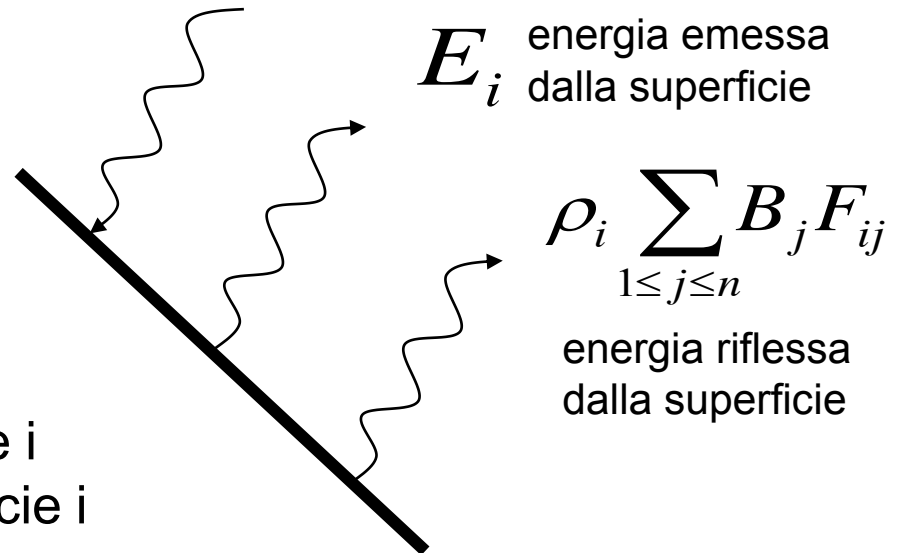
E_i = energia emessa dalla superficie i

ρ_i = coeff. di riflettenza della superficie i

B_j = radiosity della superficie j

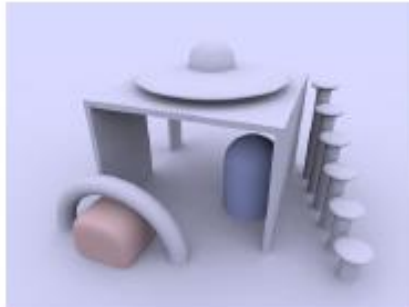
F_{ij} = fattore di forma della superficie j rispetto alla superficie i

$$\sum_{1 \leq j \leq n} B_j F_{ij} \quad \text{energia che raggiunge la superficie da altre superfici}$$



illuminazione, modelli globali

Radiosity



photograph



simulation

illuminazione, modelli globali

Radiosity

Confronto di scene basate su ray-tracing con e senza radiosity

Solo ray-tracing



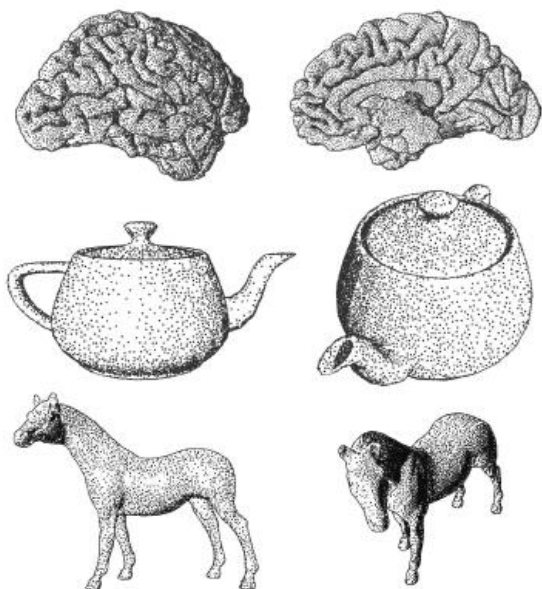
Ray-tracing & radiosity



si possono combinare
con algoritmi a due passi

Rendering non-fotorealistico

- Tecniche di rendering che non hanno come scopo il realismo ma piuttosto di realizzare delle scene che si ispirano a modelli artistici tipici della pittura o delle illustrazioni



puntinatura



Cel-shading

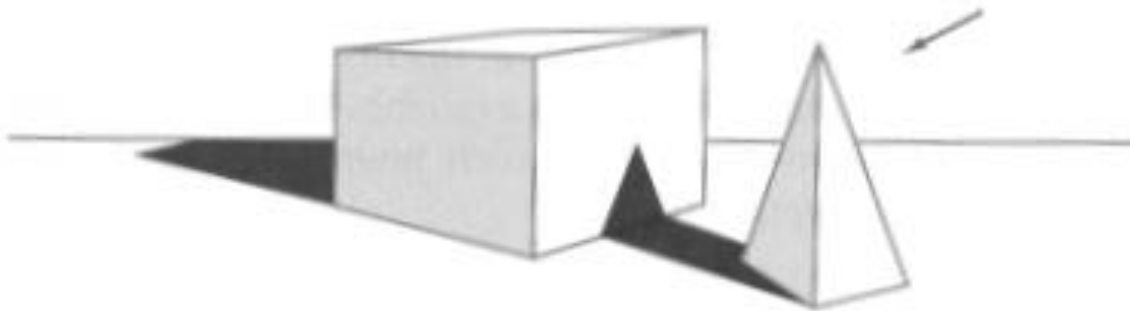
illuminazione, modelli locali



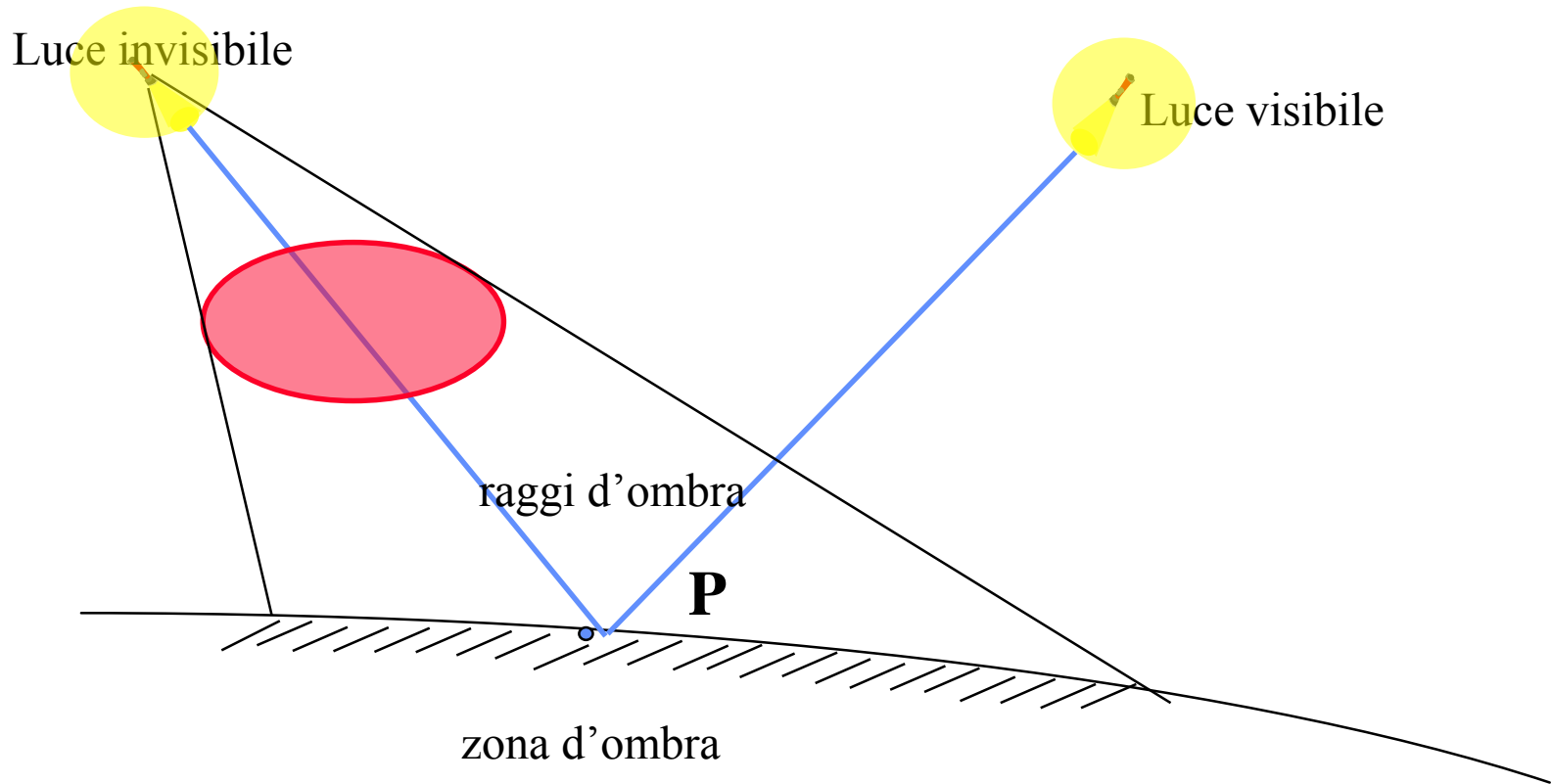
- Torniamo a parlare di modelli locali
- Vediamo una serie di algoritmi che attraverso degli artifici (“trucchi”) simulano gli effetti dell’interazione globale della luce visti precedentemente per aumentare il fotorealismo del risultato
 - OMBRE
 - TRASPARENZA
 - SHADING
 - RIFLESSIONI

Ombre

- **Parti dei poligoni nascoste alle fonti luminose ma non all'osservatore**



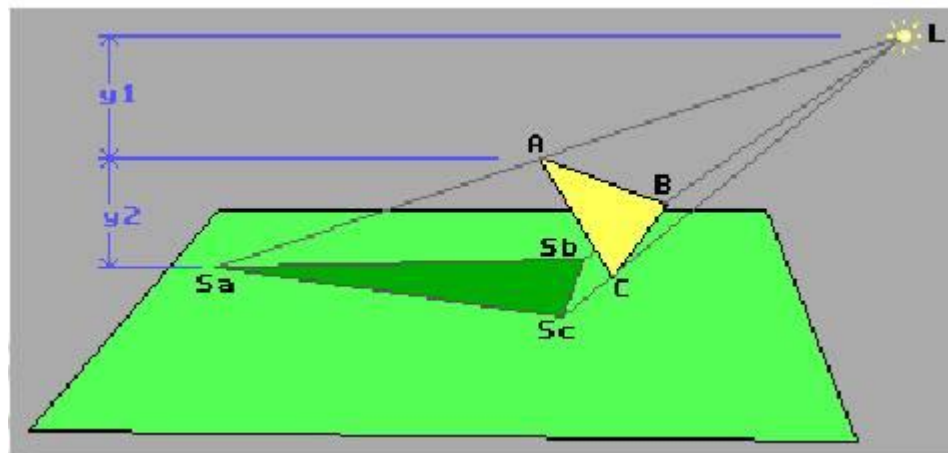
Ombre



se una luce è coperta da un oggetto,
non contribuirà al colore nel punto P considerato

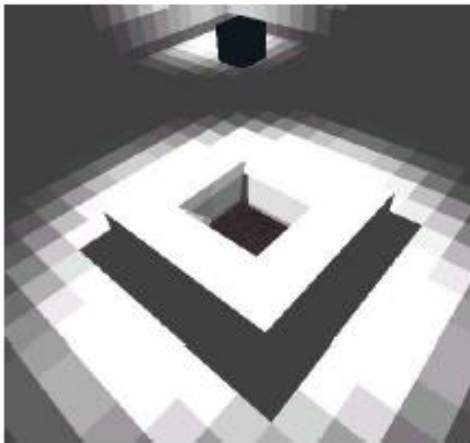
Ombre geometriche

- OpenGL non calcola automaticamente le ombre
- Esistono diverse tecniche geometriche per simulare la presenza di ombre
- Le ombre possono essere statiche o dinamiche

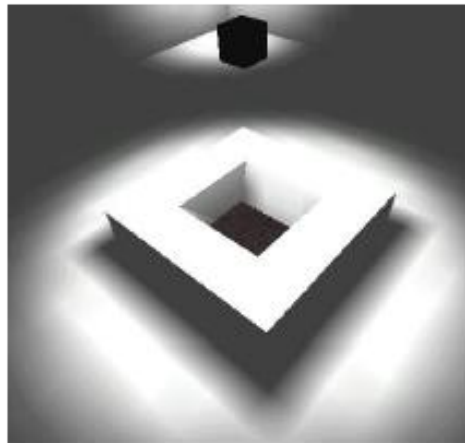


Ombre geometriche

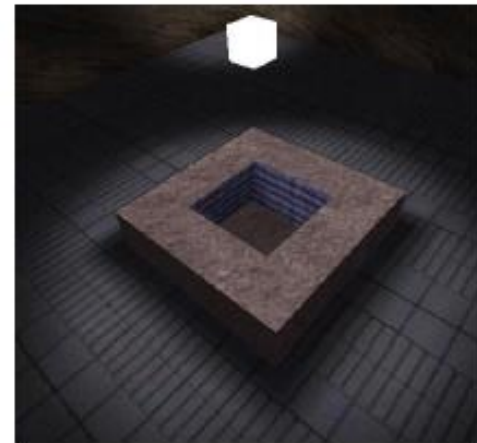
- Le ombre possono essere precalcolate (“fake”) per oggetti statici
- Si crea una light-map (mappa della luce+ombre) statica che si sovrappone alla scena e alle texture



(7) Lightmap + geomeri shadows



(8) Filtrata

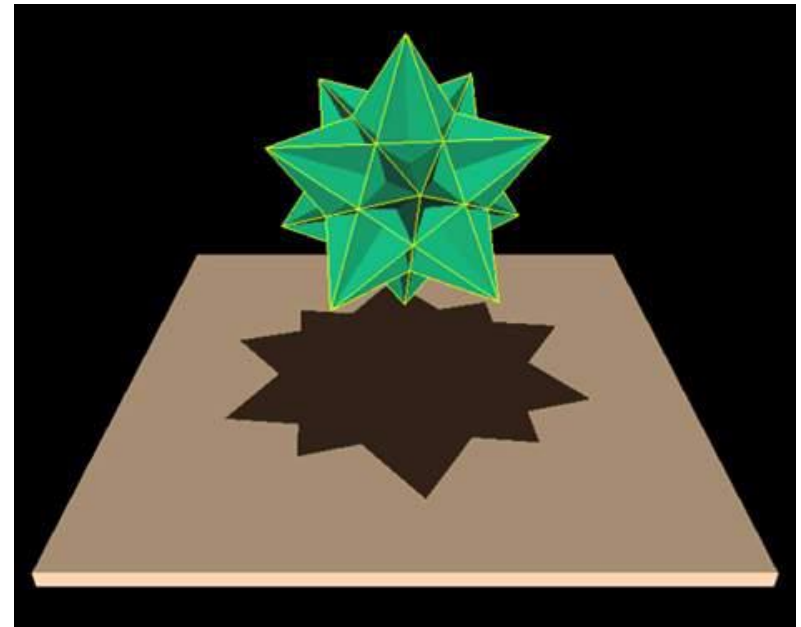


(9) Applicata all'oggetto con texture

Ombre geometriche

Ombre semplici calcolate sul piano (planar projected shadows)

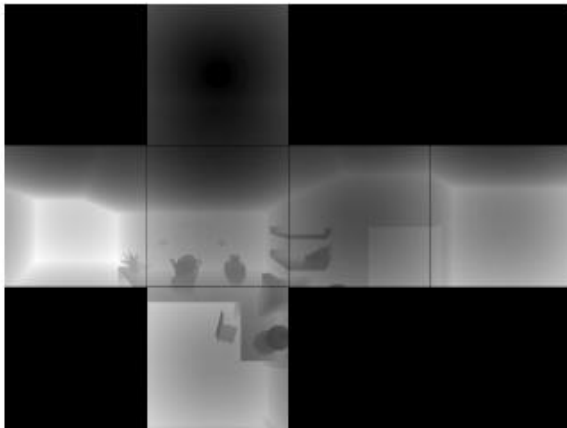
- Si disegna l'ombra come un oggetto piatto e nero sul piano
- Per disegnare l'ombra si effettua il rendering della scena con colore nero con una telecamera centrata nella sorgente di luce
- Si può utilizzare lo stencil buffer per disegnare solo le ombre sul piano



Ombre geometriche

Shadow Buffer

- Tecnica che si basa su un algoritmo parente con lo Z-buffer
- Si calcola lo Z-buffer (shadow buffer) dal punto di vista della luce
 - Se la luce è di tipo spot si calcola un solo z-buffer
 - Se la luce è una sorgente puntiforme la si considera racchiusa da un cubo e si calcolano sei z-buffer, uno per ogni faccia del cubo
- Nella 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 il valore z del punto nello z-buffer (nel sistema della telecamera) è più grande di z_l allora il punto è in ombra, in quanto esiste un oggetto che blocca la luce per quel punto.
- L'algoritmo funziona anche per più di una luce nell'ambiente.



(10) Shadow buffers

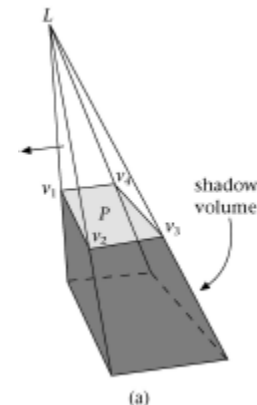
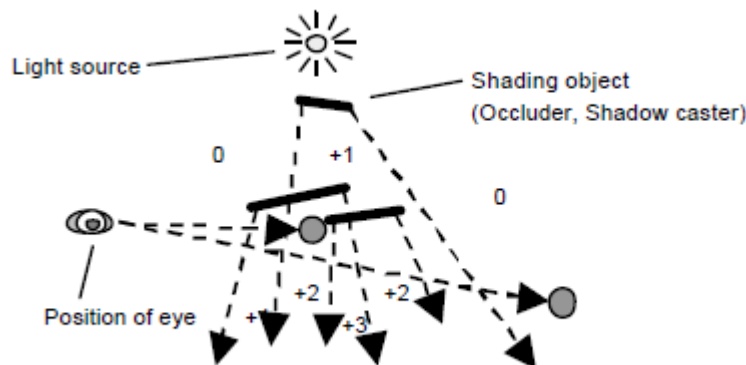


(11) Image

Ombre geometriche

Shadow Volume (Crow (1977))

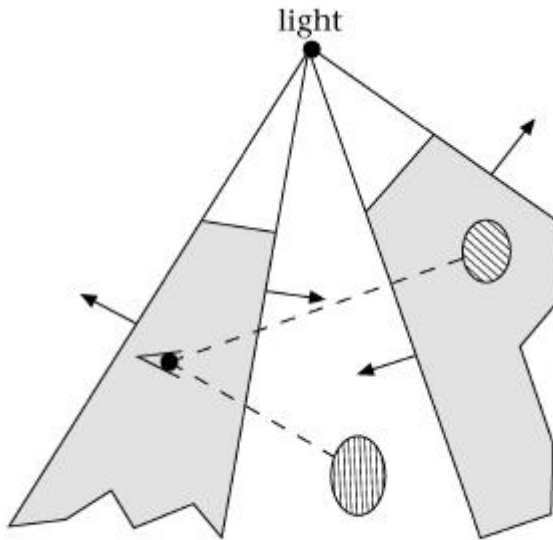
- Si costruisce un volume d'ombra di un poligono rispetto ad una sorgente di luce.
- Lo shadow volume è un tronco di piramide semi infinita, il cui vertice è la sorgente di luce ed è limitato dal poligono rispetto a cui è calcolato.
- Le facce della piramide prendono il nome di *shadow-planes*
- Per un **punto di vista non in ombra**, dato un punto P della scena, si traccia il segmento che congiunge il punto di vista e P e si contano gli attraversamenti degli shadow planes, incrementando un contatore quando l'attraversamento è in entrata e decrementandolo quando è in uscita. Se la differenza è zero il punto è nella luce, altrimenti è nell'ombra (di uno o più poligoni).
- Si ottiene utilizzando lo stencil-buffer



Ombre geometriche

Shadow Volume (Crow (1977))

- Se il punto di vista è nell'ombra si parte con un contatore uguale al numero di shadow volumes nei quali il punto di vista è contenuto.



© Slater et al.

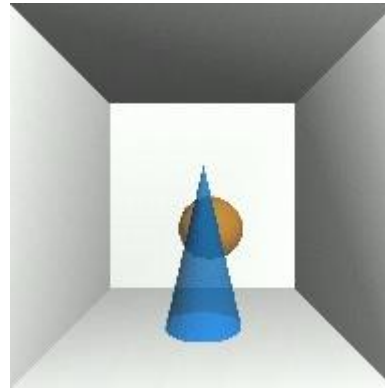
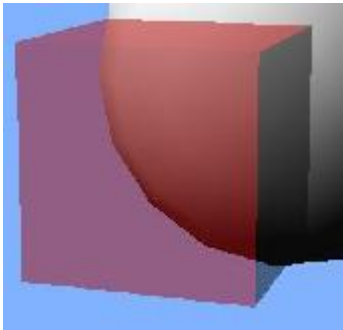
- Osservazioni:
 - più accurato rispetto a shadow buffer
 - ma richiede uso di buffer aggiuntivi (stencil)



Carmack's stencil shadowing in Doom 3

Trasparenza

- La componente α della rappresentazione RGB_α può essere usata per specificare un valore di opacità utile per simulare un effetto di trasparenza in OpenGL, detto alpha blending (tecnica molto più semplificata rispetto al ray tracing)
- Lo vediamo in laboratorio nella parte 5



illuminazione, modelli locali



- **LIGHTING**

- processo di calcolo dell'equazione di illuminazione in un generico punto/vertice dell'oggetto da illuminare.

- **SHADING**

- definizione del colore di ogni singolo pixel appartenente ad un poligono. Definisce la modalità con cui l'illuminazione (lighting) viene calcolata internamente ai poligoni.

Tecniche di Shading

- **modelli di shading:**

- Constant shading

- Faceted shading

- Shading interpolato

- Gouraud shading

- Phong shading (il più realistico)

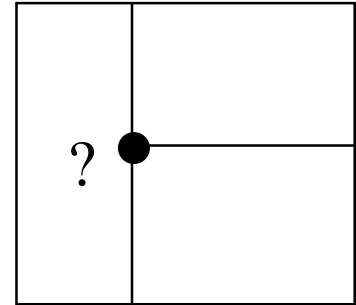
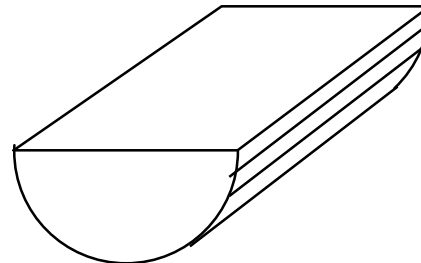
Tecniche di Shading

Vantaggi

Computazionalmente più efficienti rispetto all'applicazione point-based di metodi di illuminazione globale

Limiti

- Tecniche applicate a modelli poliedrali
- Effetto di spigoli vivi (tranne metodo di Phong)
- Algoritmi comunque costosi se per ambienti con elevato numero di poligoni



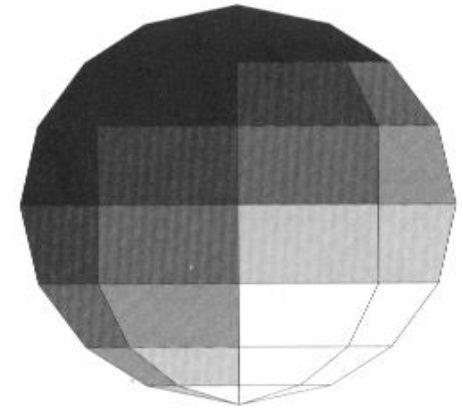
Constant e Faceted Shading

- **Constant shading:**

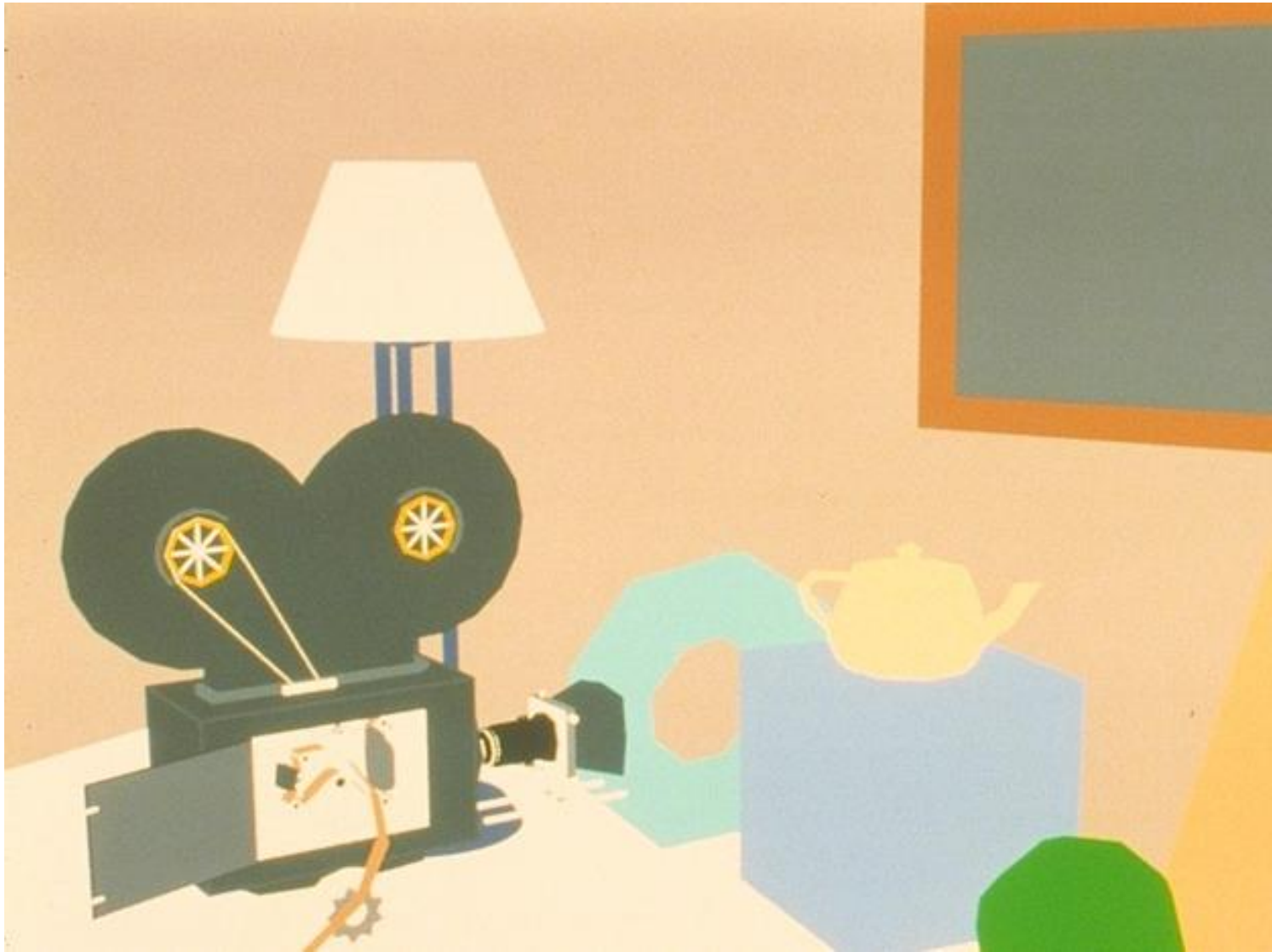
- stesso valore di intensità per ogni punto di un oggetto costituito da tanti poligoni
- vantaggi: calcolo veloce svantaggi: poco realismo

- **Faceted o flat shading**

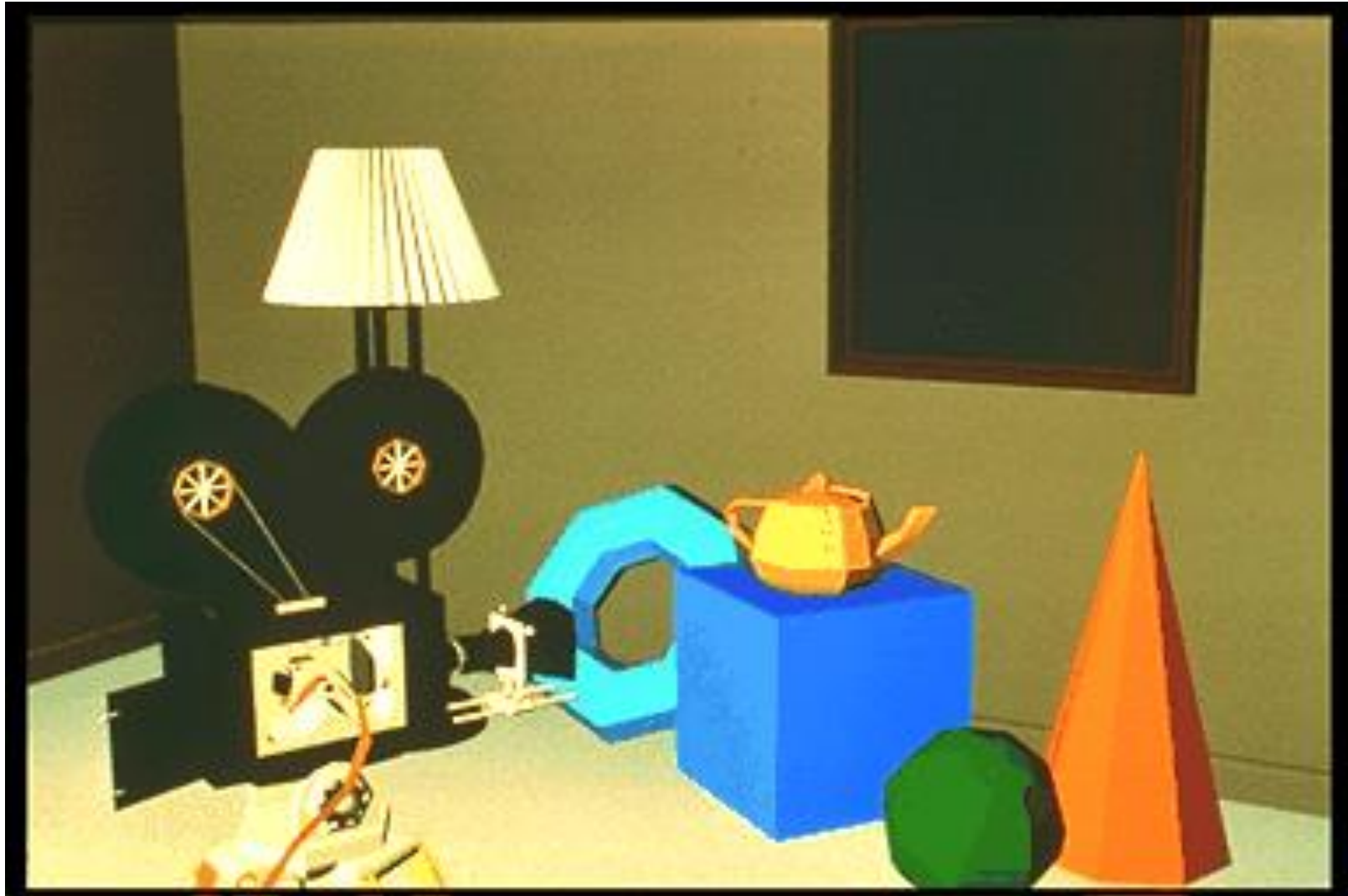
- stesso valore di intensità per ogni punto di ogni poligono
- ... ma da poligono a poligono, cambia!
- Per ogni poligono:
 intensità calcolata in un punto di riferimento
 (es: un vertice estremo oppure il baricentro)
- vantaggi: ancora veloce svantaggi: ancora scarso realismo



Constant Shading: esempio



Faceted Shading: esempio



LIMITE: Effetto di spigolosità

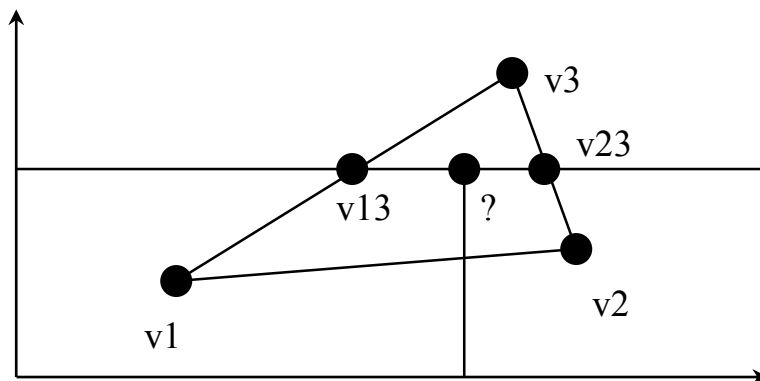
Shading interpolato

- **Caratteristica:**

- illuminazione locale calcolata sui vertici estremi dei poligoni
- A tutti i vertici estremi di un poligono si associa lo stesso vettore normale
- il colore sui vertici interni del poligono viene calcolato per interpolazione lineare (NOTA: supportato in OpenGL, `glShadeModel(GL_SMOOTH)`)
- E' la tecnica che utilizzeremo nei nostri esercizi a parte alcuni casi di oggetti definiti da superfici analitiche in cui saremo in grado di calcolare agevolmente i vettori normali di ciascun vertice estremo di un poligono (es: cilindro)

- **Problema:**

- Se lo shading interpolato viene fatto **indipendentemente** su ogni poligono, vi è discontinuità visiva ai bordi di poligoni adiacenti



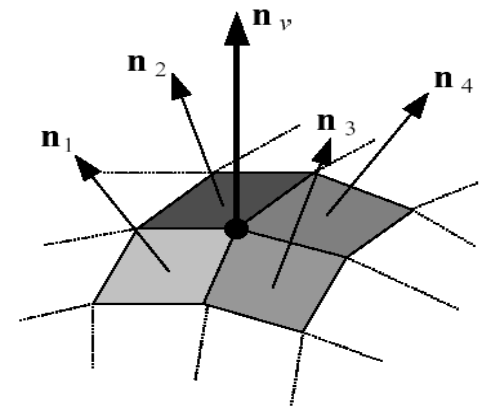
Come risolvere
questo effetto visivo
discontinuo?

Gouraud Shading

- *Intensity interpolation shading*
- **Evoluzione/miglioramento rispetto allo shading interpolato di base**
 - eliminazione discontinuità sui bordi delle faccette
- **il calcolo dell'illuminazione locale avviene ancora ai vertici estremi dei poligoni con interpolazione del colore all'interno dei poligoni**

- **Ma i vettori normali associati ad ogni vertice estremo di un poligono (utilizzati per il calcolo dell'illuminazione locale) sono calcolati come media dei vettori normali dei poligoni adiacenti a ciascun vertice**

Esempio

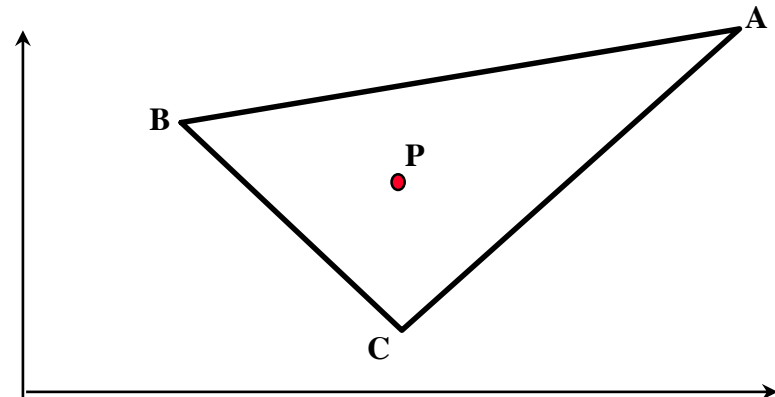
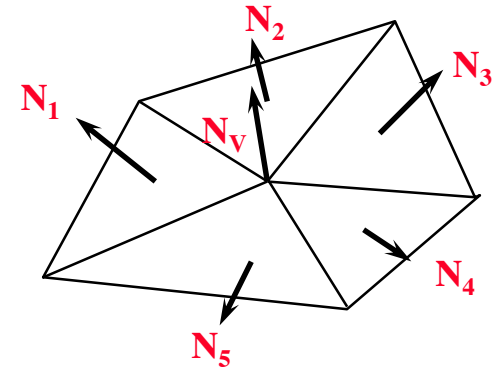


$$\mathbf{n}_v = \frac{\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4}{|\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4|}$$

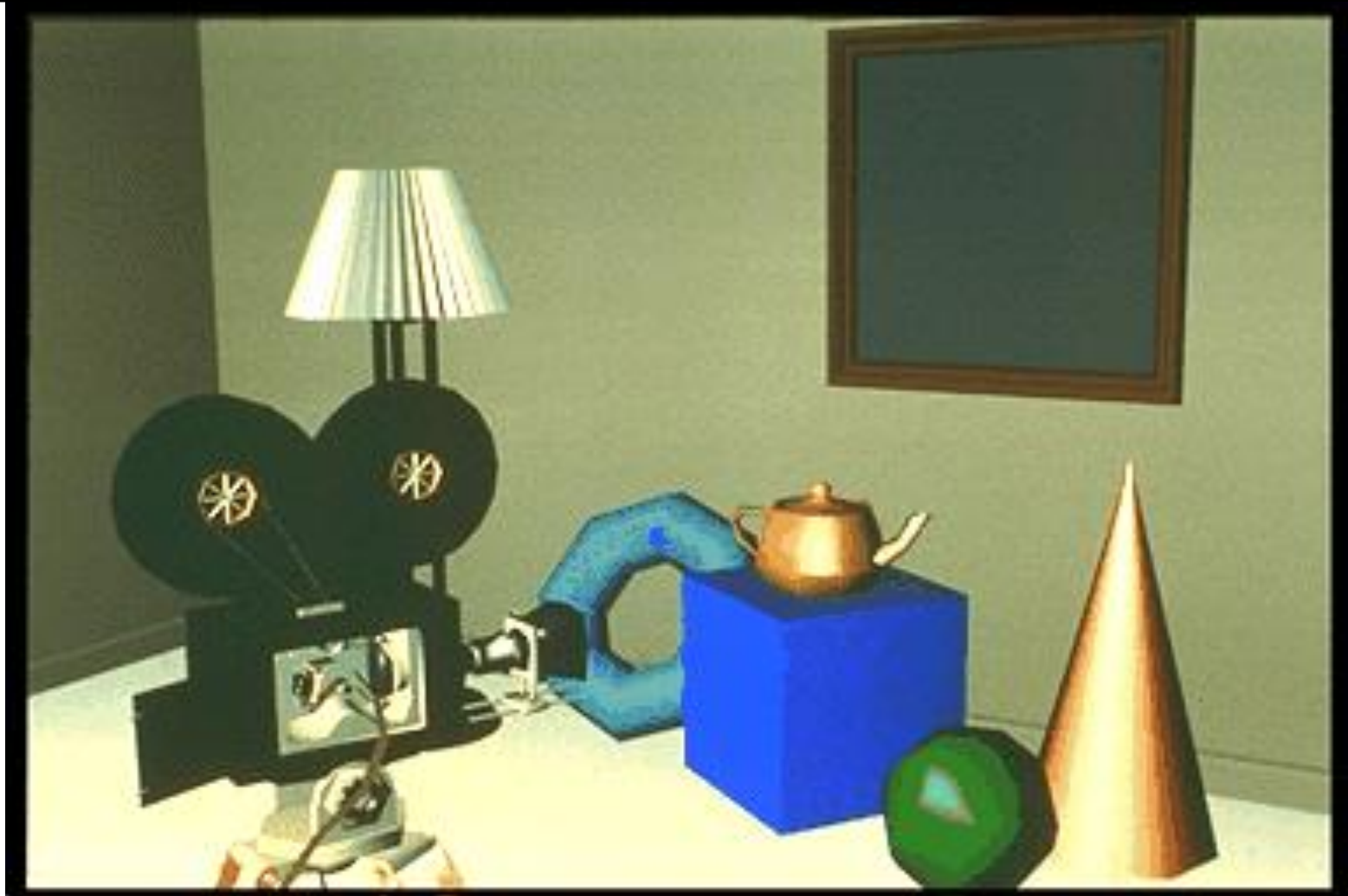
Gouraud Shading

Algoritmo

- Per ogni poligono:
calcolare il vettore normale
- Per ogni vertice:
determinare il vettore medio unitario
delle normali alle facce adiacenti
a quel vertice:
$$N_V = \frac{\sum_{k=1}^n N_k}{\left| \sum_{k=1}^n N_k \right|}$$
- Applicare modello di illuminazione
a ciascun vertice per calcolare
l'intensità I nel vertice
- interpolare linearmente le intensità
dei vertici sulla superficie del poligono



Gouraud Shading: esempio



LIMITE: Gouraud shading spesso sufficiente, ma non così realistico se si rappresentano oggetti ad alto coefficiente di riflessione speculare

interpolazione colore

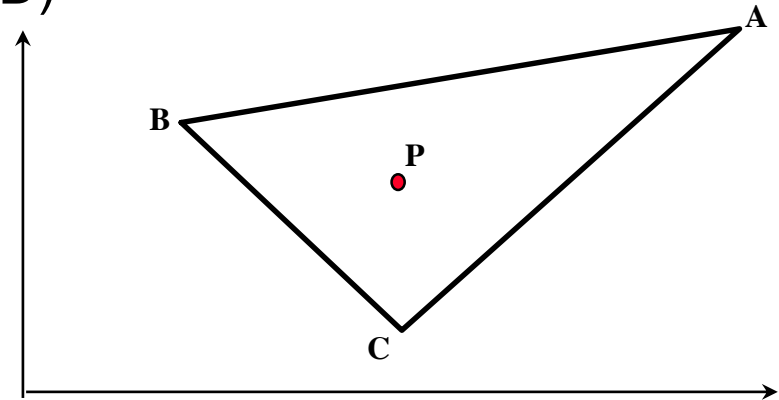
Si utilizza la notazione in **coordinate baricentriche** (α, β, γ) di un punto $P(x, y)$ rispetto ai vertici di un triangolo su un piano 2D (IL triangolo ABC in figura è la proiezione sul piano image di un triangolo 3D)

$$P = \alpha A + \beta B + \gamma C$$



$$\alpha + \beta + \gamma = 1$$

Se $0 \leq \alpha \leq 1$ $0 \leq \beta \leq 1$ $0 \leq \gamma \leq 1$ il punto P giace all'interno del triangolo



Algoritmo di shading/rasterizzazione di un singolo triangolo

(c_0, c_1, c_2 sono i colori dei pixel A, B, C calcolati dal modello di illuminazione locale sul triangolo 3D)

```
for all x do
  for all y do
    compute (alpha, beta, gamma) for (x,y)
    if ( 0 < alpha < 1 and
        0 < beta < 1 and
        0 < gamma < 1 ) then
      c = alpha c0 + beta c1 + gamma c2
      drawpixel(x,y) with color c
```

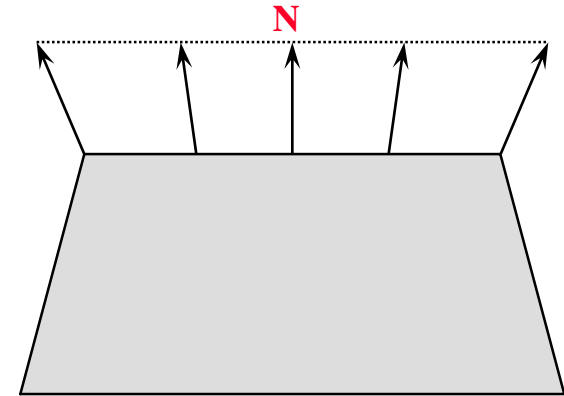
(x, y) sono
coordinate pixel

Z-Buffer & rasterizzazione/shading

- Z-buffer lavora assieme ad algoritmi di rasterizzazione/shading nel modello di pipeline tradizionale
- Rivediamo l'algoritmo di rasterizzazione/shading :
 1. si inizializzano il color buffer e il depth buffer (chiamato anche zbuffer)
 2. per ogni poligono (in ordine casuale) si determinano i pixel corrispondenti alla sua proiezione sul piano di proiezione, utilizzando l'algoritmo di rasterizzazione basato sulle coordinate baricentriche
 3. Per ogni pixel (x,y) del poligono in esame si calcola la z
se $z >$ del valore attuale nel depth buffer
 1. si aggiorna il valore nello zbuffer: **zbuffer(x,y)=z**
 2. si aggiorna il valore dell'intensità nel color buffer
colorbuffer(x,y) = $\alpha * c_0 + \beta * c_1 + \gamma * c_2$altrimenti non viene fatta nessuna operazione

Phong Shading

- **La migliore delle tecniche di shading classiche...**
- **... ma più costosa** (non presente utilizzando la pipeline fissa di OpenGL, si può ottenere con pipeline programmabile nel linguaggio di shading GLSL)
- ***Normal vector interpolating shading***
 - si interpolano i vettori normali all'interno dei poligoni anzichè interpolare i colori
 - si applica il modello di illuminazione locale su ciascun vertice interno ai poligoni
- **Vantaggi rispetto a Gouraud**
 - maggiore realismo

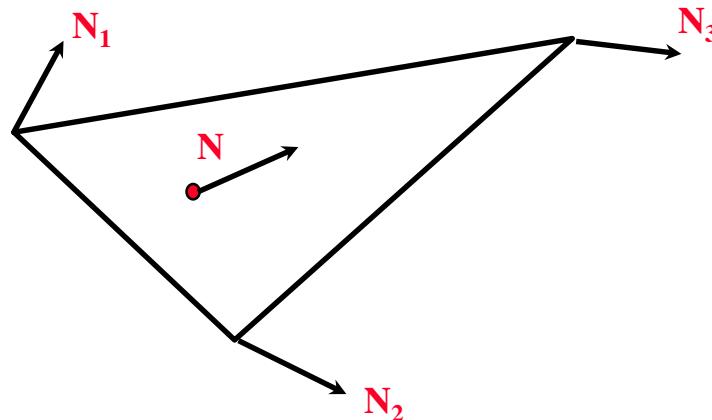


!!! Non confondere Phong-shading con il modello di illuminazione di Phong

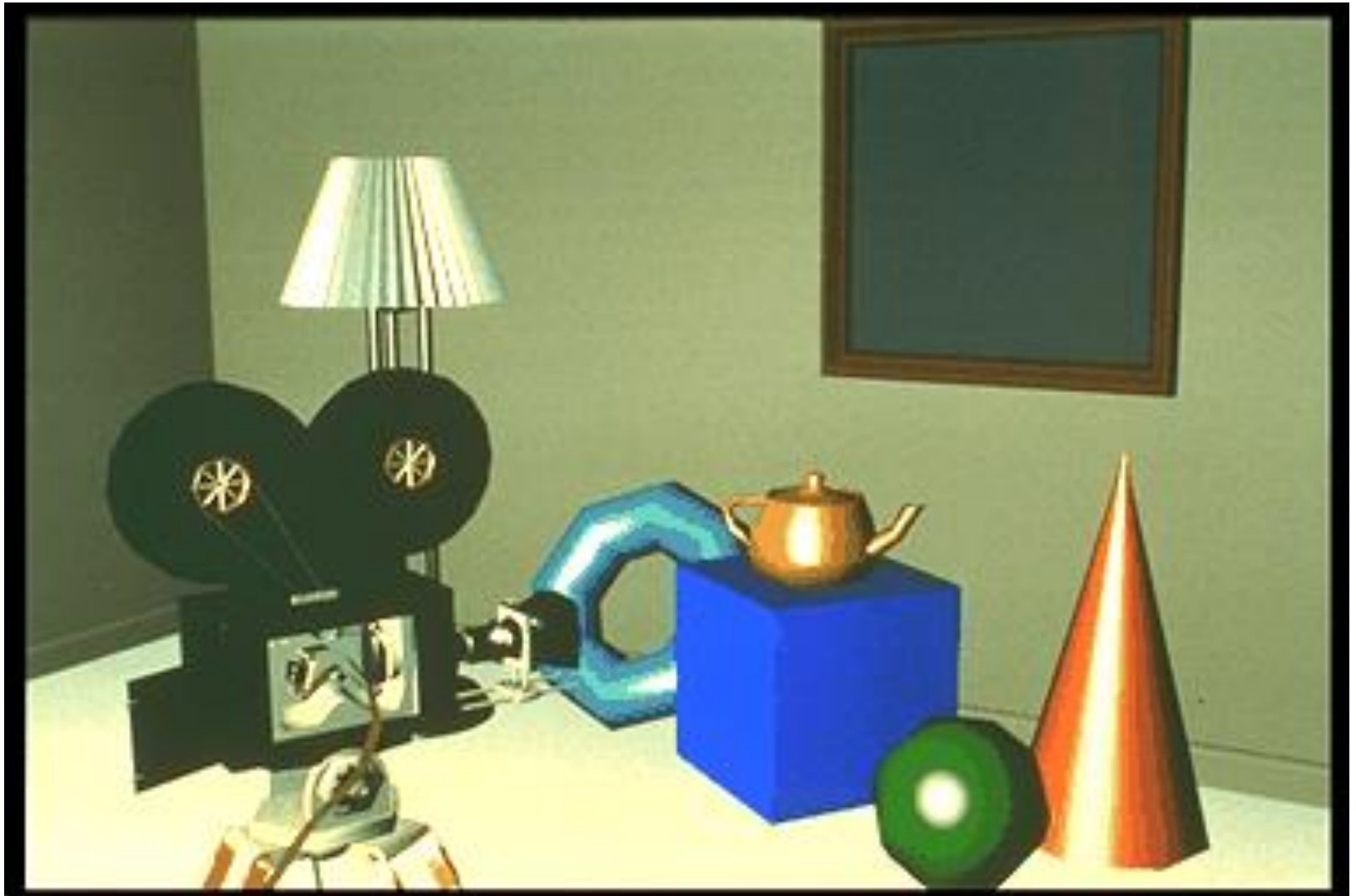
Phong Shading

Algoritmo

- Per ogni faccia: calcolare la normale
- Per ogni vertice: determinare il vettore medio unitario delle normali alle facce adiacenti a quel vertice (oppure calcolo diretto della normale se è dato una rappresentazione analitica della superficie)
- Interpolare linearmente le normali ai vertici sulla superficie del poligono
- Applicare un modello di illuminazione locale all'interno del poligono sfruttando le normali interpolate



Phong Shading: esempio



Shading: altri confronti

Faceted



Gouraud



Phong

Shading: altri confronti



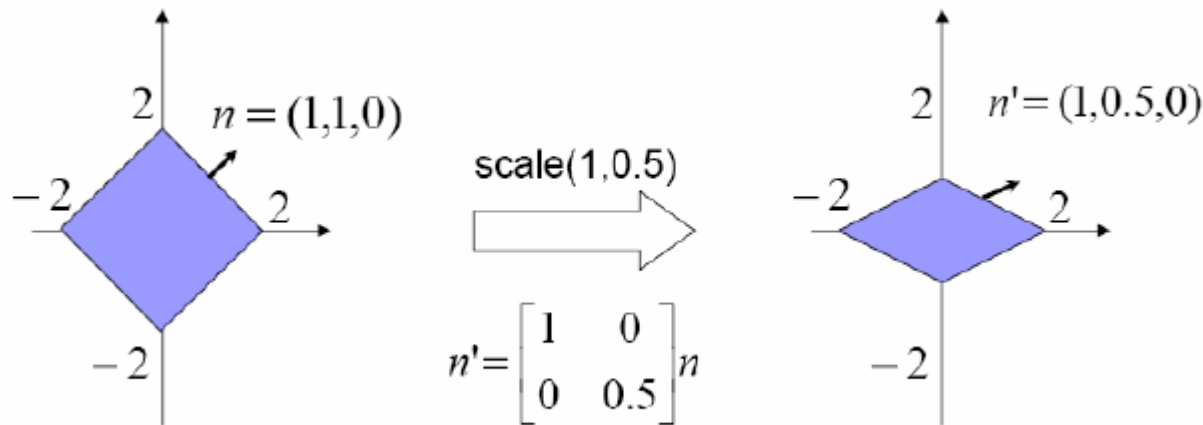
Faceted

Gouraud

Phong

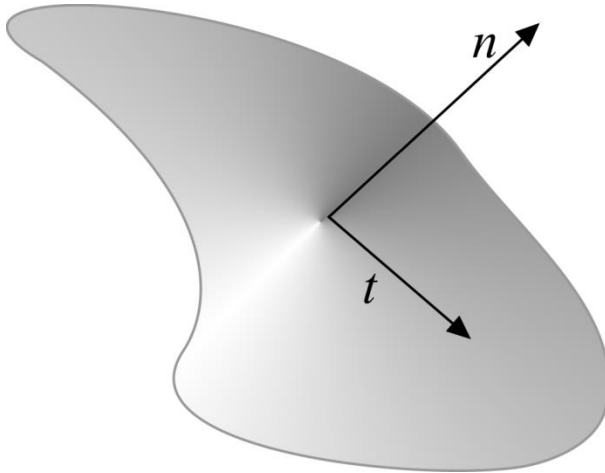
illuminazione, trasformazione delle normali

- Abbiamo visto che i vettori normali ai poligoni sono essenziali per il calcolo dell'illuminazione
- in generale, in seguito ad una trasformazione M il vettore normale ad una superficie/poligono n quando viene trasformato in n' tramite M può risultare non più normale alla superficie trasformata.



- Pertanto l'engine grafico deve applicare al vettore normale una trasformazione K per garantire che il vettore risultante sia ancora normale alla superficie trasformata . Questa trasformazione viene eseguita automaticamente nella pipeline (trasparente al programmatore). Vediamo come è fatta K

illuminazione, trasformazione delle normali



$$t = p - q$$

$$n' = Kn$$

$$t' = Mt$$

sappiamo che $n^T t = 0$

vogliamo avere che $n'^T t' = 0$

$$n'^T t' = (Kn)^T (Mt) = n^T K^T Mt$$

se poniamo $K^T M = I \Rightarrow K^T = M^{-1} \Rightarrow K = (M^{-1})^T$

avremo $n'^T t' = n^T K^T Mt = 0$

quindi i vettori normali vanno trasformati per la trasposta dell'inversa della matrice di modellazione/vista (ModelView)



Texture Mapping

Finitura visiva

Finitura “visiva” delle superfici degli oggetti:

– Texture mapping

- metodo per aggiungere dettaglio ad una scena 3D mediante l'applicazione di una texture (texture mapping 3D, texture mapping 2D)
- la superficie resta liscia



– Bump mapping

- perturbazione dei vettori normali alla superficie
- la superficie mostra l'effetto di rugosità/materiale
- la superficie resta liscia

– Environment mapping

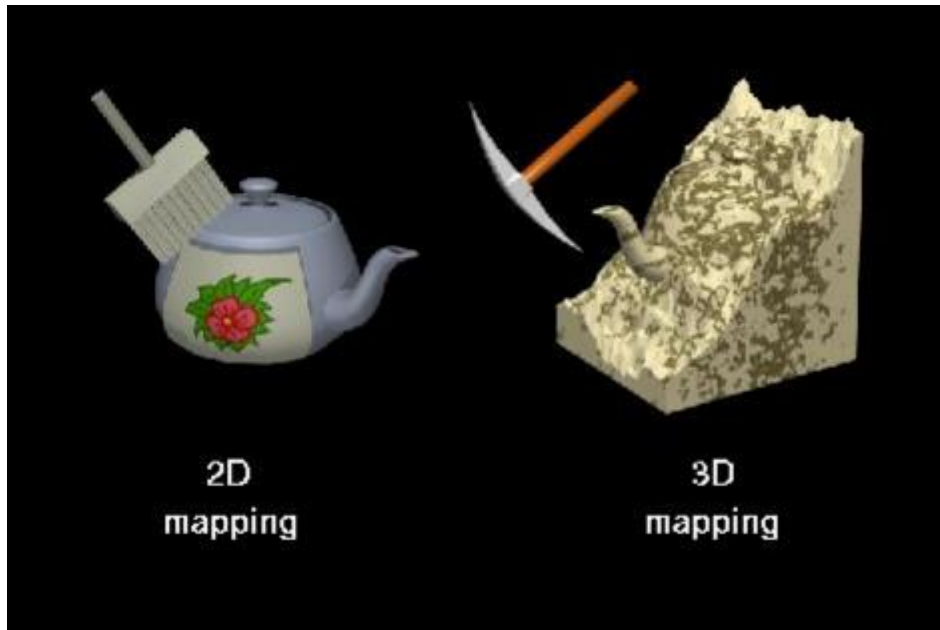
Texture Mapping

- **Texture mapping 2D**

- “Incollare” una immagine 2D sulla superficie di un oggetto

- **Texture mapping 3D**

- Costruire una texture 3D e “intagliare” l’oggetto (*carving*)

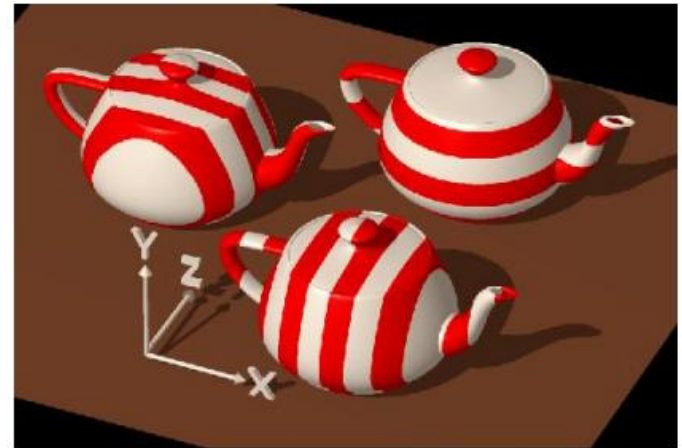


Texture Mapping 3D

- Approccio **procedurale**, si crea una funzione matematica per creare una texture 3D: $f(x, y, z) = c$ with $c \in \mathbb{R}^3$

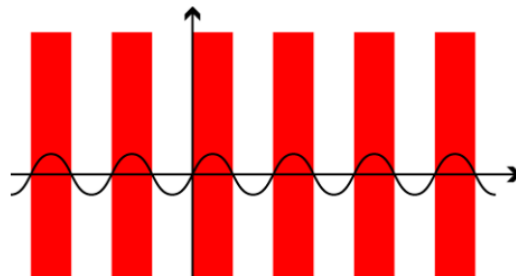
- c è un colore
- ad ogni punto (x_p, y_p, z_p) che appartiene alla superficie di un oggetto si applica la funzione di texture

$$f(x_p, y_p, z_p) = c_p$$



- Esempio di texture 3D a strisce lungo asse x

```
stripe(  $x_p, y_p, z_p$  )  
{  
  if (  $\sin x_p > 0$  )  
    return color0;  
  else  
    return color1;  
}
```



Texture Mapping 2D

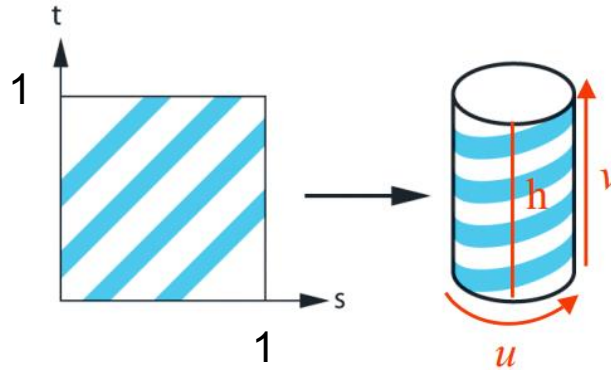
- **Visualizzazione di decorazioni, tinteggi e stampe presenti sulle superfici degli oggetti per un effetto di maggiore realismo**
- **Idea**
 - E' come “incollare” un foglio di carta con un'immagine sulla superficie esterna di un oggetto, flettendo/piegando il foglio per seguire la forma dell'oggetto, ma non trazionandolo
- **Input**
 - Geometria 3D dell'oggetto
 - Immagine 2D
- **Output**
 - Geometria 3D dell'oggetto “texturizzata”
- Una volta incollata la texture è soggetta alle stesse trasformazioni dell'oggetto nella pipeline di output visivo
- **Parleremo di texture mapping 2D nei seguenti casi:**
 - 1) **Two-stage mapping** (tecnica automatica per il texture mapping di oggetti 3D)
 - 2) **Texture mapping di poligoni**

Texture Mapping 2D

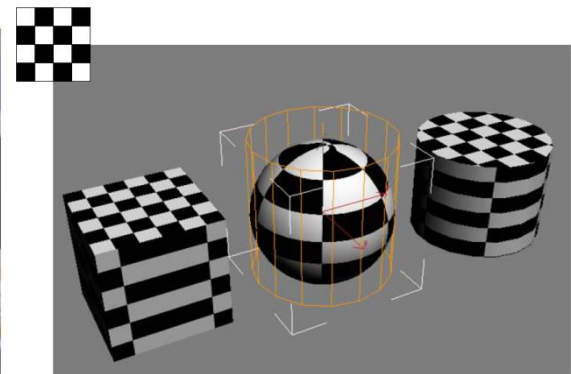
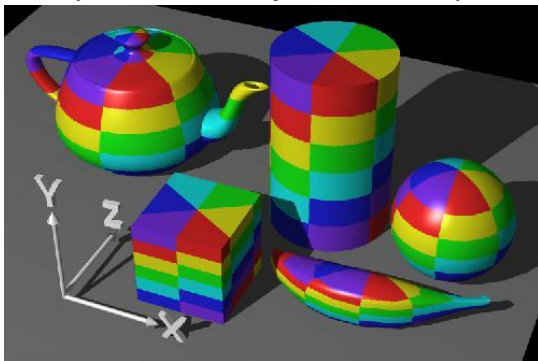
Two-stage mapping - Texture mapping cilindrico

Fase 1) si esegue il mapping della texture su una superficie intermedia CILINDRICA che si esprime in coordinate cilindriche

$$\begin{cases} u = 2\pi s \\ v = ht \end{cases}$$



Fase 2) si esegue il mapping della superficie intermedia sulla superficie dell'oggetto
(vedi slide più avanti)

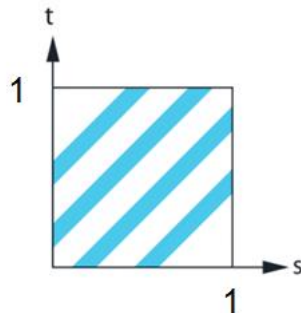


Texture Mapping 2D

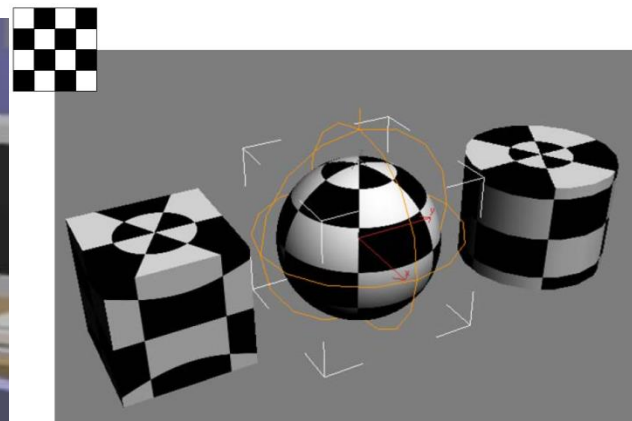
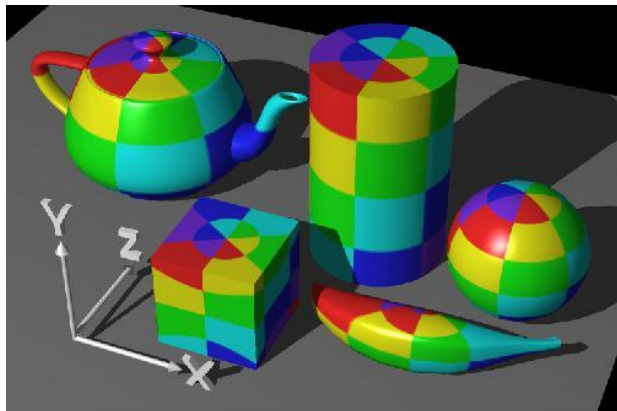
Two-stage mapping - Texture mapping sferico

Fase 1) si esegue il mapping della texture su una superficie intermedia SFERICA che si esprime in coordinate sferiche

$$\begin{cases} u = 2\pi s \\ v = \pi t \end{cases}$$



Fase 2) si esegue il mapping della superficie intermedia sulla superficie dell'oggetto
(vedi slide più avanti)



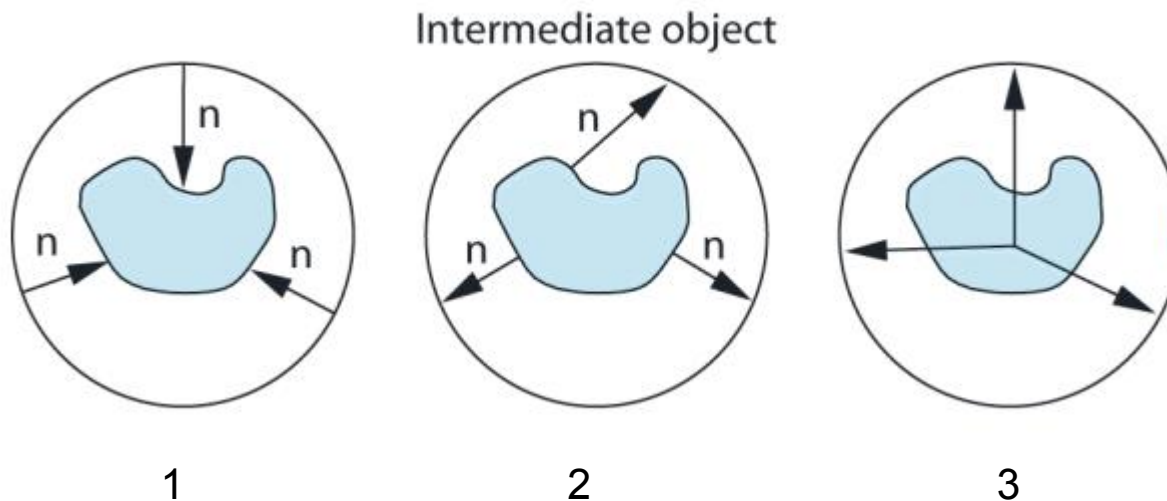
Texture Mapping 2D

Two-stage mapping - Texture mapping sferico

Fase 2) si esegue il mapping della superficie intermedia sulla superficie dell'oggetto

Ci sono varie strategie:

1. intersezione delle normali dalla superficie intermedia all'oggetto
2. intersezione delle normali dall'oggetto alla superficie intermedia
3. intersezione del vettore dal centro dell'oggetto alla superficie intermedia



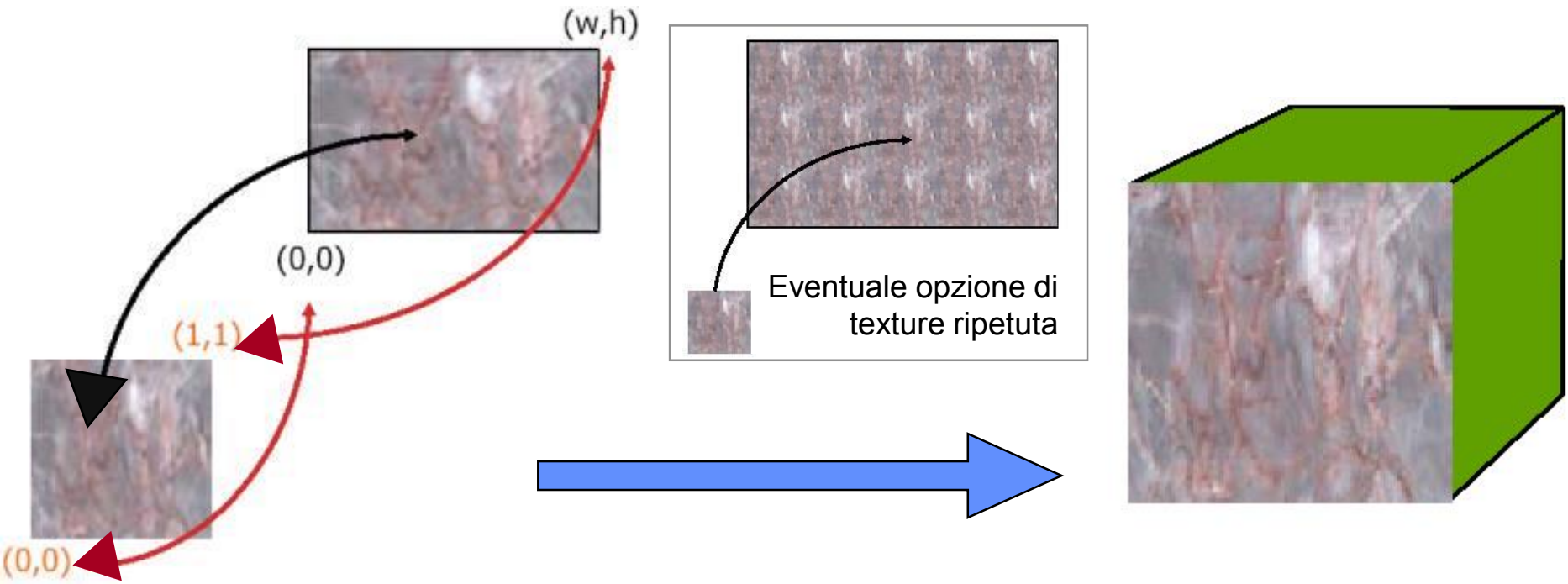
Texture Mapping 2D

Texture mapping di poligoni

- Nello spazio di coordinate 2D delle textures si definisce una trasformazione che mappa la texture (nxm pixels) nel quadrato di lato unitario $[0,1] \times [0,1]$
- Si definisce poi la trasformazione che mappa il poligono sulle coordinate normalizzate $[0,1] \times [0,1]$ (interamente o parzialmente)

Sequenza delle trasformazioni:

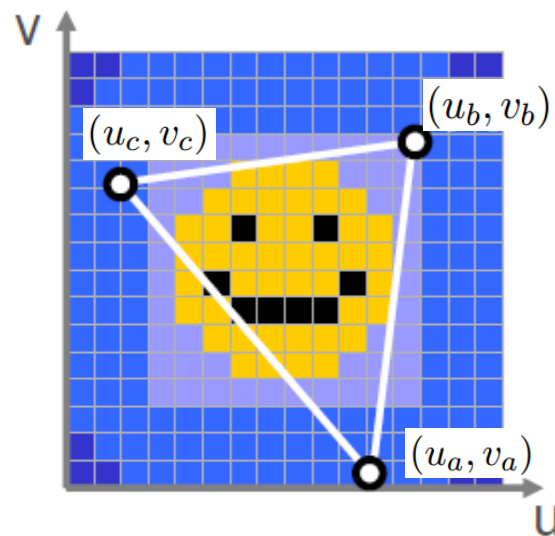
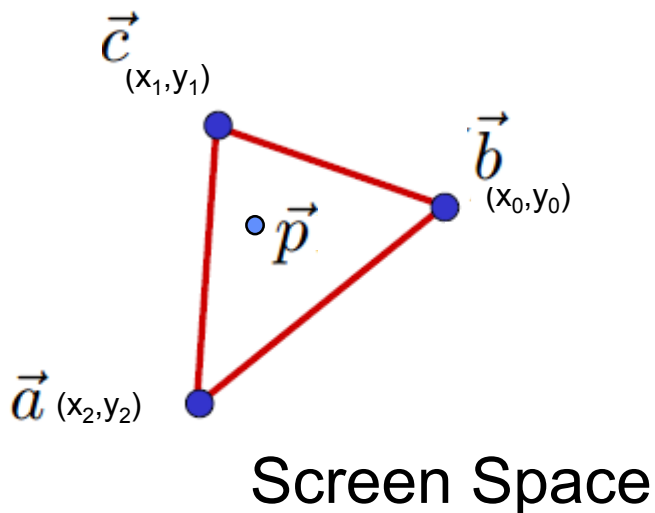
$$I_n \times I_m \longrightarrow [0,w] \times [0,h] \longrightarrow [0,1] \times [0,1] \longrightarrow S(u,v) \text{ poligono in } R^3$$



Texture Mapping 2D

Texture mapping di poligoni

- Ad ogni vertice 3D di un triangolo (a,b,c) su cui si deve applicare una texture assegno le sue corrispondenti coordinate u, v nello spazio tessitura



dato un punto del triangolo (piano immagine) in coordinate baricentriche

$$\vec{p}(\beta, \gamma) = \vec{a} + \beta(\vec{b} - \vec{a}) + \gamma(\vec{c} - \vec{a})$$

si calcolano le coordinate texture come:

$$u(\beta, \gamma) = u_a + \beta(u_b - u_a) + \gamma(u_c - u_a)$$

$$v(\beta, \gamma) = v_a + \beta(v_b - v_a) + \gamma(v_c - v_a)$$

Texture Space

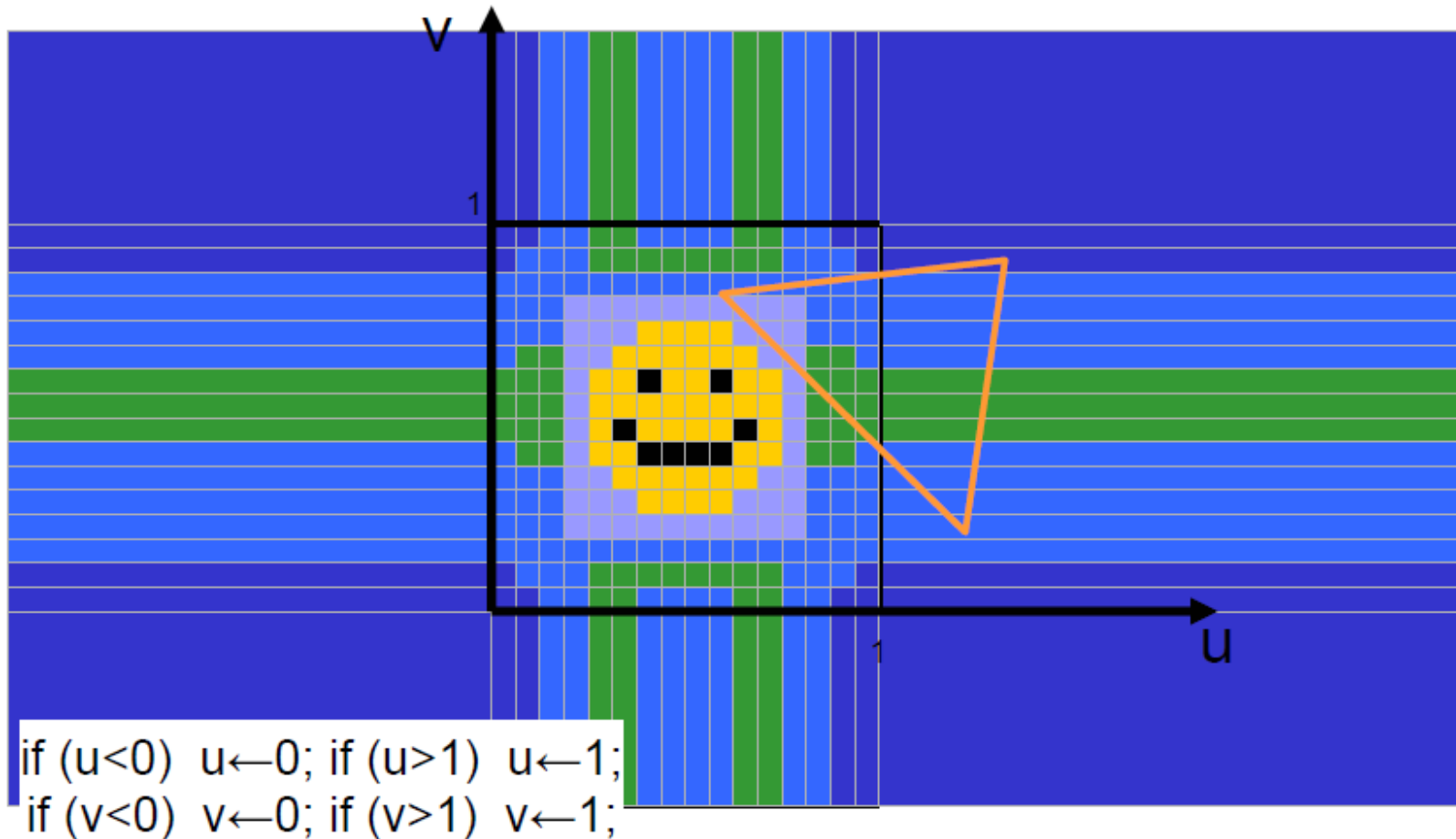
texel = un pixel della texture

Texture Mapping 2D

Texture mapping di poligoni

Occorre definire il valore delle texture anche fuori dal lato unitario $[0,1] \times [0,1]$

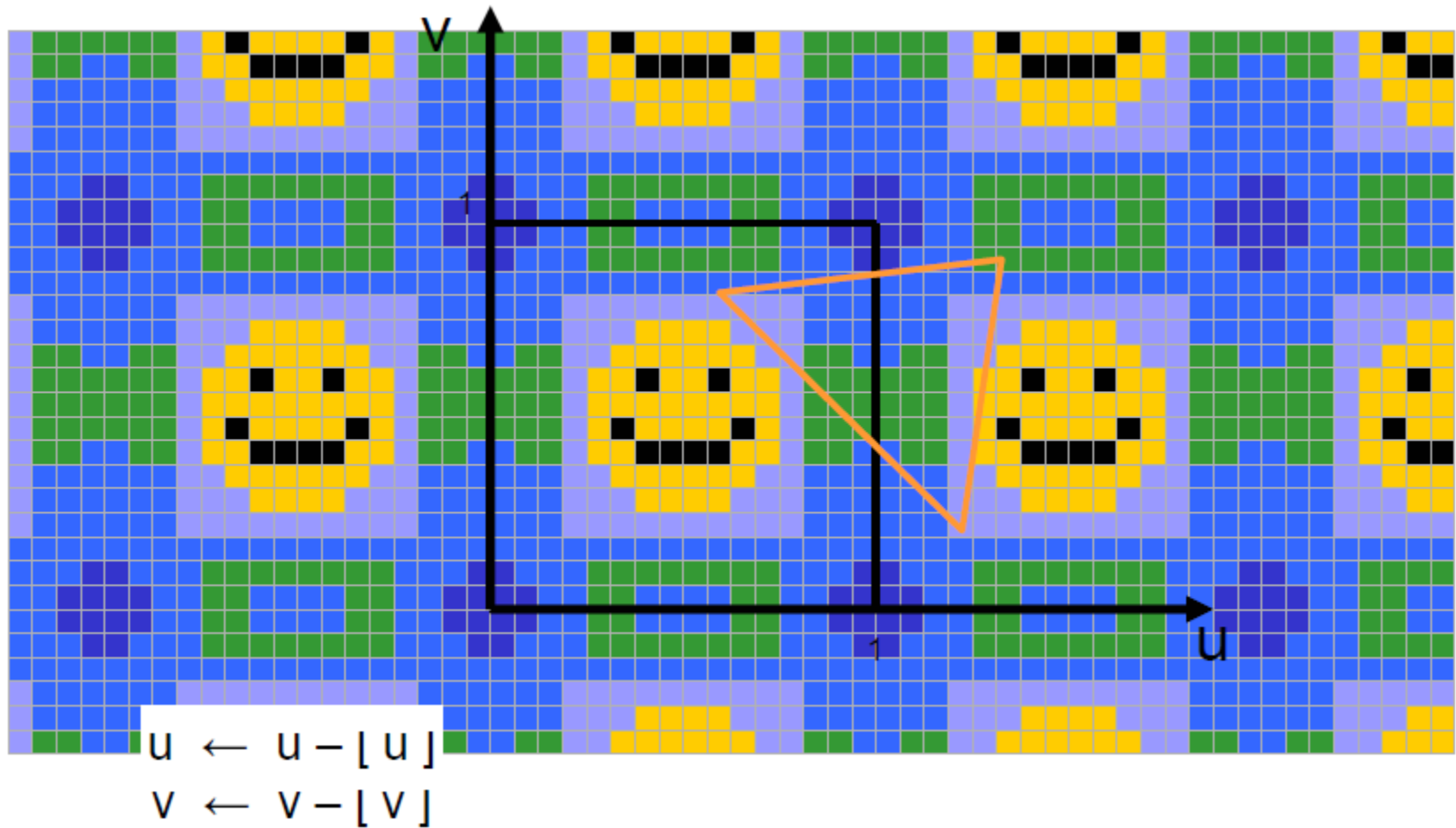
Procedimento fuori dai bordi: modalità clamp



Texture Mapping 2D

Texture mapping di poligoni

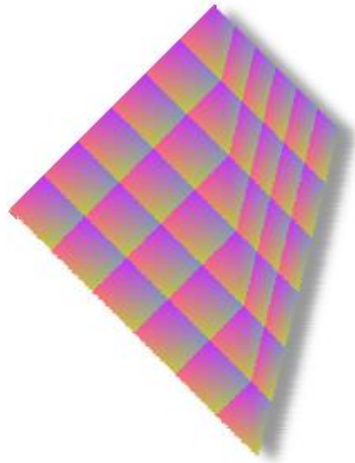
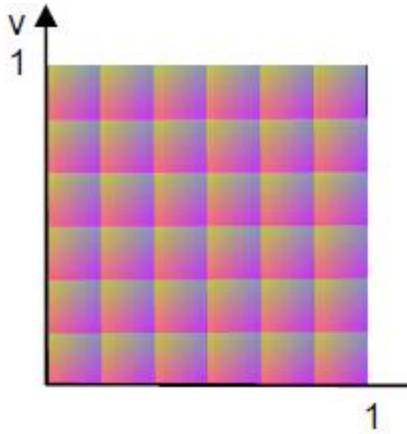
Procedimento fuori dai bordi: modalità repeat



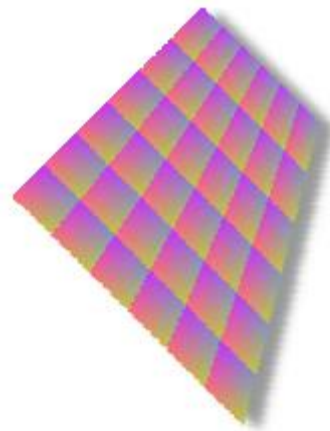
Texture Mapping 2D

Texture mapping di poligoni

correzione prospettica durante interpolazione



senza correzione



con correzione

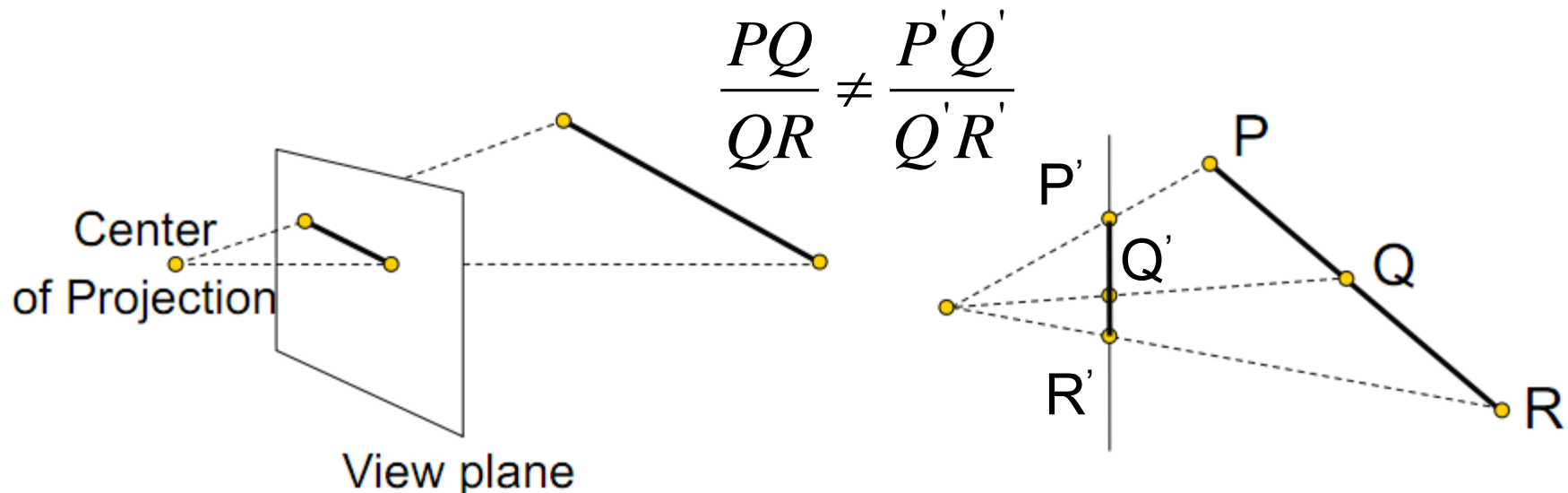
Texture Mapping 2D

Texture mapping di poligoni

correzione prospettica durante interpolazione

Problema: la proiezione prospettica non mantiene gli stessi rapporti di lunghezza tra segmenti. In particolare, i rapporti tra le lunghezze di segmenti in “eye-coordinates” non corrispondono ai rapporti tra le lunghezze dei corrispondenti segmenti proiettati sul piano immagine. Di conseguenza una “semplice” interpolazione lineare dei texels non è corretta quando siamo in un ambiente 3D.

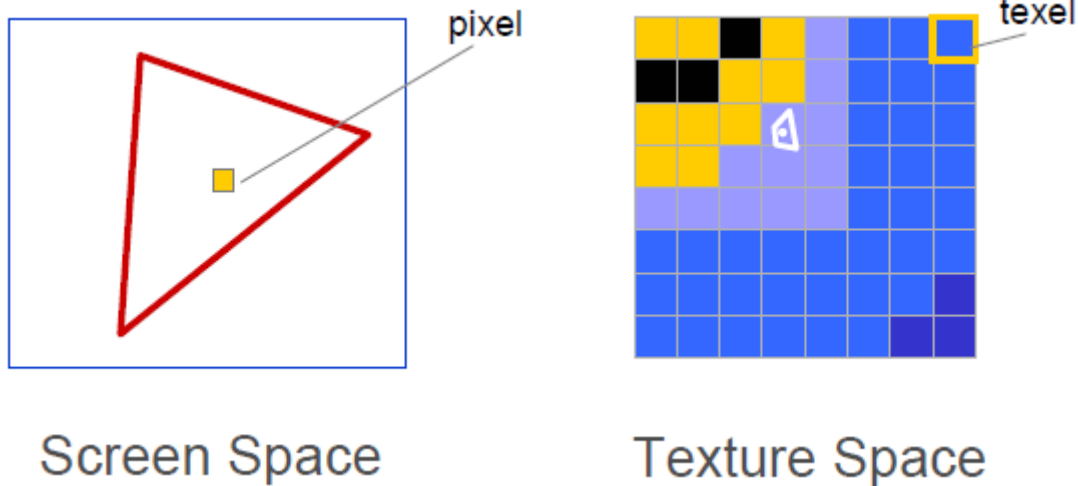
E' necessario applicare una correzione prospettica (non vediamo le equazioni)



Texture filtering

ALIASING

MAGNIFICATION



un pixel della scena equivale a meno di un texel

Texture filtering

ALIASING

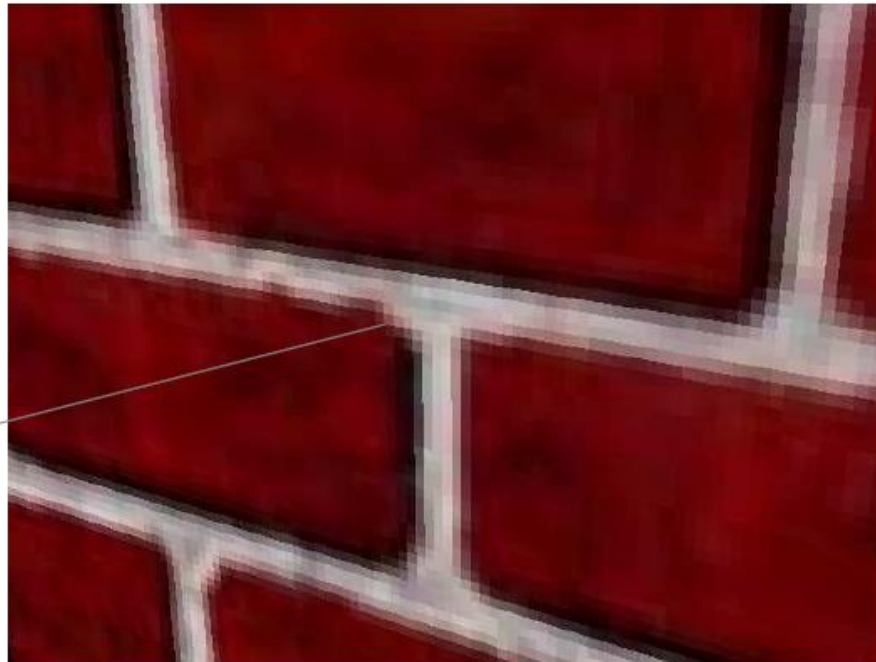
MAGNIFICATION

NEAREST FILTERING: per il pixel si seleziona il colore del texel



texture 128x128

"si vedono i texel !"

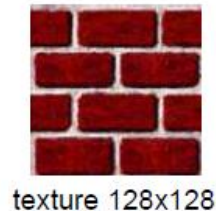


Texture filtering

ALIASING

MAGNIFICATION

BILINEAR INTERPOLATION: per il pixel si seleziona la media pesata dei 4 texel vicini



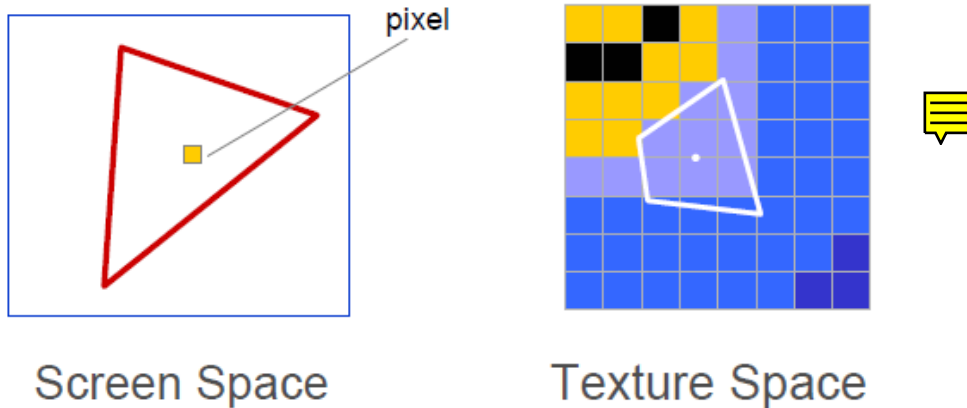
qualità migliore, ma più lento !!!



Texture filtering

ALIASING

MINIFICATION



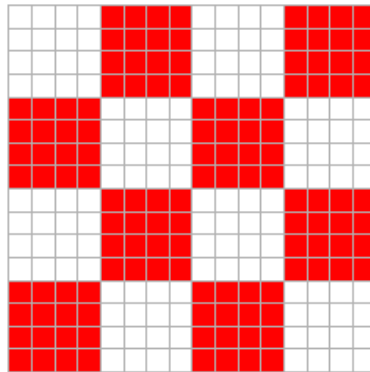
un pixel della scena equivale a più di un texel

più problematico

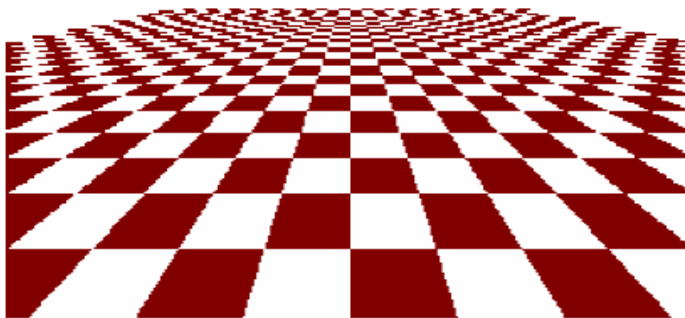
Texture filtering

ALIASING

MINIFICATION

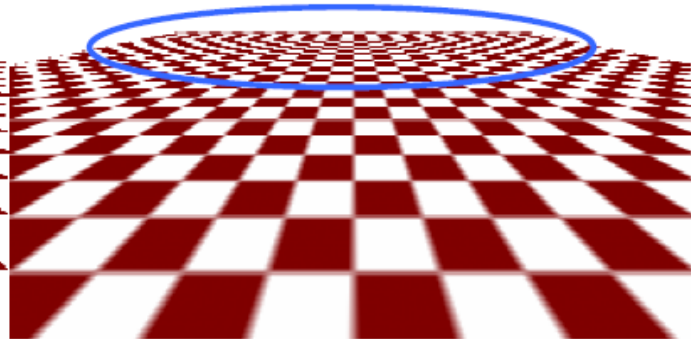


Nearest Filtering



Bilinear interpolation

non risolve il problema

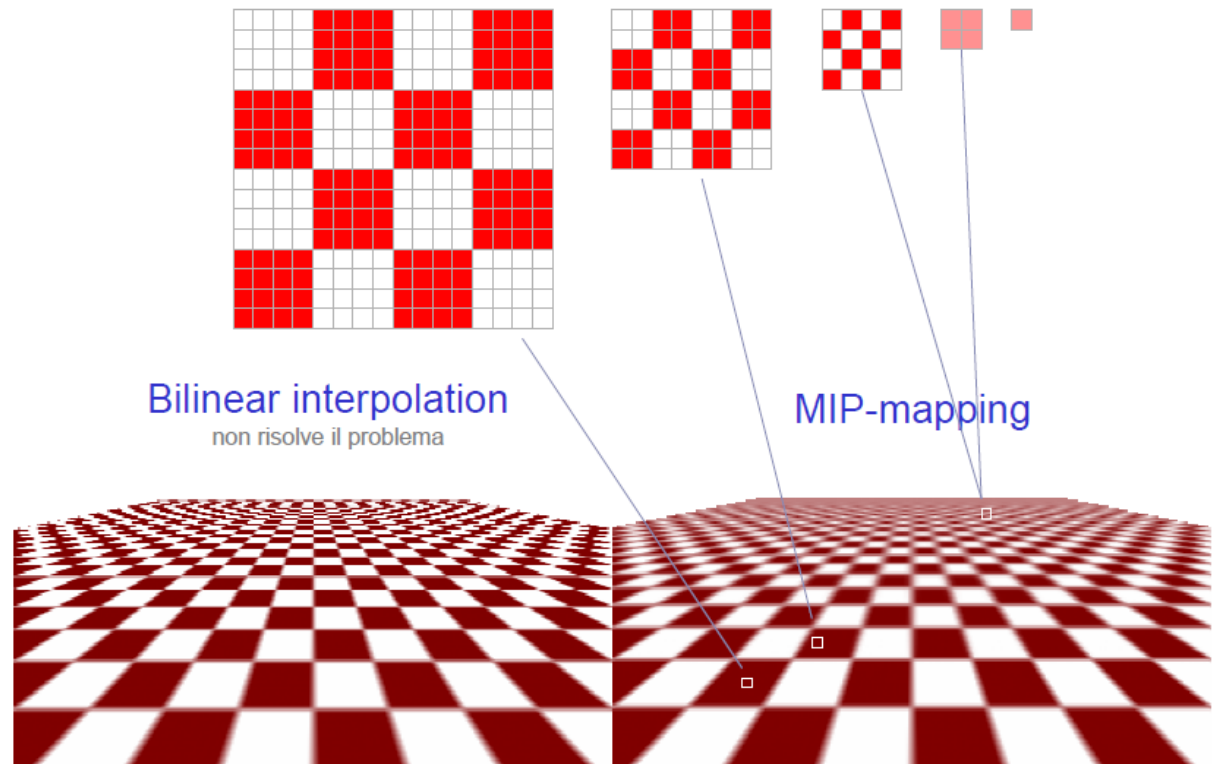


Texture filtering

ALIASING

MINIFICATION

MIP-Mapping: insieme di texture precalcolate (più piccole, multi-risoluzione, con filtri antialiasing) associate ad una texture principale; ha lo scopo di aumentare la velocità di rendering e di ridurre gli artefatti a scapito di una maggiore richiesta di memoria. Durante il rendering, dato un pixel da colorare, si sceglie la texture nella mipmap al livello di dettaglio che minimizza la minificazione e si usa interpolazione bilineare.



Texture filtering

ALIASING

MINIFICATION

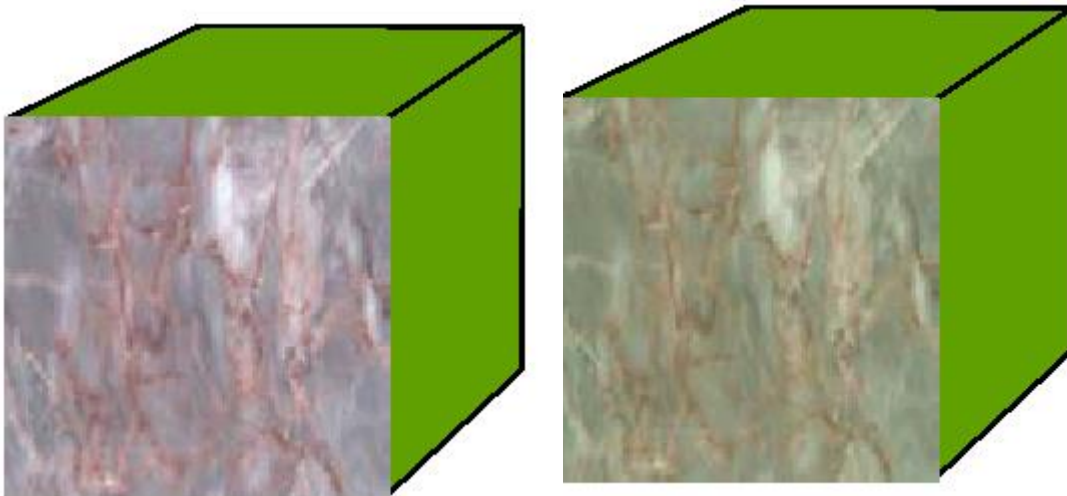
Anisotropic filtering: filtro anisotropico (non uniforme in ogni direzione, al contrario dei filtri isotropi visti precedentemente), causa meno sfocamento in corrispondenza di superfici inclinate rispetto alla telecamera conservando maggiore dettaglio. Il filtro anisotropico è in pratica in grado di interpolare una texture in modo non lineare nelle due direzioni.



Texture Mapping e colore superfici

- Texture trattata come una decalcomania opaca (impedisce di vedere il colore di base dell'oggetto)
 - Colore di base dei pixels corrispondenti **sostituito** con nuovo colore dovuto alla texture
- Texture trattata come una decalcomania translucida con colori miscelati a quelli del colore di base
 - Colore di base dei pixels corrispondenti **aggiunto con** nuovo colore dovuto alla texture
- ... vari gradi di “trasparenza” intermedi

Esempio: texture mapping opaco e più trasparente



Gradi crescenti di
trasparenza...

20%



Texture Mapping e colore superfici

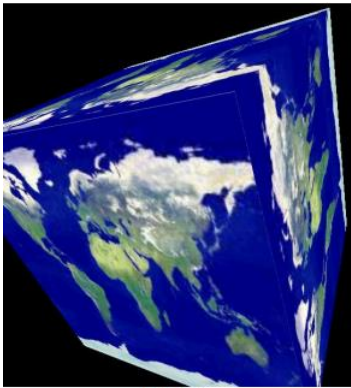
- **Ogni pixel della texture ha quattro informazioni legate al colore:**
 - componente rossa (R)
 - componente verde (G)
 - componente blu (B)
 - componente di trasparenza (A)
- **Componente A indica livello di trasparenza del pixel della texture, tra 0 e 1:**
 - $A=0$: superficie completamente opaca
 - $A=1$: superficie completamente trasparente
- **Così come i colori della texture si possono combinare con i colori del pixel anche il valore del parametro alpha della texture si può combinare con il valore alpha del pixel**

Texture Mapping e colore superfici

- Più nel dettaglio: modalità di combinazione della texture (anche in OpenGL) con il colore del frammento (pixel) dovuto all'illuminazione

C_t = colore texture

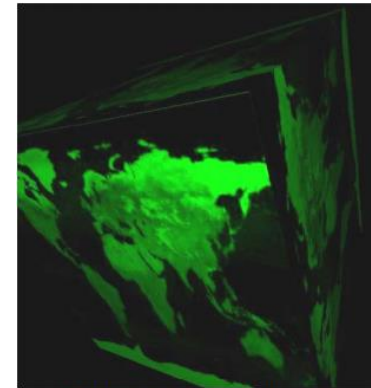
C_f = colore frammento (pixel) causato dall'illuminazione



Replace

$$C = C_t$$

(scarta il colore dell'illuminazione)



Modulate with a Green Cube

$$C = C_f * C_t$$

(modulazione)

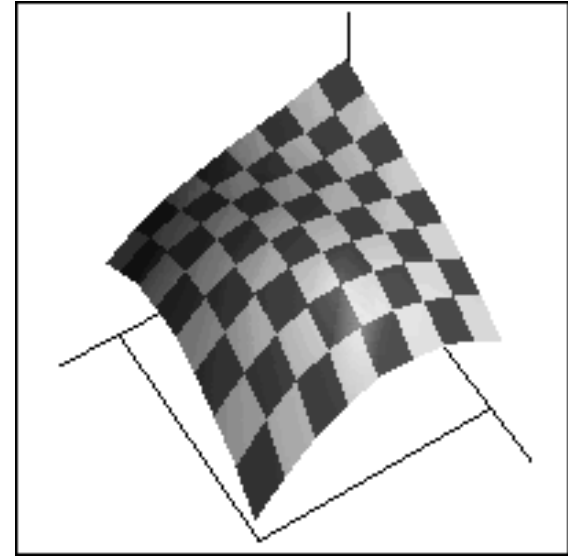
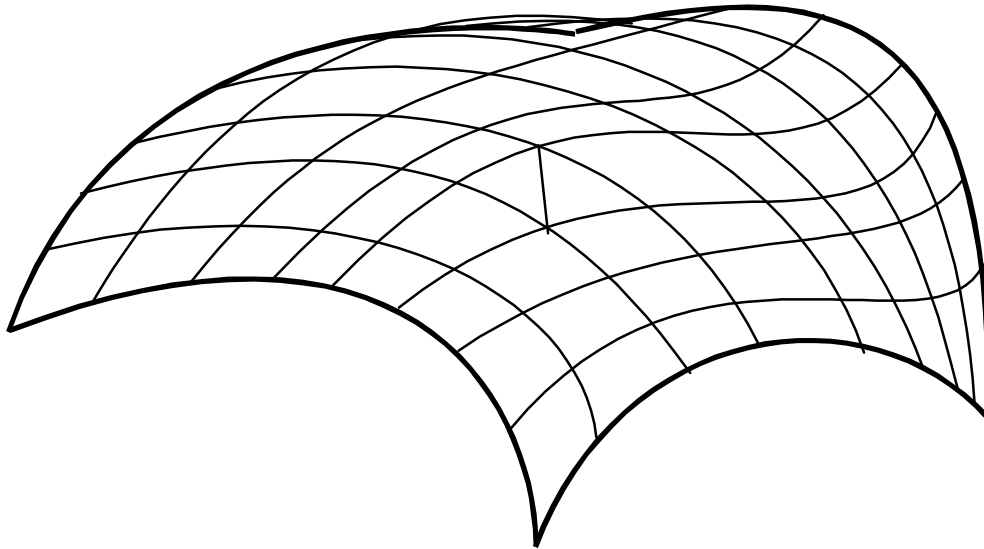
tabella riassuntiva

C_c = un colore esterno

Function	RGB	Alpha
Replace	C_t	A_t
Modulate	$C_f C_t$	$A_f A_t$
Decal	$C_f (1 - A_t) + C_t A_t$	A_f
Blend	$C_f (1 - C_t) + C_c C_t$	$A_f A_t$
Add	$C_f + C_t$	$A_f A_t$

Texture Mapping - esempi

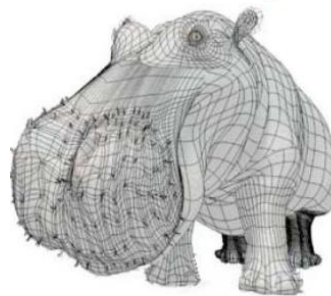
Un altro esempio...



Un limite

- Texture mapping quando applicato a superfici generiche provoca deformazioni dell'immagine originaria (non si conserva la distanza)

Texture Mapping - esempi



geometric model



*geometric model
+
shading*



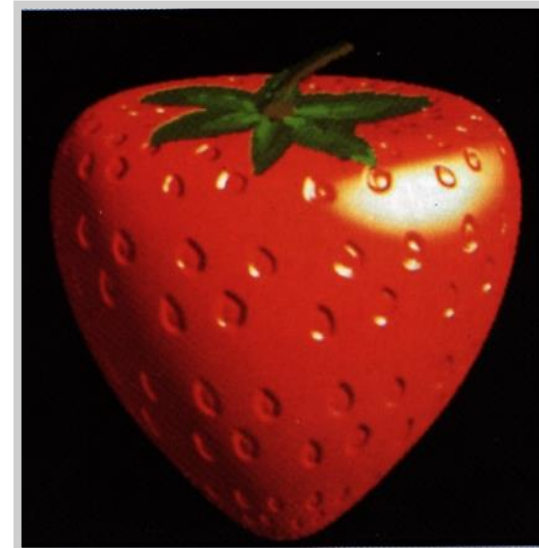
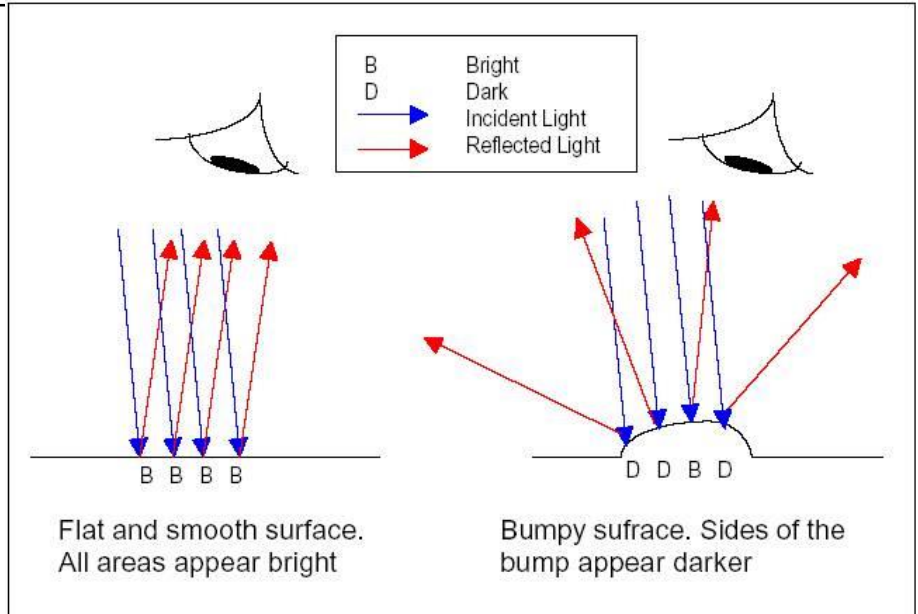
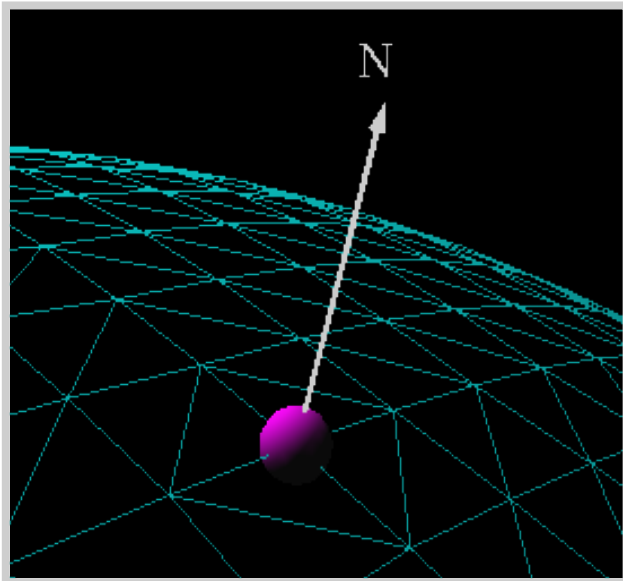
*geometric model
+
shading
+
textures*

Bump mapping

Idea:

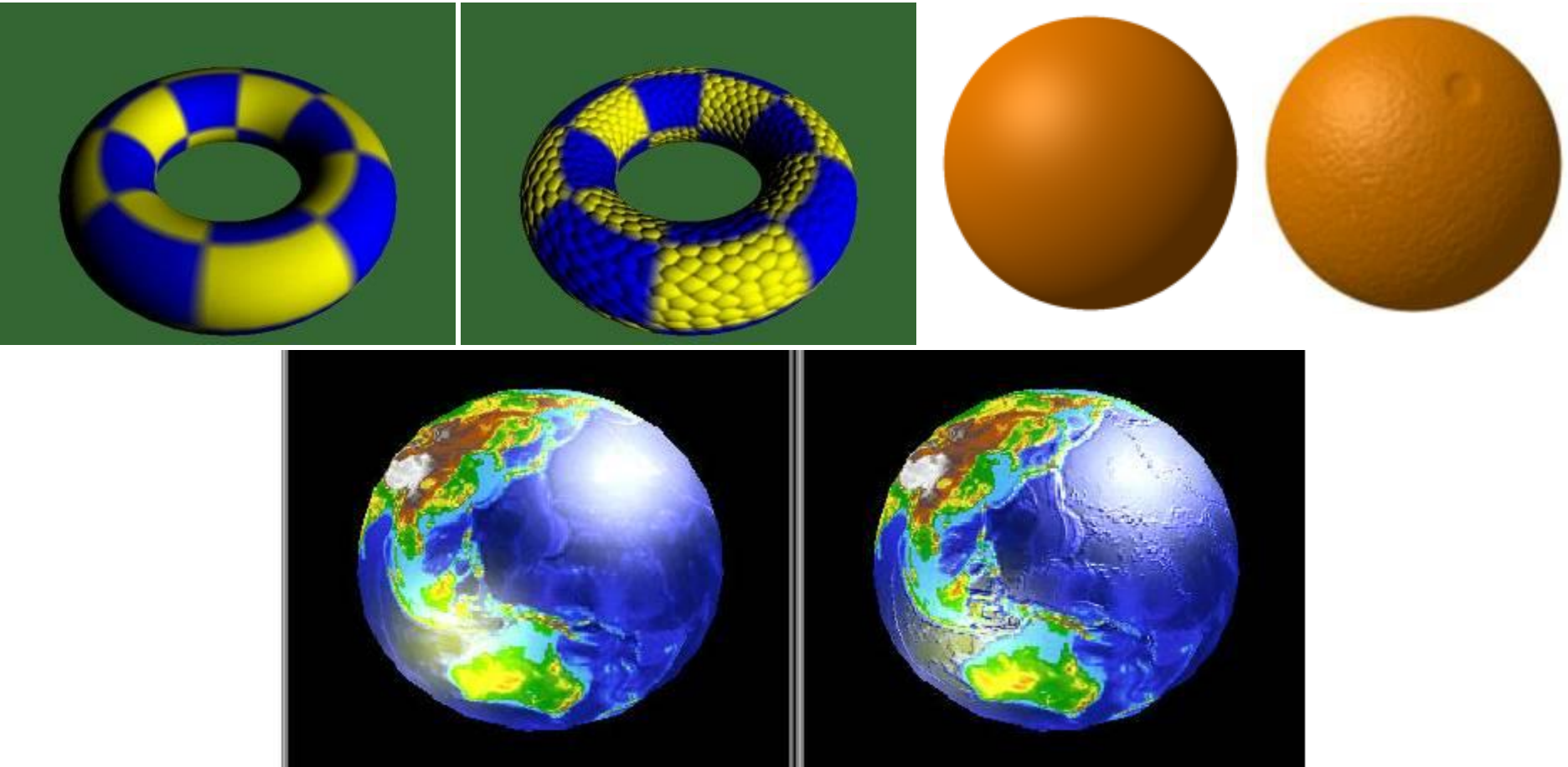
fare apparire una superficie liscia
come se fosse “rugosa”
modificando le normali alla superficie
utilizzando valori di perturbazione
memorizzati in una texture

Esempio



Bump mapping

Con e senza bump mapping...



Blinn, James F. "Simulation of Wrinkled Surfaces", Computer Graphics, Vol. 12 (3), pp. 286-292 SIGGRAPH-ACM (August 1978)

Environment mapping

- tecnica per rendere l'effetto di riflessione su una superficie speculare
- utilizza il modello di illuminazione locale e il texture mapping
- la scena si racchiude in un cubo e si generano sei immagini corrispondenti a sei telecamere poste nel centro della scena e con piano immagine coincidente con le facce del cubo
- le immagini si compongono in una texture che prende il nome di environment map che viene mappata sugli oggetti della scena

