

INFORMATICA GRAFICA – SSD ING-INF/05

Sistemi di elaborazione delle informazioni

a.a. 2007/2008

CAP 6. Rendering grafico

Lighting and Shading

Illuminazione

luce finale

=

ambiente

+

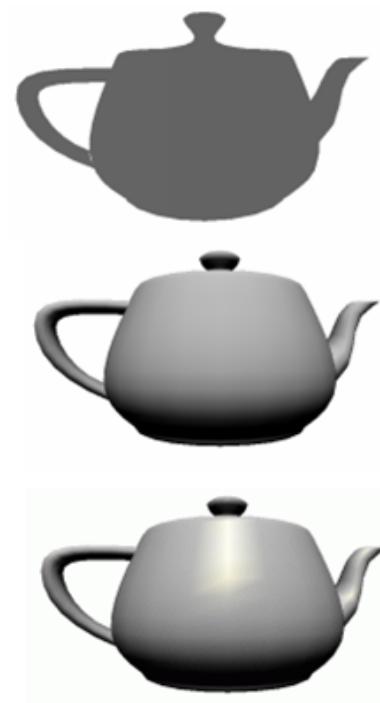
riflessione diffusa

+

riflessione speculare

+

emissione



Modelli di illuminazione

- Rendering tenendo conto dei **principi fisici** che regolano la diffusione e riflessione dei raggi luminosi che incidono sulle superfici esterne degli oggetti della scena.
- E' necessario tenere conto non solo della geometria, ma anche degli **aspetti ottici** e fisiologici correlati alla diffusione, riflessione e percezione della luce.
- La **radiazione incidente** su un corpo può essere:
 - assorbita
 - trasmessa (\Rightarrow passa attraverso)
 - riflessa - sia diffusamente che specularmente

Modelli di illuminazione

- Calcolare l'intensità luminosa sulle superfici di una scena ... ci serve un **modello di illuminazione**
- Sostanzialmente una particolare equazione che ci consente di **calcolare** l'intensità luminosa in un punto dello spazio come funzione:
 - dell'intensità della radiazione incidente
 - delle proprietà geometriche e fisiche della superficie a cui il punto appartiene.

Sorgenti di luce

- Sorgenti possono essere posizionate all'**infinito** oppure in un punto al finito.
- Di conseguenza, i raggi di luce saranno **paralleli o divergenti**, rispettivamente.
- Le sorgenti luminose possono essere **puntiformi** o **distribuite** su una estensione finita.
- Le scene possono avere una sola sorgente o numerose sorgenti di luce.
- Si può immaginare che la luce emessa abbia **uguale intensità in tutte le direzioni**, oppure che la sorgente emetta in qualche **direzione preferenziale** (sorgenti luminose direzionali).

Radiazioni luminose

- **Radiazione assorbita**: è la porzione di radiazione incidente assorbita da un corpo - se un corpo assorbisse tutta la luce incidente non sarebbe visibile (**corpo nero**).
- **Radiazione trasmessa**: è la radiazione incidente trasmessa, in parte, attraverso il corpo quando questo è **trasparente**
- **Radiazione riflessa diffusamente**: porzione di luce incidente che è assorbita dallo strato superficiale del corpo, e viene poi **emessa** in tutte le direzioni consentite dalla posizione e orientamento delle superfici esterne
 - Ø il corpo, nel caso di diffusione perfetta della radiazione, si comporta esattamente come un **corpo emittente**
- **Radiazione riflessa specularmente**: la radiazione incidente non attraversa lo strato esterno del corpo, ma è riflessa direttamente nel piano definito dalla direzione della **normale** alla superficie nel punto considerato e dalla direzione della luce incidente

Modello di Phong

- Modello dovuto a **Phong** Bui-Tran, prima metà degli anni '70
- Schema fisico di interazione **semplificato**:
 - Solo sorgenti **puntiformi**
 - No **inter-riflessioni**
 - Calcolo locale dell'equazione di illuminazione
 - Approssimazione con due costanti della funzione di riflessione
- Simula il comportamento di materiali opachi, cioè non modella la *trasmissione* - no materiali **trasparenti** o semi-trasparenti!
- Metodo: semplificazione del fenomeno della riflessione usando le leggi della fisica che regolano la riflessione speculare (**Fresnel**) e la riflessione diffusa (**Lambert**)

Modello di Phong

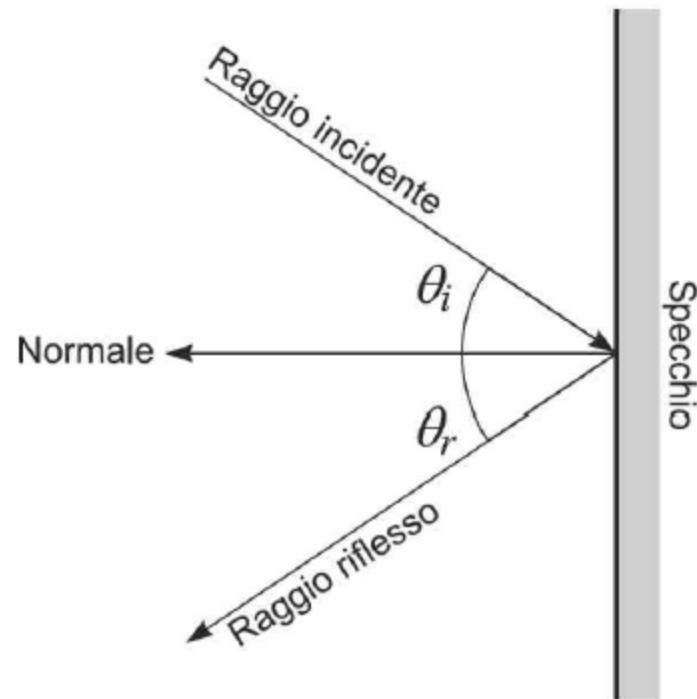
- Quando un raggio di luce passa da un mezzo ad un altro con diverso indice di rifrazione raggiunta la superficie di separazione parte del raggio viene **trasmessa** e parte **riflessa**

- La somma delle energie dei due raggi è uguale all'energia del raggio incidente

- Semplifico: non c'è trasmissione e si ha solo **riflessione**

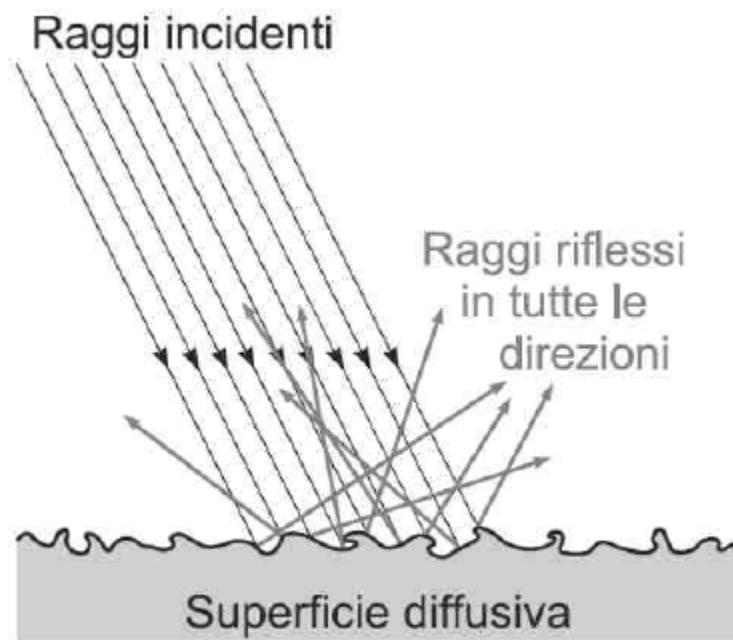
- L'angolo di **incidenza** è uguale all'angolo di **riflessione**

- E' un buon modello per materiali molto **lisci e lucidi**



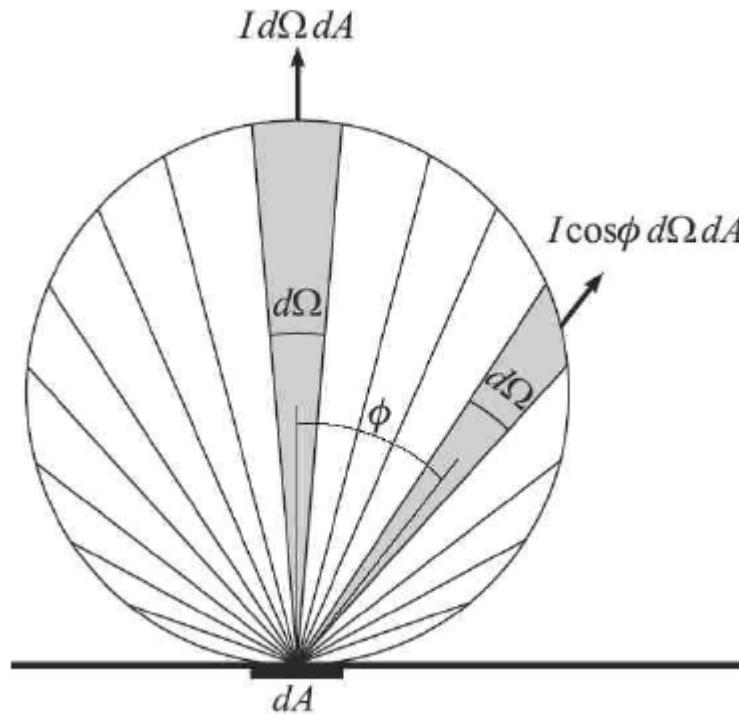
Riflessione diffusa: Legge di Lambert

- Materiali **molto opachi** (es. gesso e legno) hanno una superficie che, a livello microscopico, presentano piccole sfaccettature (microfacets) che riflettono un raggio incidente in una direzione casuale



Riflessione diffusa: Legge di Lambert

- ☐ Integrando su scala **macroscopica** la luce si riflette uniformemente verso tutte le direzioni, con intensità proporzionale al rapporto tra la **direzione** del raggio incidente e la **normale** alla superficie approssimante in quel punto



Modellazione della riflessione diffusiva

- ❖ Sorgenti luminose puntiformi:

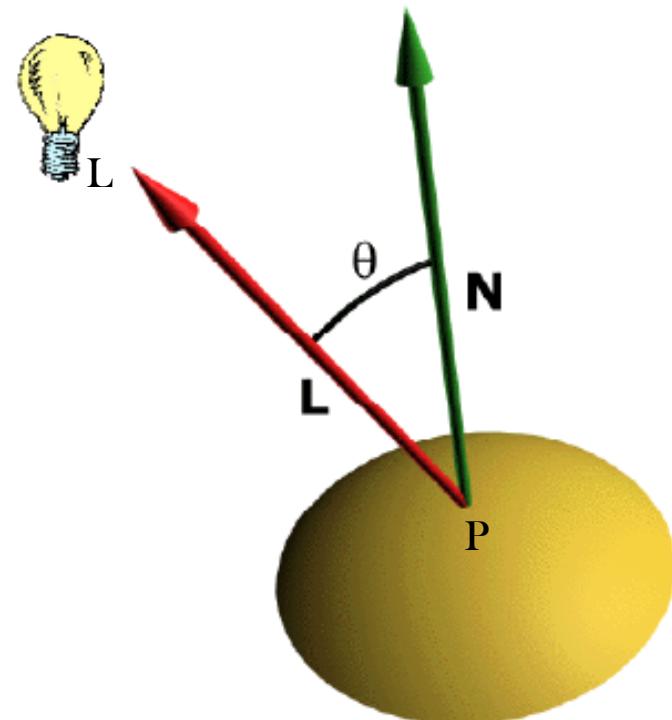
- ❖ posizione nella scena
- ❖ intensità della luce emessa

- ❖ Per calcolare in P con normale \vec{N} :

$$\vec{L} = |L - P|$$

- ❖ Dipendenza solo da θ

$$\cos(\theta) = \vec{L} \cdot \vec{N}$$



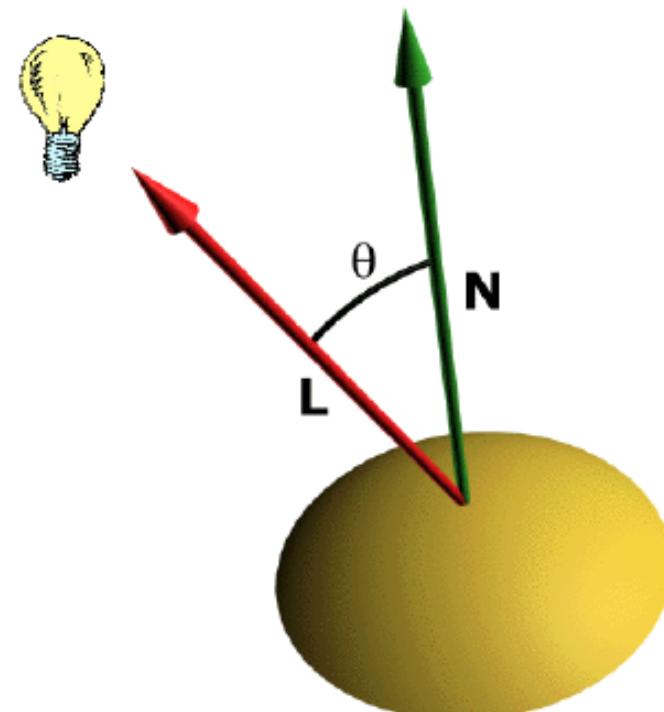
Modellazione della riflessione diffusiva

- ❖ Si approssima la funzione di riflessione diffusa della superficie come una costante k_d dipendente dal materiale
- ❖ Equazione di illuminazione (solo diffusiva)

$$I = I_p k_d \cos \theta$$

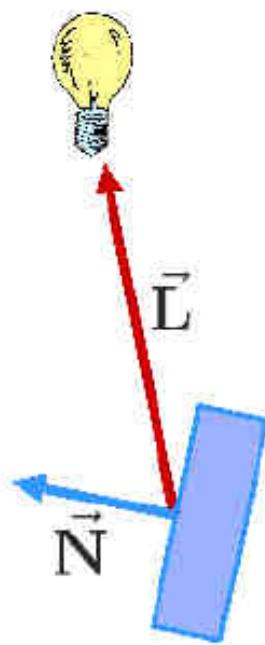
o meglio

$$I = I_p k_d (\vec{N} \cdot \vec{L})$$

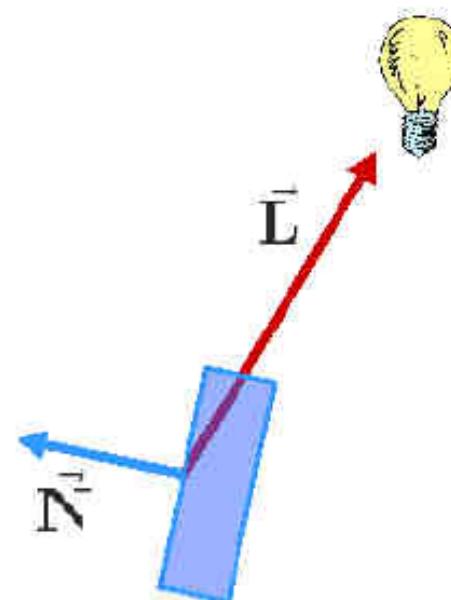


Modellazione della riflessione diffusiva

- ❖ Si considera solo per valori di θ compresi tra 0 e $\pi/2$



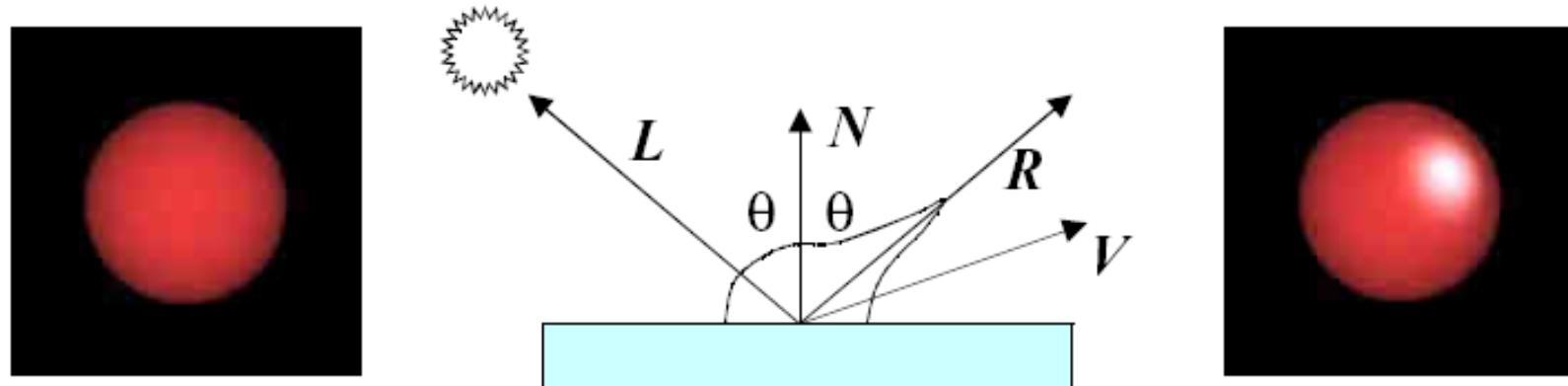
OK



NO

Modellazione della riflessione speculare

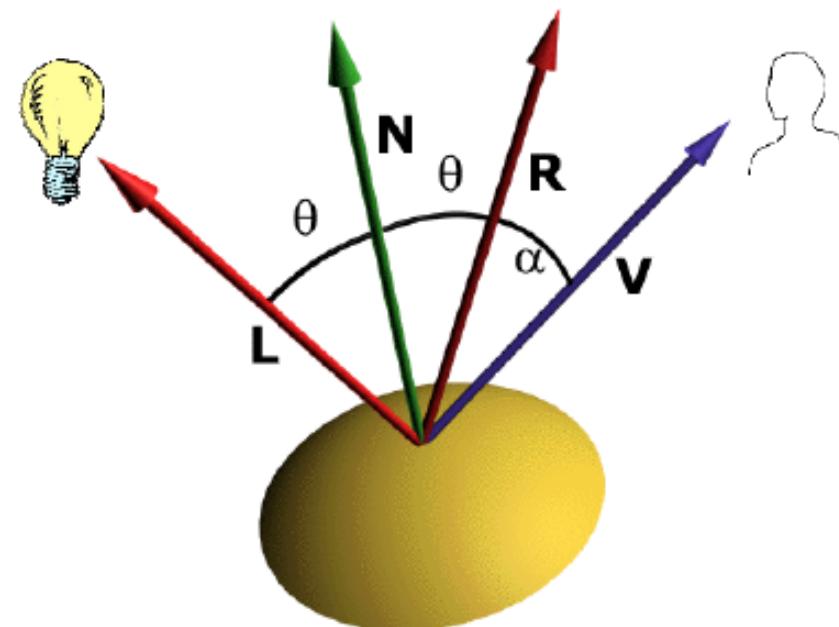
- ❖ Novità sostanziale: *riflettore non perfetto*



- ❖ Approssimazione empirica di una riflessione più realistica rispetto alla legge di Fresnel
- ❖ Consequenza: *specular highlight*

Modellazione della riflessione speculare

- ❖ Dipendenza dall'angolo α compreso tra la direzione di riflessione ideale e la direzione di vista
- ❖ Riflessione massima per $\alpha = 0$
- ❖ Decadimento più o meno rapido all'aumentare di α

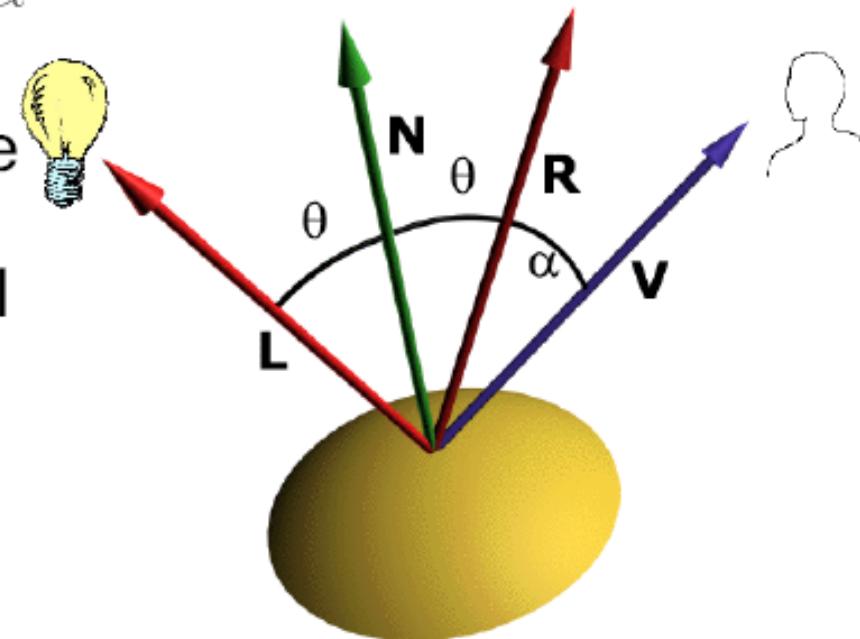


La riflessione speculare di un oggetto è particolarmente importante nelle animazioni, dato che questa necessariamente si modifica quando l'oggetto (o il view point!) si muove – cambia θ e/o α . Determina fortemente il realismo della scena!!

Modellazione della riflessione speculare

- ❖ Questo comportamento si modella elevando alla n il coseno dell'angolo α
- ❖ Il parametro n è detto esponente di riflessione speculare (*specular reflection exponent*) del materiale
- ❖ Il vettore \vec{R} si calcola

$$\vec{R} = 2\vec{N}(\vec{N} \cdot \vec{L}) - \vec{L}$$

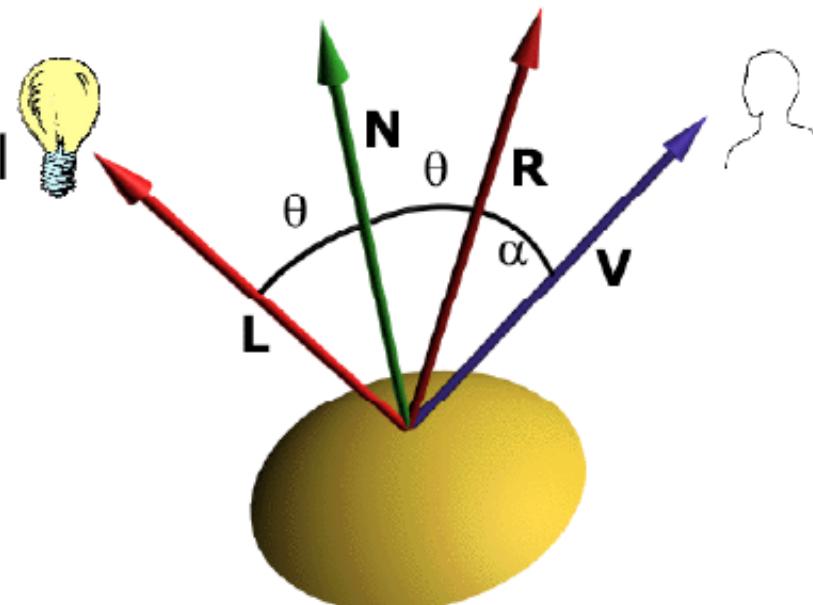


Modellazione della riflessione speculare

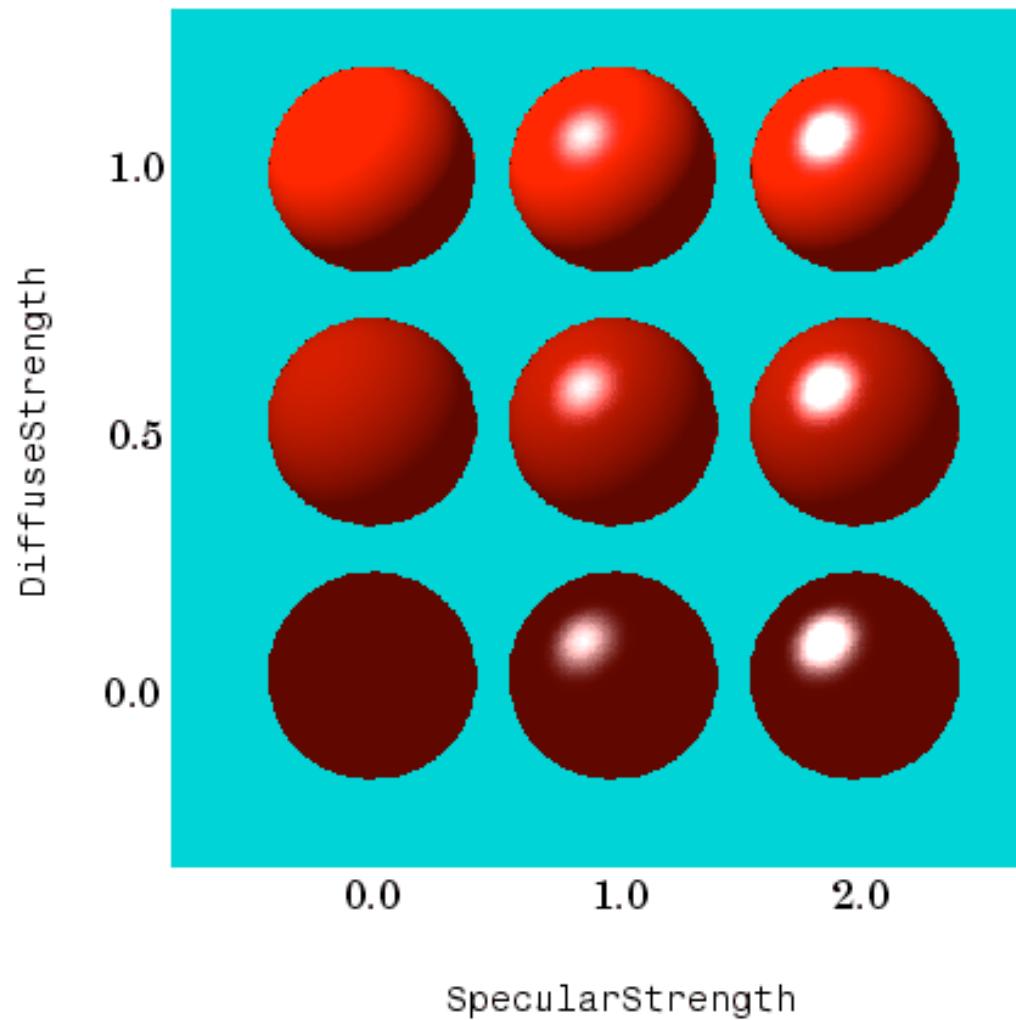
- ❖ Equazione di illuminazione (solo speculare)

$$I = I_p k_s \cos^n \alpha$$

- ❖ Parametro k_s modella il comportamento della superficie insieme a n



Esempio: effetto della riflessione speculare e diffusa



Modellazione della componente ambientale

- ❖ Le inter-riflessioni tra oggetti diversi nella scena non sono modellate in modo accurato dal modello di Phong
- ❖ Sono approssimate dalla componente:

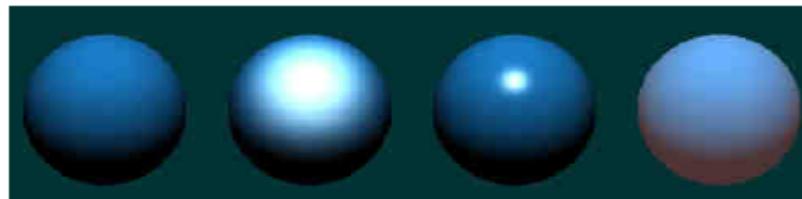
$$I = I_a k_a$$

- ❖ I_a modella la radiazione luminosa totale emessa nella scena
- ❖ k_a modella la riflettività del materiale
- ❖ I_a è costante per tutti i punti di tutti gli oggetti

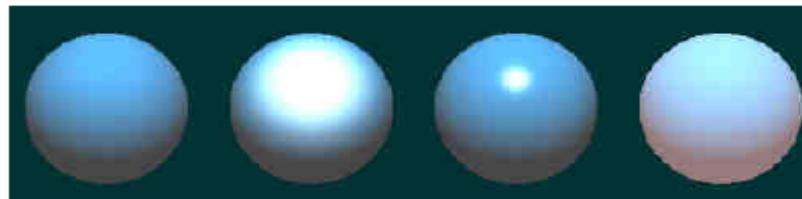
Modellazione della componente ambientale

- ❖ La componente ambientale aggiunge realismo alla scena

Senza



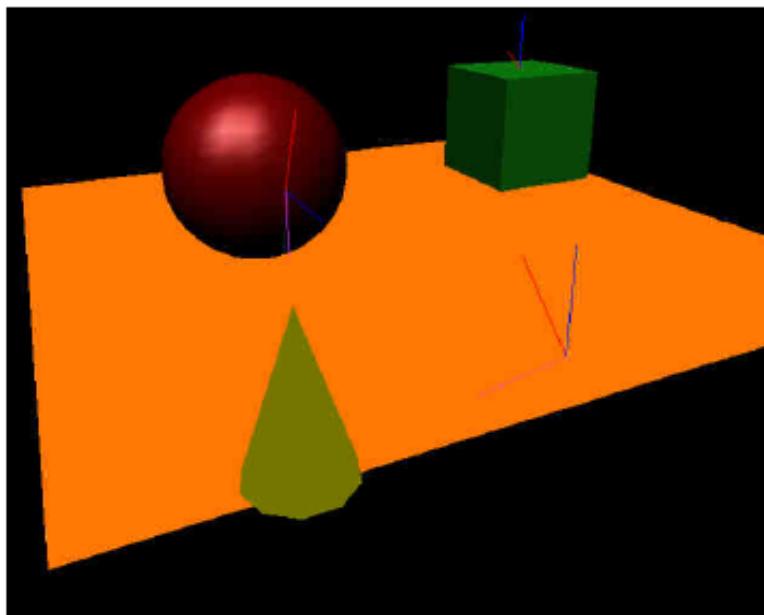
Con



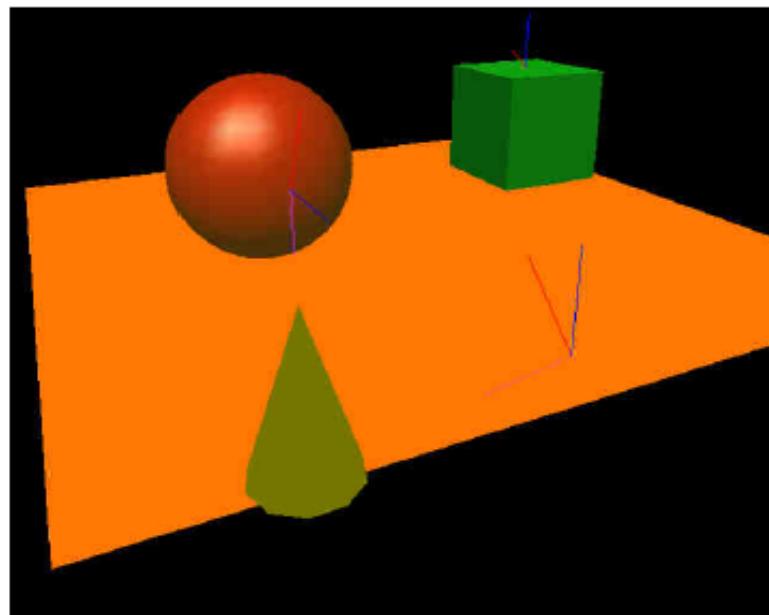
Una luce ambiente è priva di *direzione*: illumina in modo uniforme tutti gli oggetti presenti nella scena.

Modellazione della componente ambientale

- ❖ La componente ambientale aggiunge realismo alla scena



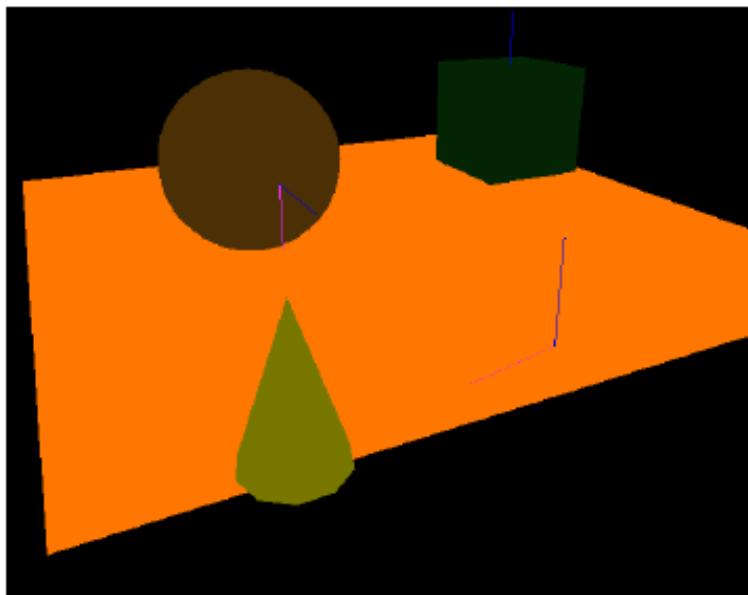
Senza



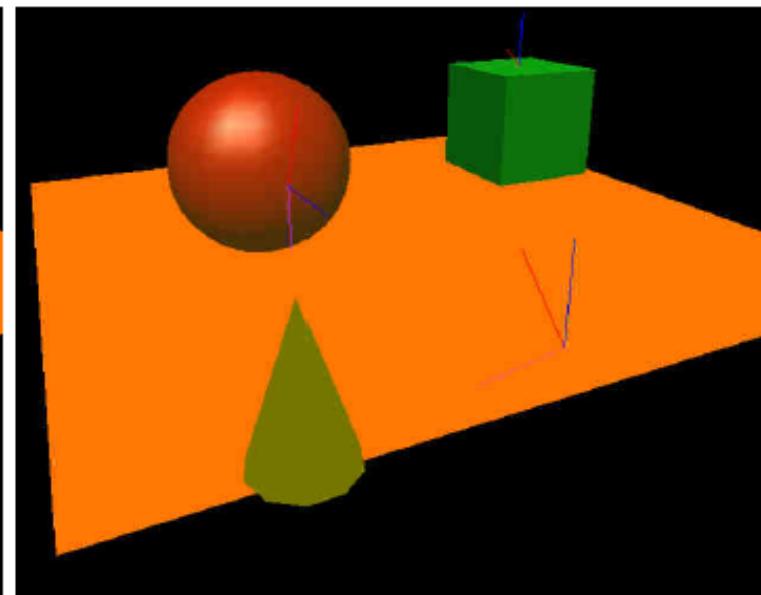
Con

Modellazione della componente ambientale

❖ Ma da sola non basta!



Solo ambientale



Con riflessioni

Mettere tutto insieme

- ❖ Tutti i contributi descritti si vanno a sommare per calcolare l'equazione di illuminazione
- ❖ Sommatoria su tutte le sorgenti luminose presenti nella scena

$$I = I_a k_a + \sum_p I_p [k_d (\vec{N} \cdot \vec{L}) + k_s (\vec{R} \cdot \vec{V})^n]$$

Radiazione ambientale

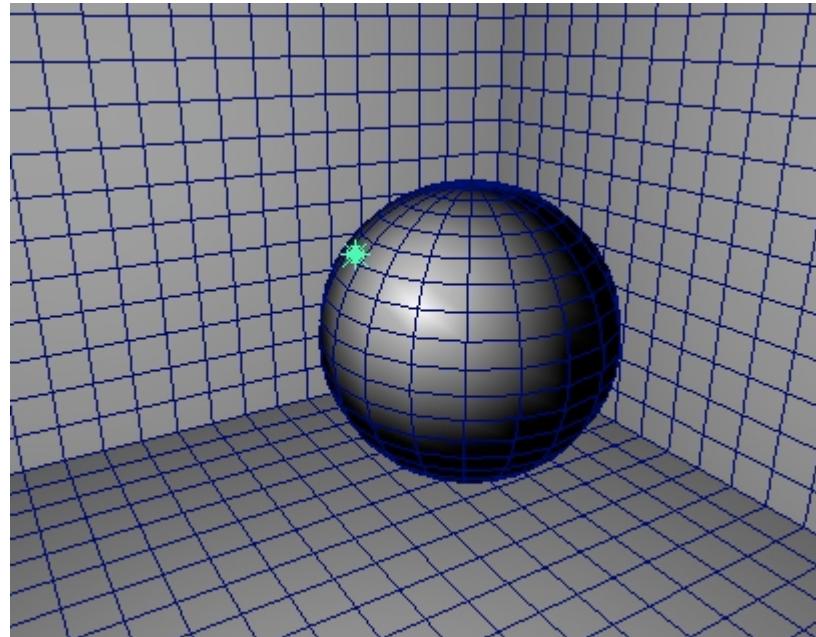
Radiazione diffusiva

Radiazione speculare

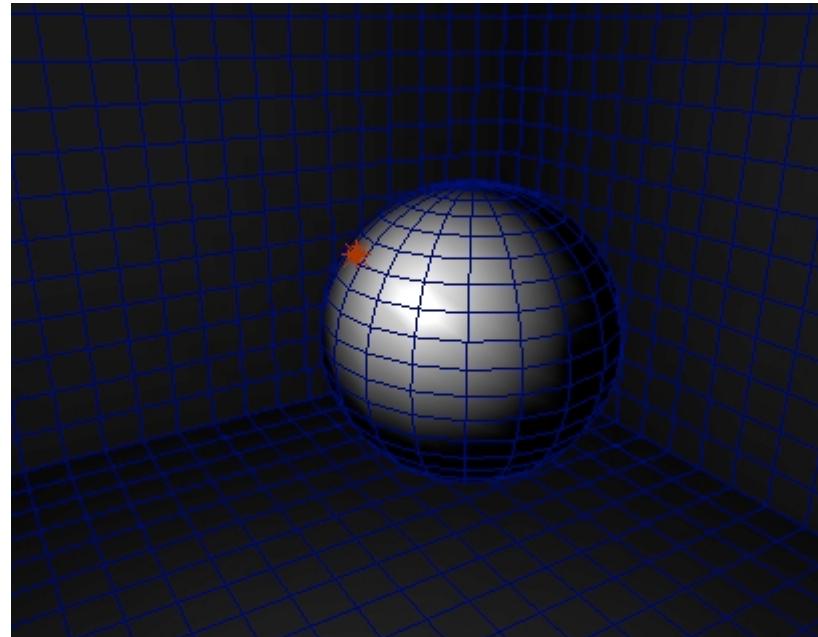
Attenuazione dell'illuminazione

- L'**attenuazione** definisce come l'intensità della luce diminuisce con la distanza dalla sorgente – solo per sorgenti posizionali
- *x OpenGL* - Possiamo definire una attenuazione costante, lineare o quadratica con la distanza
 - Attenuation factor = $1 / (kc + kl*d + kq*d^2)$
 - dove
 - kc** = constant attenuation factor (default = 1.0)
 - kl** = linear attenuation factor (default = 0.0)
 - kq** = quadratic attenuation factor (default = 0.0)

Attenuazione dell'illuminazione

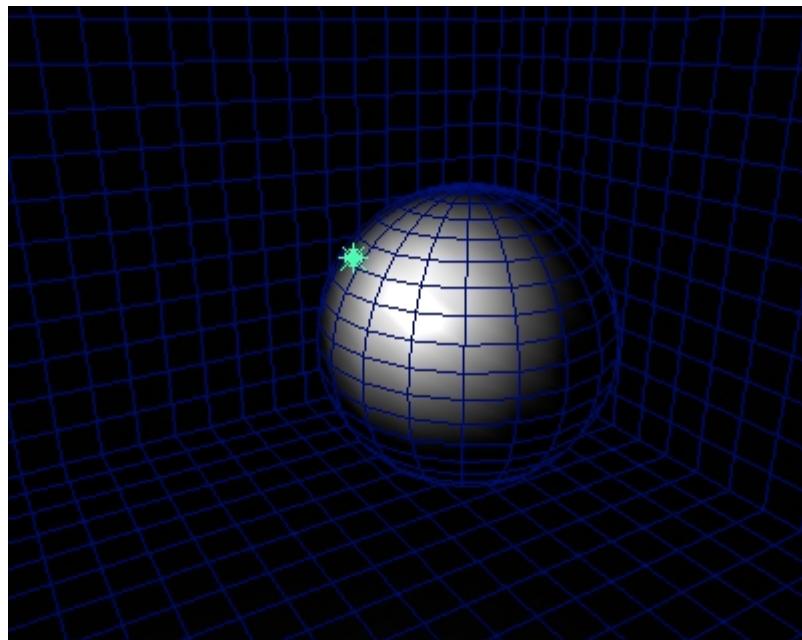


Nessuna attenuazione



Attenuazione lineare

Attenuazione quadratica



Mettere tutto insieme + attenuazione

- ❖ Si può tenere conto dell'attenuazione dell'intensità dell'illuminazione all'aumentare della distanza

$$f_{\text{att}} = \begin{cases} \frac{1}{c_1 + c_2 d_L + c_3 d_L^2} & \text{se } \frac{1}{c_1 + c_2 d_L + c_3 d_L^2} < 1 \\ 1 & \text{altrimenti} \end{cases}$$

- ❖ Inserendo il fattore di attenuazione

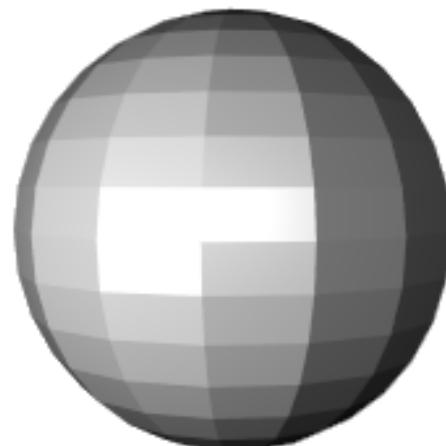
$$I = I_a k_a + \sum_p f_{\text{att}_p} I_p [k_d (\vec{\mathbf{N}} \cdot \vec{\mathbf{L}}) + k_s (\vec{\mathbf{R}} \cdot \vec{\mathbf{V}})^n]$$

Shading

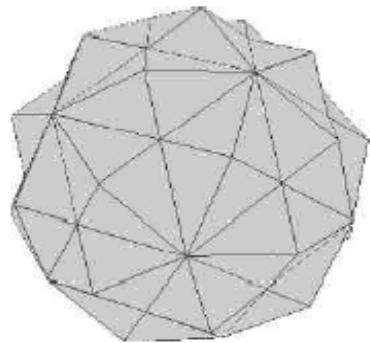
- Il modello di illuminazione di Phong ci fornisce un metodo empirico per capire *come* deve essere calcolata l'interazione tra luce e materia (combinazione di radiazione ambiente, diffusa e speculare).
- **Shading**: è il processo di modifica del colore in funzione dell'angolo e della distanza dalla/e sorgente/i di luce/i per creare un effetto realistico.

Flat Surface Rendering

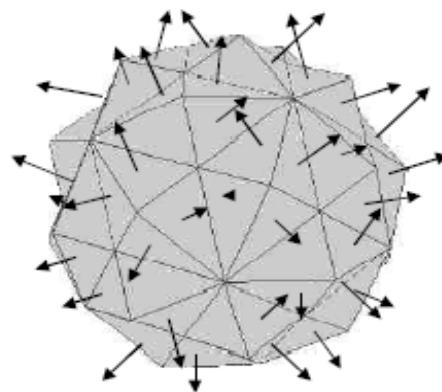
- E' il metodo più semplice da utilizzare per il rendering della superficie di un poligono
 - Lo stesso colore viene assegnato a tutte le parti della superficie
 - Il valore di illuminazione calcolato per un suo punto è usato su tutta la superficie.
- Il Flat surface rendering è estremamente veloce ... ma è quasi sempre irrealistico.



Flat Surface Rendering



Dato l'oggetto per cui calcolare l'equazione di illuminazione I ...



...calcolare le normali in ogni faccia...

...e calcolo I una sola volta per faccia



Flat Surface Rendering

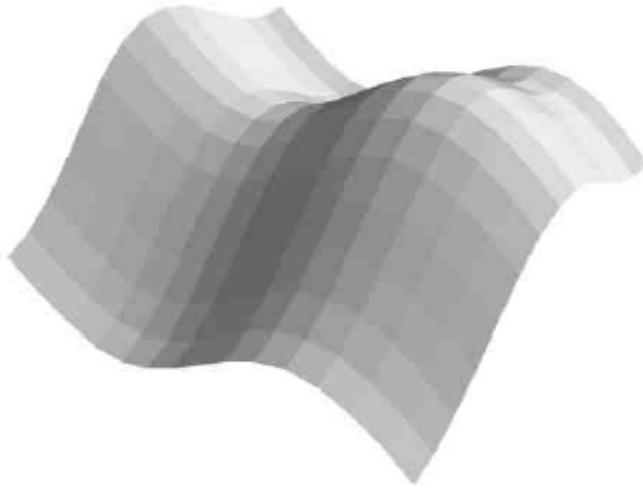
Se:

- ❖ sorgenti luminose solo direzionali ($I\vec{N} \cdot \vec{L} = k$ per tutta la superficie)
- ❖ osservatore a distanza infinita dalla scena (proiezione parallela) $\Rightarrow \vec{N} \cdot \vec{V} = k$ e $\vec{R} \cdot \vec{V} = k$ per tutta la superficie

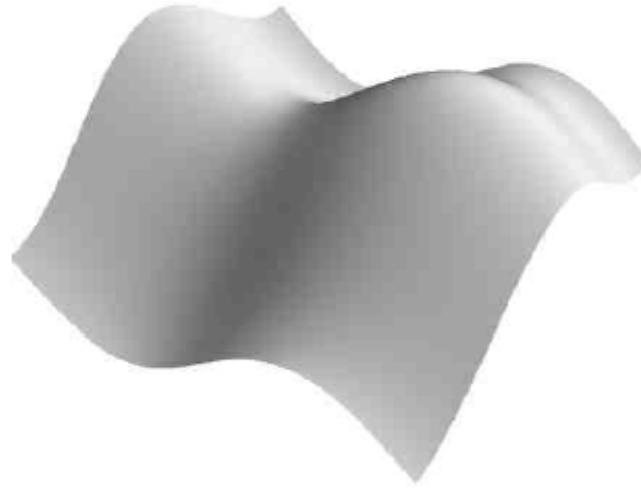
=> il Flat Surface Rendering è la migliore approssimazione possibile ... ma questo è un caso particolare!

Flat Surface Rendering

- ❖ Problema: il modello discreto rappresenta in modo approssimato una superficie curva e continua



Com'è

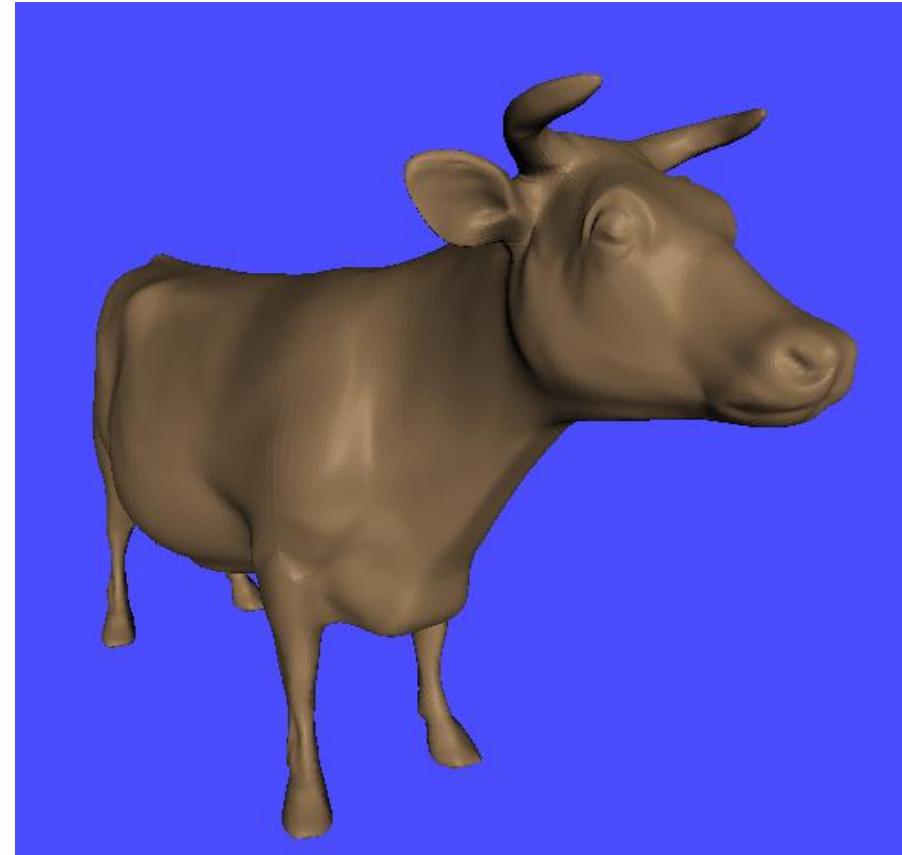


Come dovrebbe essere

... allora se aumentassi l'approssimazione?

Flat Surface Rendering

- Possiamo aumentare l'approssimazione aggiungendo *molti, moltissimi poligoni* ...



- ... tuttavia ... ancora qualche problema ... e in più ho un carico computazionale notevole!

Gouraud Surface Rendering

- Gouraud surface shading
- è stato sviluppato negli anni '70 da Henri Gouraud
- University of Utah, con Ivan Sutherland e David Evans
- Anche chiamato **intensity-interpolation surface rendering**
- I livelli di intensità sono calcolati in ciascun vertice e interpolati lungo la superficie

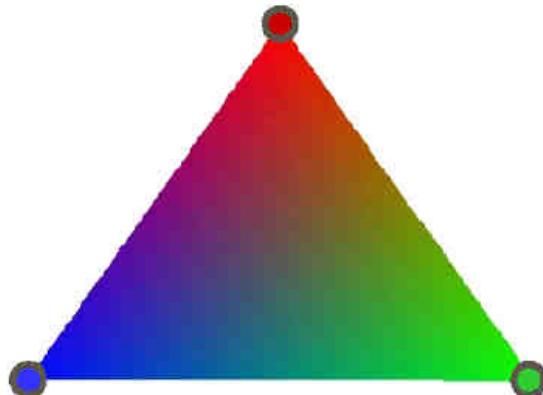


Gouraud Surface Rendering (cont...)

□ Nel rendering di un poligono, il **Gouraud surface rendering** opera come segue:

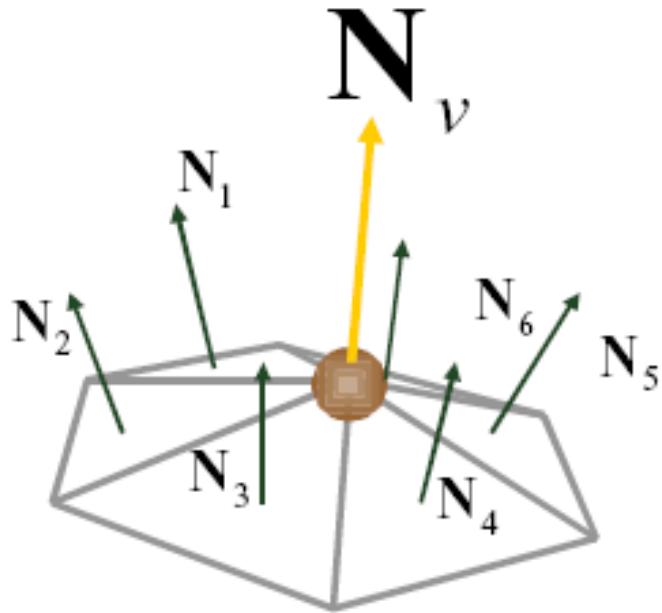
1. Determinare il vettore unitario normale per ciascun vertice del poligono
2. Applicare un modello di illuminazione a ciascun vertice del poligono, per trovare l'intensità della luce in quel punto
3. Interpola linearmente le intensità trovate per i vertici lungo la superficie *proiettata* del poligono

q Tipicamente il Gouraud shading è implementato all'interno del processo di *visible surface detection*



... ma quale è la normale
in un vertice? ->

Gouraud Surface Rendering (cont...)



- Il vettore unitario normale medio in v è dato da:

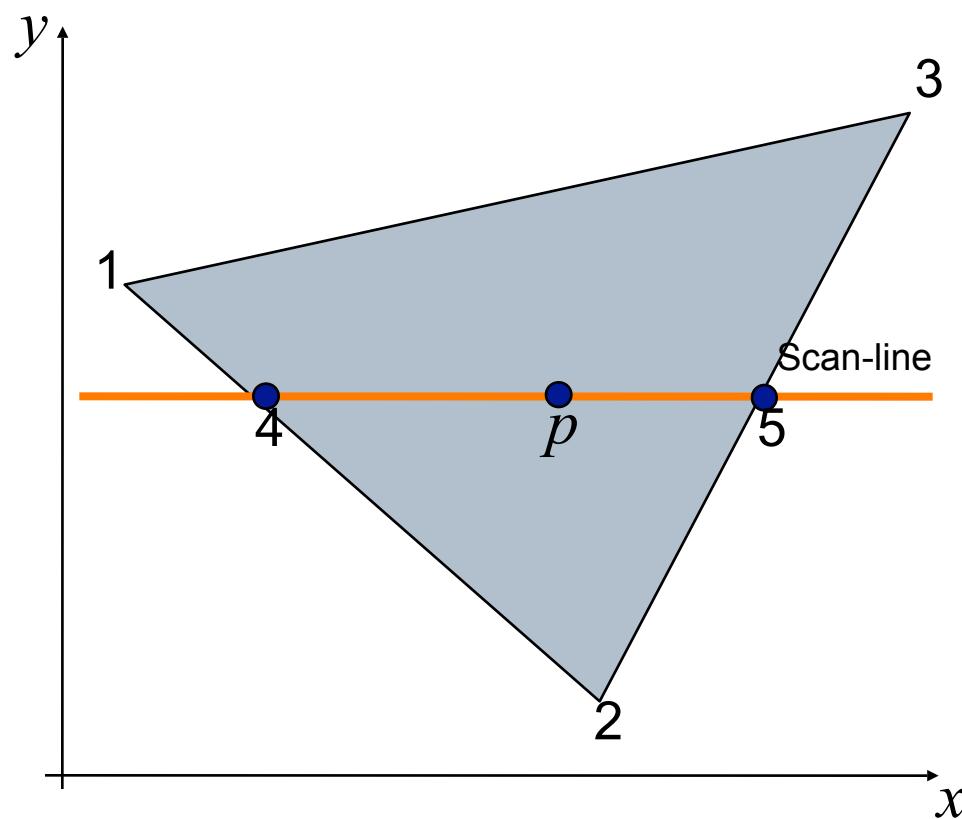
$$N_v = \frac{N_1 + N_2 + N_3 + N_4}{|N_1 + N_2 + N_3 + N_4|}$$

- o più generalmente:

$$N_v = \frac{\sum_{i=1}^n N_i}{\left| \sum_{i=1}^n N_i \right|}$$

Gouraud Surface Rendering (cont...)

- I valori di illuminazione sono calcolati attraverso una *interpolazione lineare* lungo ogni *scan-line*
- L'interpolazione viene applicata alle 3 componenti RGB separatamente

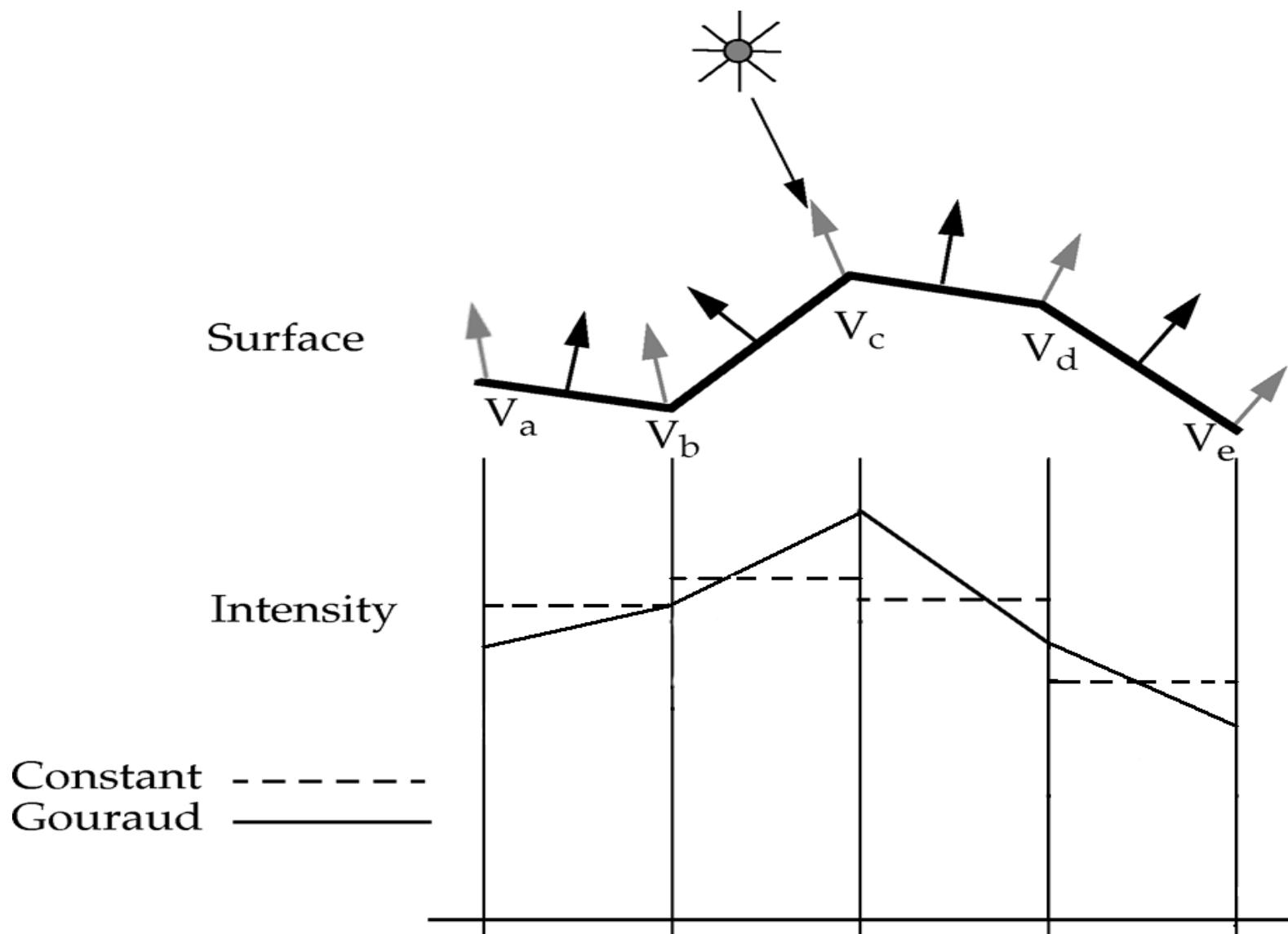


$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2$$

$$I_5 = \frac{y_5 - y_2}{y_3 - y_2} I_3 + \frac{y_3 - y_5}{y_3 - y_2} I_2$$

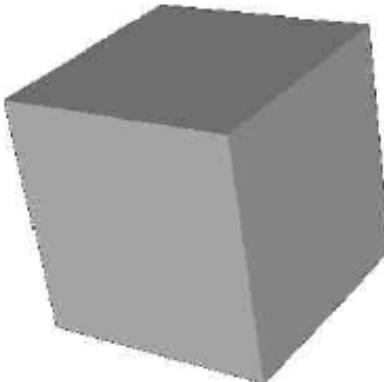
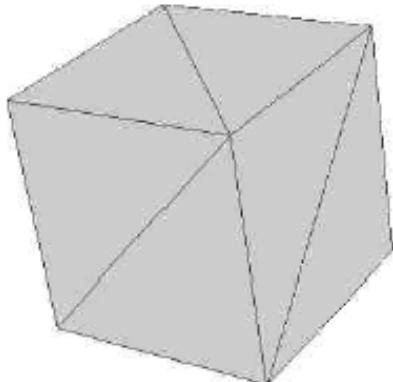
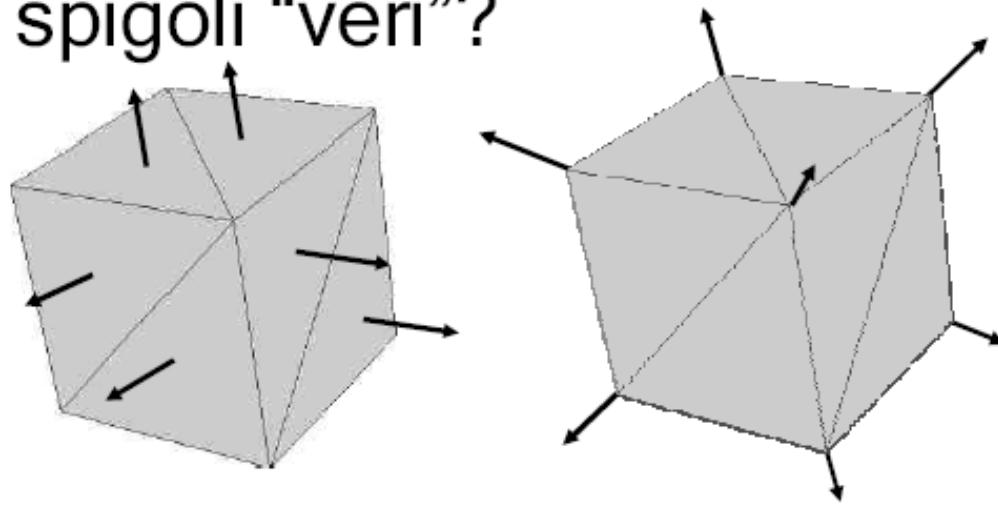
$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5$$

Vantaggi del Gouraud Surface Rendering

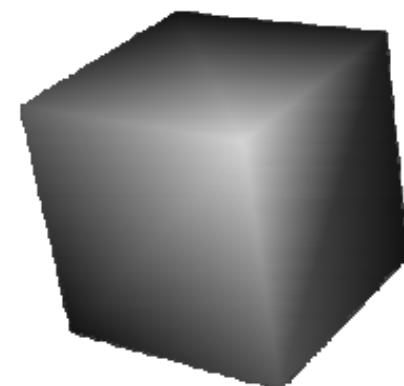


Gouraud Surface Rendering (cont...)

- ❖ Problema: gli spigoli “veri”?



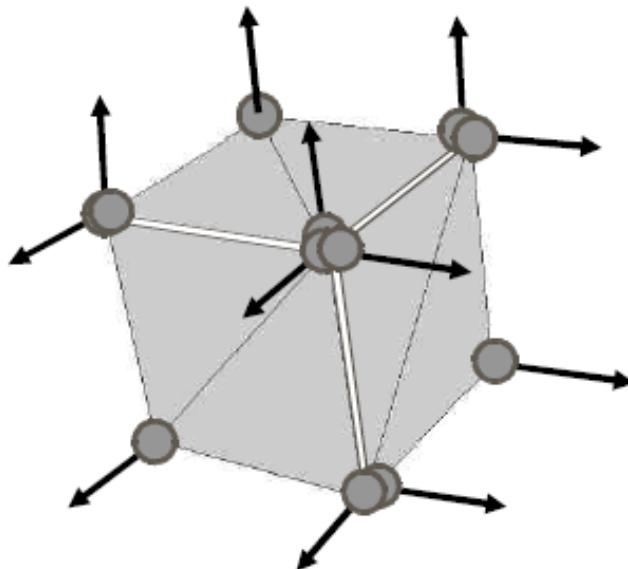
shading costante



Gouraud shading

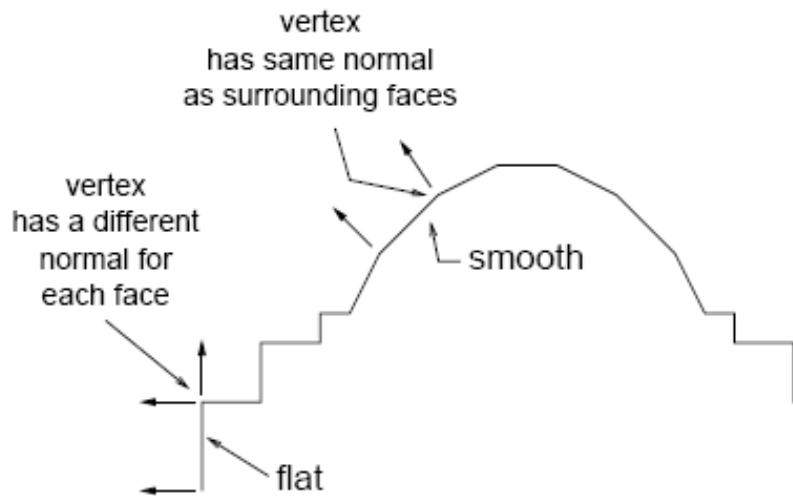
Gouraud Surface Rendering (cont...)

- ❖ Soluzione: si utilizzano normali diverse per i due lati dello spigolo
- ❖ La struttura dati deve memorizzare le adiacenze e le diverse tipologie



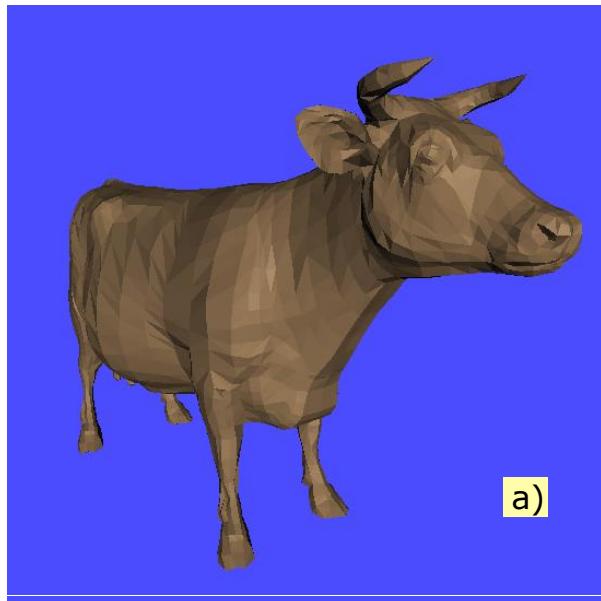
Gouraud Surface Rendering (cont...)

- In realtà la normale scelta nel vertice dipende se si è in presenza di facce con angoli *smooth* o *sharp*
- Quindi nel rendering di un poligono
 - Se questo ha un angolo netto con i poligoni adiacenti usa la normale del poligono in esame
 - Se invece i poligoni adiacenti hanno normali con un angolo più acuto allora usa la normale media su tutti i poligoni adiacenti



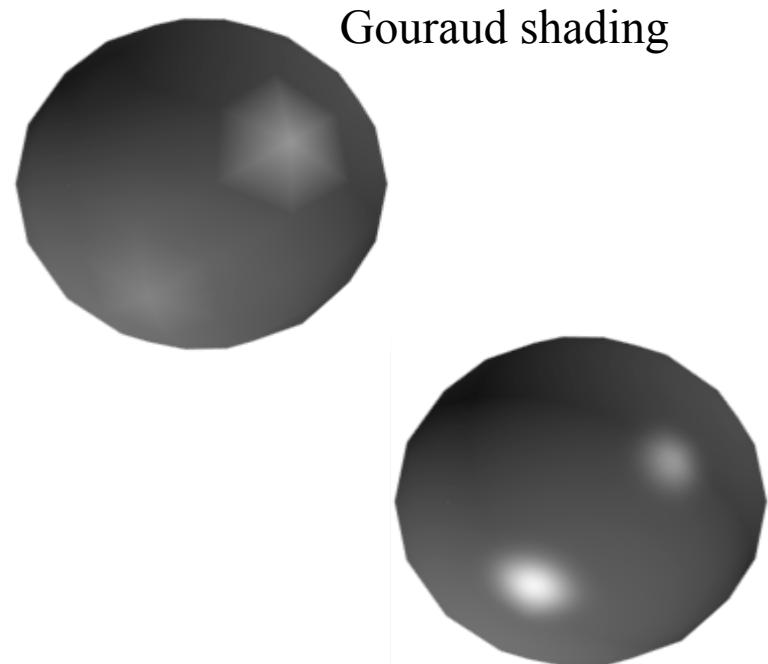
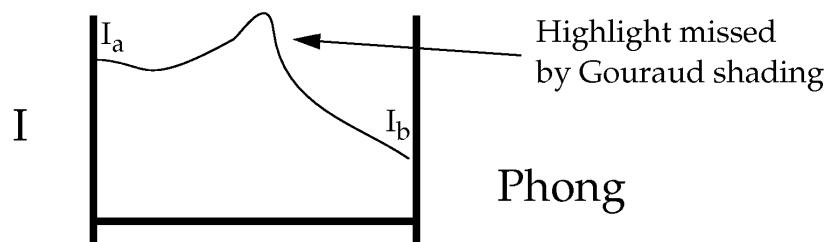
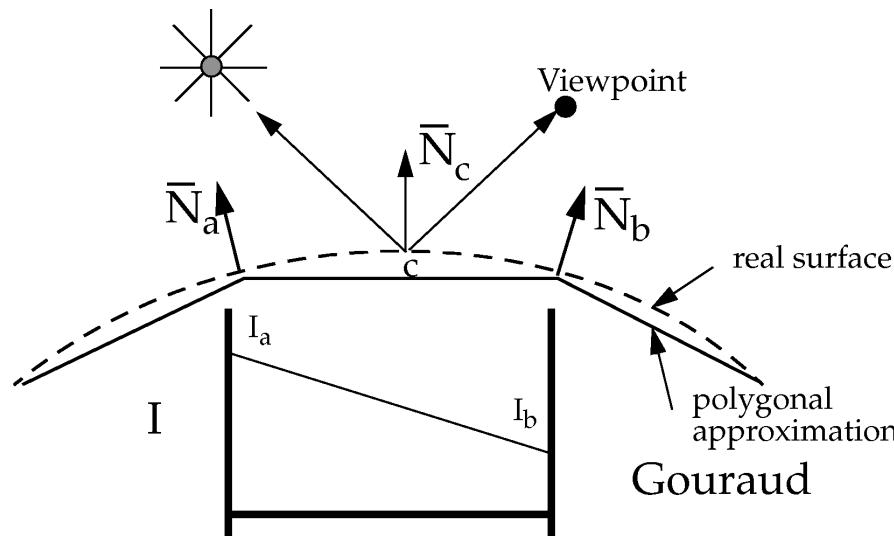
Esempio: Gouraud Surface Rendering

Figura a) e c): stesso numero di poligoni!!



Problemi con Gouraud Shading

- Lo shading di Gouraud fallisce nei casi in cui ho, o dovrei avere, un grado di illuminazione *intenso*
- In particolare lo shading di Gouraud ha un problema quando è *significativa* la riflessione speculare



“altro” shading ... qual’è? =>

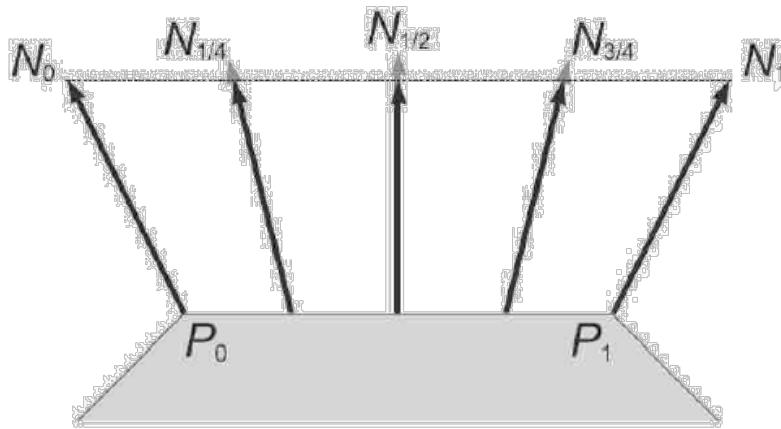
Gouraud Shading: Pro e Contro

- ❖ Gouraud shading: ottimale rapporto qualità/prezzo
- ❖ Risultati non eccezionali per superfici dotate di un alto coefficiente di riflessione speculare
- ❖ Problema: per n alto lo *specular highlight* deve essere piccolo, invece si “propaga” per tutta la faccia (per interpolazione) se cade vicino a un vertice, si “perde” se è interno

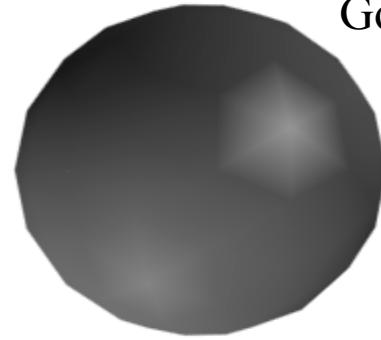
Il si “perde” lo abbiamo visto nella slide precedente ... il si “propaga” che vuol dire?

Phong Shading

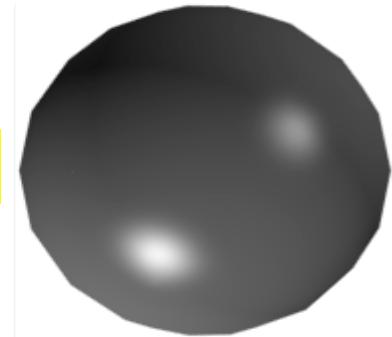
- ❖ Soluzione: si interpola nello spazio delle normali e si calcola l'equazione di illuminazione in ogni pixel



Phong shading!!!!



Gouraud shading



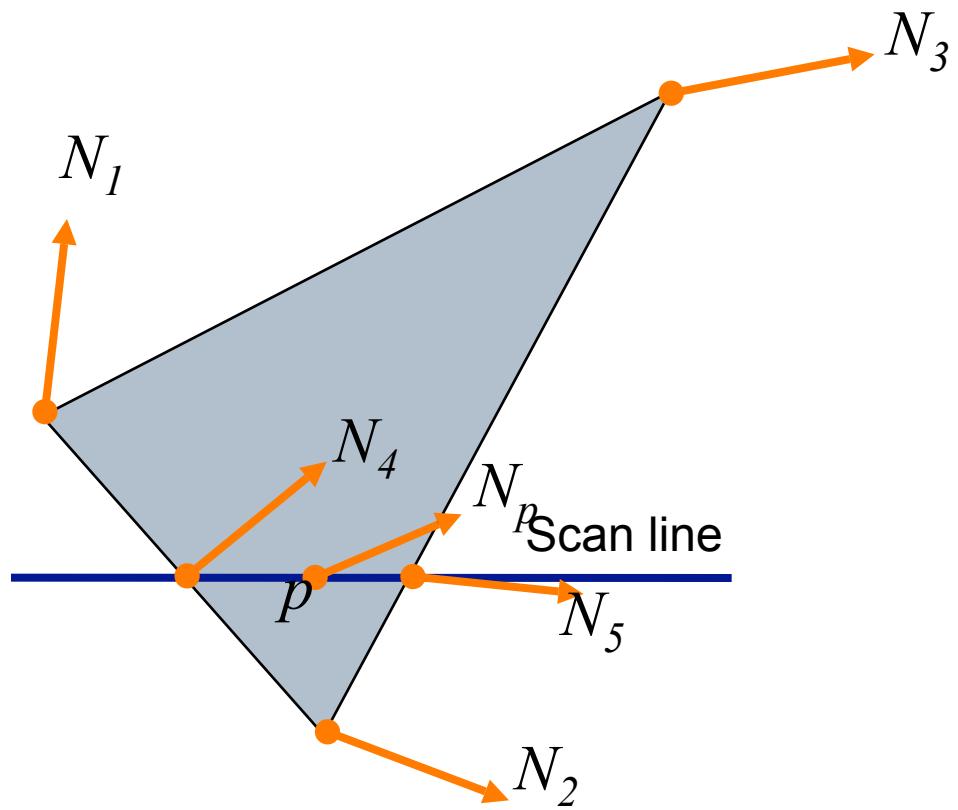
□ **Phong shading (o normal-vector interpolation rendering)**: è un metodo di interpolazione lineare delle normali lungo la superficie di una faccia, con il fine di applicare il *Phong lighting model* in ciascun punto con una normale "piu' adeguata".

Attenzione: il "Phong Shading" non è il modello di illuminazione di Phong – spesso sono confusi.

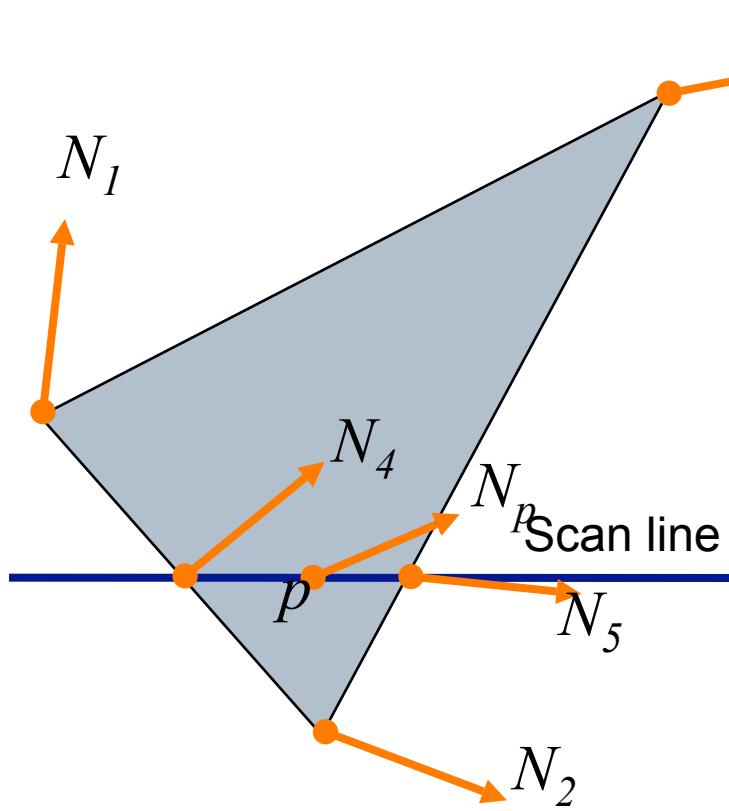
Phong Surface Rendering (cont...)

- Nel rendering di un poligono, il Phong surface rendering procede come segue:
 1. Determina il vettore normale unitario medio per ciascun vertice del poligono
 2. Interpola linearmente le normali nei vertici lungo l'area *proiettata* del poligono
 3. Applica un modello di illuminazione ai punti (pixel) lungo le scan line usando i vettori normali ottenuti dall'interpolazione

Phong Surface Rendering (cont...)



Phong Surface Rendering (cont...)



$$N_4 = \frac{y_4 - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y_4}{y_1 - y_2} N_2$$

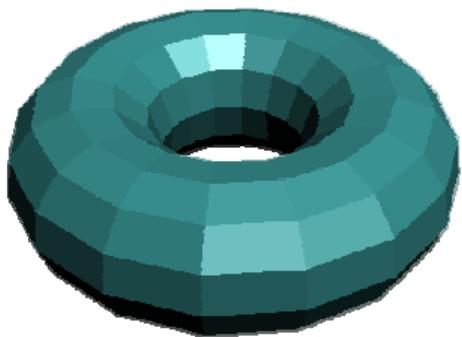
$$N_5 = \frac{y_5 - y_2}{y_3 - y_2} N_3 + \frac{y_3 - y_5}{y_3 - y_2} N_2$$

$$N_p = \frac{y_p - y_5}{y_4 - y_5} N_4 + \frac{y_4 - y_p}{y_4 - y_5} N_5$$

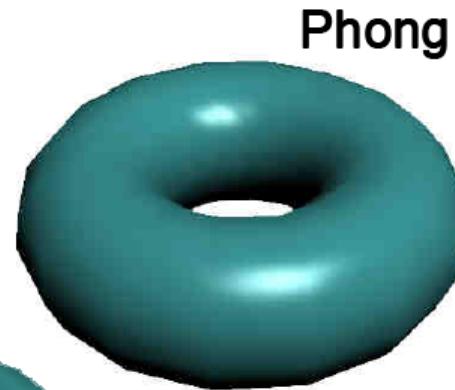
Phong Surface Rendering Implementation

- Il Phong shading è molto più lento del Gouraud shading, dato che il *lighting model* è valutato per ogni punto del poligono
- Tuttavia, esistono approcci ottimizzati del Phong surface rendering che lo rendono meno “oneroso”
- Anche il Phong shading è implementato all’interno del processo di *visible surface detection*

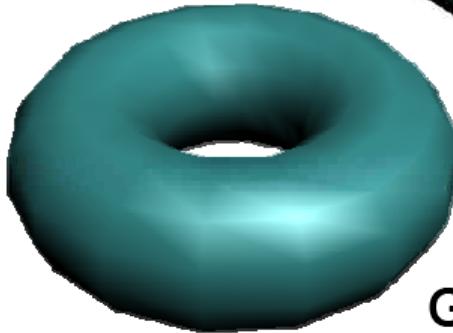
Confronto: Flat, Gouraud e Phong Shading



Costante



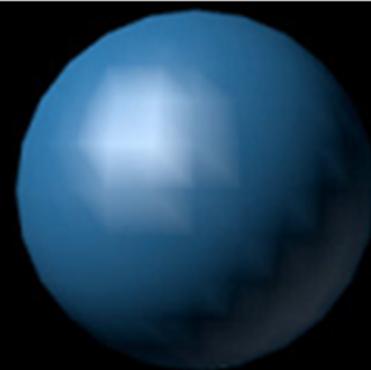
Phong



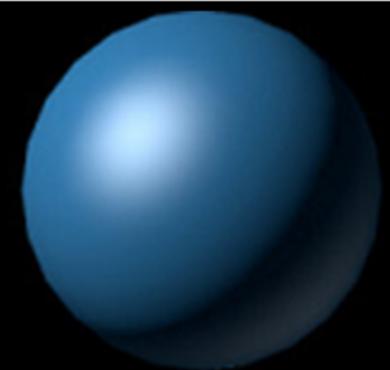
Gouraud



Flat shading



Gouraud shading

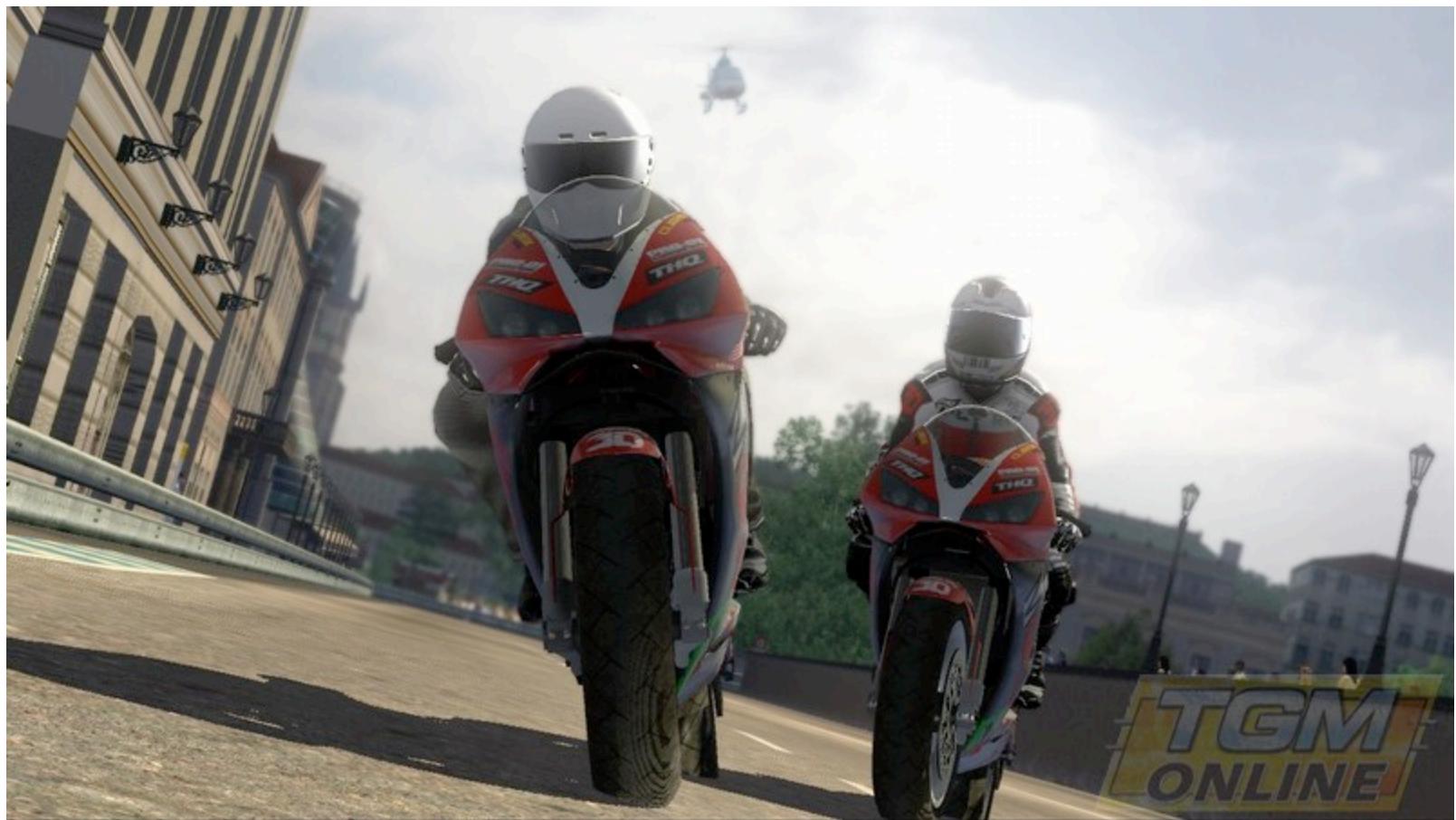


Phong Shading

Esempio di Phong Shading



Esempio di Phong Shading



In conclusione ...

- Per un rendering realistico delle superfici approssimate da poligoni abbiamo bisogno di un metodo per determinare l'illuminazione della scena
- Flat shading ... veloce, ma non realistico
- Gouraud shading ... meglio, ma non gestisce adeguatamente la *specular reflections*
- Phong shading ... ancora meglio, ma può essere “*lento*”

Shading in OpenGL

```
glBegin(GL_TRIANGLES);  
    glNormal3fv( n );  
    glVertex3fv( v0 );  
    glVertex3fv( v1 );  
    glVertex3fv( v2 );  
glEnd();
```

flat shading !

```
glBegin(GL_TRIANGLES);  
    glNormal3fv( n0 );  
    glVertex3fv( v0 );  
    glNormal3fv( n1 );  
    glVertex3fv( v1 );  
    glNormal3fv( n2 );  
    glVertex3fv( v2 );  
glEnd();
```

Gouraud shading

Shading in OpenGL

- Scorcianoia:
 - se setto:

Prendo solo una normale
di faccia che vale per tutti i
vertici del poligono

```
glShadeModel(GL_FLAT);
```

gli attributi non vengono interpolati
ma rimangono costanti nella faccia
(utile per le triangle strip e i triangle fan),
finchè non rimetto

```
glShadeModel(GL_SMOOTH);
```

Default!