

Marco Tarini - Computer Graphics 2023/2024  
Università degli Studi di Milano

## Lighting 2/3: Lambert e Phong



20

### Materiali non uniformi

- ✓ Una superficie può presentare una BRDF potenzialmente diversa in ogni suo punto  $\mathbf{p}$ 
  - ⇒ Si parla allora di materiale non uniforme, o “spatially varying” (che varia nello spazio)
  - ⇒ In questo caso, la funzione BRDF prende in input anche il punto  $\mathbf{p}$  :

$$f_r(\mathbf{p}, \vec{\omega}_i, \vec{\omega}_r)$$

=


“quanta della luce che raggiunge il punto  $\mathbf{p}$  dalla direzione  $\vec{\omega}_i$  viene riflessa nella direzione  $\vec{\omega}_r$ ”

- ✓ Altrimenti, si parla di materiale *uniforme*




21

### Materiali non uniformi



Mesh con materiale uniforme

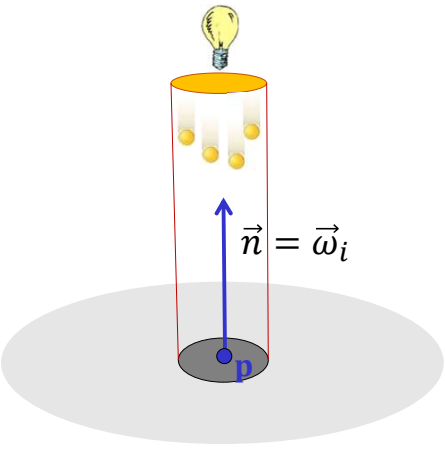
Mesh con materiale non uniforme  
(la variazione di materiale è codificata in una tessitura)




22

### Sottoproblema

✓ Se dalla dir  $\vec{\omega}_i$  arrivano  $L$  lumens ,  
quante ne riceve un intorno di  $\mathbf{p}$  con normale  $\vec{n}$  ?



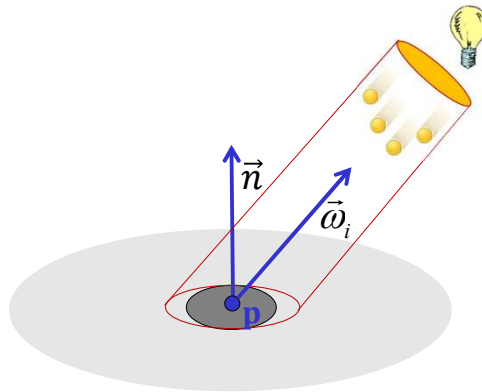
$\vec{n} = \vec{\omega}_i$



23

### Sottoproblema

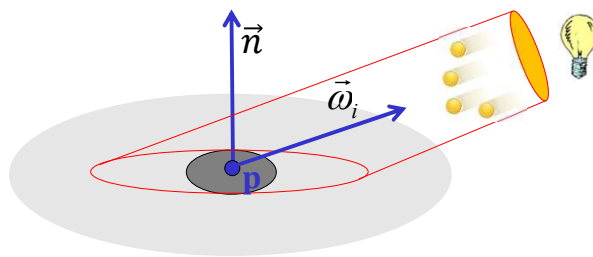
- ✓ Se dalla dir  $\vec{\omega}_i$  arrivano  $L$  lumens ,  
quante ne riceve un intorno di  $\mathbf{p}$  con normale  $\vec{n}_p$  ?



24

### Sottoproblema

- ✓ Se dalla dir  $\vec{\omega}_i$  arrivano  $L$  lumens ,  
quante ne riceve un intorno di  $\mathbf{p}$  con normale  $\vec{n}$  ?

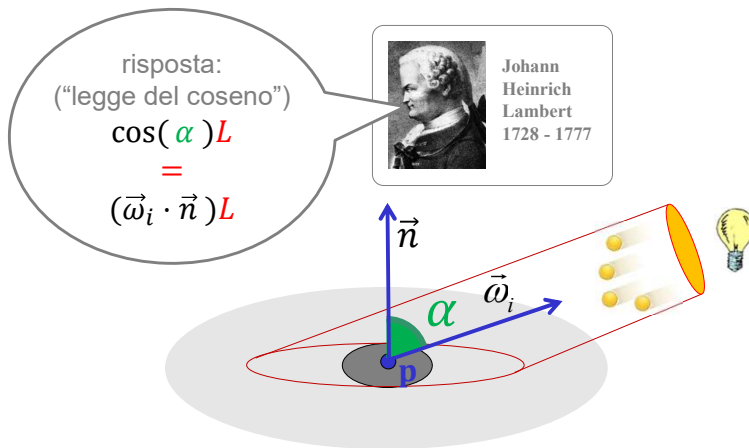


25

### Sottoproblema

✓ Se dalla dir  $\vec{\omega}_i$  arrivano  $L$  lumens ,  
quante ne riceve un intorno di  $\mathbf{p}$  con normale  $\vec{n}$  ?

risposta:  
("legge del coseno")  
 $\cos(\alpha)L$   
=  
 $(\vec{\omega}_i \cdot \vec{n})L$

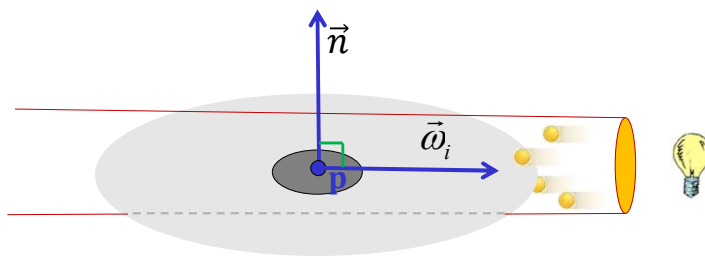


27

### Sottoproblema

✓ Se dalla dir  $\vec{\omega}_i$  arrivano  $L$  lumens ,  
quante ne riceve un intorno di  $\mathbf{p}$  con normale  $\vec{n}$  ?

Caso luce perfettamente radente:  
 $(\vec{\omega}_i \cdot \vec{n}) = 0$   
infatti nessun fotone colpisce l'intorno

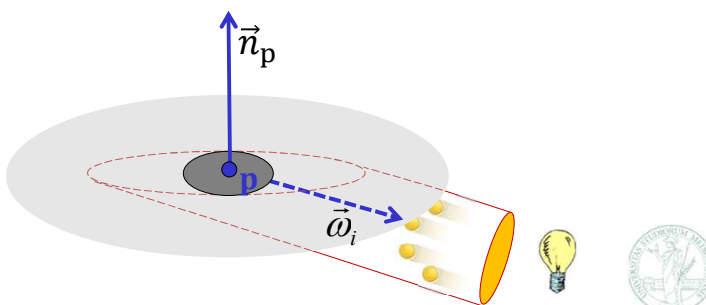


29

## Sottoproblema

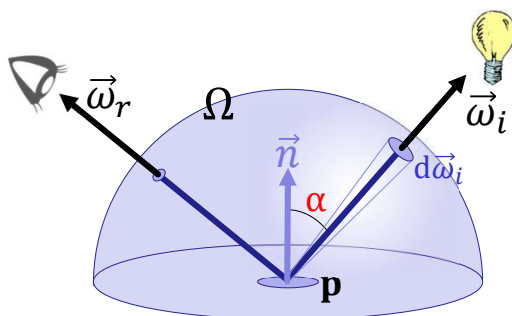
✓ Se dalla dir  $\vec{\omega}_i$  arrivano  $L$  lumens ,  
 quante ne riceve un intorno di  $\mathbf{p}$  con normale  $\vec{n}$  ?

Caso luce da dietro:  
 $(\vec{\omega}_i \cdot \vec{n}_p) < 0$   
 allora la risposta è 0  
 (non certo un numero negativo)  
 perché l'intorno si trova nella sua stessa ombra



30

## Lighting locale in una forma molto generale: l'equazione della radianza



$$L_o(\mathbf{p}, \vec{\omega}_r) = L_e(\mathbf{p}, \vec{\omega}_r) + L_r(\mathbf{p}, \vec{\omega}_r)$$

$$L_r(\mathbf{p}, \vec{\omega}_r) = \int_{\vec{\omega}_i \in \Omega} f_r(\mathbf{p}, \vec{\omega}_i, \vec{\omega}_r) L_i(x, \vec{\omega}_i) \underbrace{(\vec{\omega}_i \cdot \vec{n})}_{\cos(\alpha)} d\vec{\omega}_i$$

31

### Lighting locale in una forma molto generale: l'equazione della radianza

luce osservata guardando p dalla direz.  $\vec{\omega}_r$ 
luce emessa da p in direz.  $\vec{\omega}_r$ 
luce riflessa da p in direz.  $\vec{\omega}_r$

$$L_o(\mathbf{p}, \vec{\omega}_r) = L_e(\mathbf{p}, \vec{\omega}_r) + L_r(\mathbf{p}, \vec{\omega}_r)$$

$$L_r(\mathbf{p}, \vec{\omega}_r) = \int_{\vec{\omega}_i \in \Omega} f_r(\mathbf{p}, \vec{\omega}_i, \vec{\omega}_r) L_i(x, \vec{\omega}_i) \underbrace{(\vec{\omega}_i \cdot \vec{n})}_{\cos(\alpha)} d\vec{\omega}_i$$

32

### Luce emessa e luce riflessa

- ✓ La **luce emessa** è inclusa solo durante il rendering di quei rari oggetti che emettono luce propria
  - ⇒ Ad esempio, nel renderizzare una lampadina led, oppure la superficie il sole in un paesaggio
- ✓ Modella il *percorso* diretto della luce: **sorgente** → POV (invece che: **sorgente** → **superficie** → POV)
  - ⇒ Nota: affinché questa sorgente di luce illumini gli *altri* oggetti, è necessario includerla nella nel fattore  $L_i(x, \vec{\omega}_i)$  usato nel calcolo della **luce riflessa** durante il tutti i rendering di questi altri oggetti
- ✓ Questa componente può venire calcolata come una semplice costante, da aggiungere alla luce riflessa
  - ⇒ Oppure tre costanti, una per ciascuno dei tre canali RGB
  - ⇒ Questa è detta «componente emissiva» del modelli di lighting (per es, da `three.js`)
- ✓ Concentriamoci sulla luce riflessa

33

## Lighting locale in una forma molto generale: l'equazione della radianza [ simboli e spiegazione ]

$\mathbf{p}$  : punto sulla superficie da illuminare  
 $\vec{n}$  : normale della superficie in  $\mathbf{p}$   
 $\Omega$  : insieme di tutte le direzioni possibili di provenienza della luce  
⇒ una semisfera unitaria davanti a  $\mathbf{p}$   
 $\vec{\omega}_r$  : la direzione verso l'osservatore  
⇒ va da  $\mathbf{p}$  verso il POV  
 $\vec{\omega}_i$  : una possibile direzione di provenienza della luce  $\in \Omega$   
⇒ va da  $\mathbf{p}$  verso l'esterno  
 $d\vec{\omega}_i$  : un intorno di  $\vec{\omega}_i$  (angolo solido)  
 $L_i(\mathbf{p}, \vec{\omega}_i)$  : intensità della luce che raggiunge  $\mathbf{p}$  da una data direzione  $\vec{\omega}_i$  .  
Descrive l'**ambiente di illuminazione**  
Espressa in quanta luce per angolo solido. Quindi...  
 $L_i(\mathbf{p}, \vec{\omega}_i) d\vec{\omega}_i$  : quanta luce arriva verso  $\mathbf{p}$  dall'intorno di  $\vec{\omega}_i$   
 $(\vec{\omega}_i \cdot \vec{n})$  : di questa, quanta viene ricevuta dall'intorno di  $\mathbf{p}$  (**legge del coseno**)  
 $f_r(\mathbf{p}, \vec{\omega}_i, \vec{\omega}_r)$  : di questa, quanta viene riflessa proprio della direzione  $\vec{\omega}_r$  .  
E' la BRDF del punto  $\mathbf{p}$  . Descrive il **materiale** (nel punto  $\mathbf{p}$ )

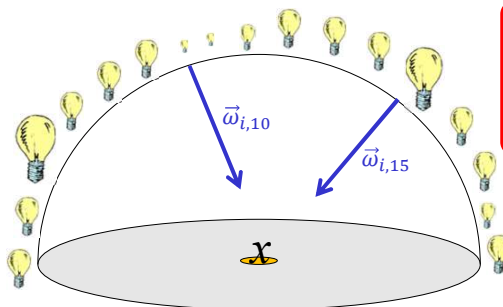
34

## Ambiente di illuminazione: luce incidente

✓ Per ogni posizione  $\mathbf{p}$   
 $L_i(\mathbf{p}, \vec{\omega}_i)$  modella  
la distribuz di luce incidente

$L_i(\mathbf{p}, \vec{\omega}_i)$  = quanta luce arriva nel punt  $\mathbf{p}$  dalla direzione  $\vec{\omega}_i$

Modella un *ambiente di illuminazione*.  
Esempio di ambienti di illuminazione:  
• stanza con finestra aperta  
• giornata di sole  
• giornata coperta  
• una discoteca



Cioè (nella metafora dei lucidi precedenti): quante "palle da tennis" sono tirate verso  $\mathbf{p}$  da ciascuna direzione!

35

## Alcune assunzioni semplificatrici

- ✓ L'equazione della radianza vista è troppo onerosa per il computo del lighting in un real-time rendering
  - ⇒ Può essere impiegata nel lighting nei rendering offline
- ✓ Operiamo ora una sequenza di assunzioni semplificatrici, fino a ridurla ad una semplice espressione che può essere computata in modo pratico e veloce (per ogni frammento di un rendering in real time)
  - ⇒ Otterremo così un "modello di illuminazione" semplice ed efficiente
  - ⇒ In seguito, raffineremo alcune delle assunzioni per incrementare un po' la qualità del rendering (avvicinandosi un po' all'equazione di radianza generale) senza aggravare troppo la complessità del computo



36

## Alcune assunzioni semplificatrici

- ✓ "Materiale uniforme" (l'intero oggetto esibisce la stesso BRDF). La posizione  $\mathbf{p}$  non conta più:

$$L_r(\mathbf{p}, \vec{\omega}_r) = \int_{\vec{\omega}_i \in \Omega} f_r(\mathbf{p}, \vec{\omega}_i, \vec{\omega}_r) L_i(\mathbf{p}, \vec{\omega}_i) (\vec{\omega}_i \cdot \vec{n}) d\vec{\omega}_i$$

- ✓ La BRDF è puramente Lambertiana (o diffusiva) (vale una costante  $D$ , l'albedo). Allora

$$L_r(\mathbf{p}, \vec{\omega}_r) = \int_{\vec{\omega}_i \in \Omega} D L_i(\mathbf{p}, \vec{\omega}_i) (\vec{\omega}_i \cdot \vec{n}) d\vec{\omega}_i$$

- ✓ La luce proviene solo da una direzione costante  $\vec{\omega}_i$  (un parametro; per esempio, verso il sole), con intensità  $L$ . Allora

$$L_r(\mathbf{p}, \vec{\omega}_r) = D L (\vec{\omega}_i \cdot \vec{n})$$



37



### Equazione di lighting per materiali lambertiani (o puramente diffusivi)

✓ L'equazione di lighting risultante per una luce è

Prodotto dot,  
ma zero se negativo

Luminosità  
finale  
del pixel


$P = (\vec{n} \cdot \vec{\omega}_i) D L$

normale della sup

Una sola luce (di  
intensità  $L$ )  
investe la scena  
dalla direzione  $\vec{\omega}_i$   
(vettore unitario che  
va verso la luce)

Intensità della luce

albedo del  
materiale  
(quanto è chiaro o  
scuro il suo  
aspetto)



38

### Lighting lambertiano... a colori

✓ L'equazione di lighting vista sopra sarebbe utilizzabile per immagini in bianco e nero

✓ Per passare ad immagini a colori, un modo semplice è usare la stessa equazione separatamente per i canali Rosso, Verde, e Blu

⇒ Questa è un'approssimazione brutale, ma di utilizzo molto comune

canale  
rosso  
del pixel  
risultante

$P_R = (\vec{n} \cdot \vec{\omega}_i) D_R L_R$

$P_G = (\vec{n} \cdot \vec{\omega}_i) D_G L_G$


$P_B = (\vec{n} \cdot \vec{\omega}_i) D_B L_B$

"albedo", (per così dire)  
ma solo per quel che  
riguarda la luce rossa

Intensità della  
luce sul canale rosso  
(quanta luce *rossa*  
viene emessa dalla  
direzione  $\vec{\omega}_i$ )

Idem, per il verde

Idem, per il blue



39

### Lighting lambertiano... a colori (una riscrittura)

Scalatura del vettore RGB (nessun simbolo)


Questo simbolo donota qui il "prodotto componente per componente"

$$\begin{pmatrix} P_R \\ P_G \\ P_B \end{pmatrix} = (\vec{n} \cdot \vec{\omega}_i) \begin{pmatrix} D_R \\ D_G \\ D_B \end{pmatrix} \otimes \begin{pmatrix} L_R \\ L_G \\ L_B \end{pmatrix}$$

pixel finale

**Diffuse color**  
o **Lambertian color**  
o **Base color**.  
In pratica, la risposta alla domanda "di che colore è l'oggetto" (a prescindere dalla luce usata per illuminarlo)


Intensità della luce, su ciascun canale (in sostanza: il colore e l'intensità della luce)



40

### Interpretazione intuitiva del modello di Lambert

- ✓ Anche se il modello di materiale diffusivo (o di Lambert) è basato sulla fisica reale, possiamo anche darne una interpretazione geometrica intuitiva
- ✓ Il fattore  $(\vec{n}_p \cdot \vec{\omega}_i)$ , il coseno dell'angolo fra i due vettori, è anche una misura della similarità fra la direzione  $\vec{n}_p$  normale alla superficie e la direzione  $\vec{\omega}_i$  «verso la luce»
- ✓ Quindi, la legge del coseno di Lambert dice: «tanto più la superficie è orientata verso la luce (cioè, tanto più la sua normale è simile alla direzione di provenienza della luce), maggiormente chiara ci apparirà.»



41

## Lighting con più sorgenti di luce

- ✓ Abbiamo supposto di avere una sola sorgente di luce  
 ⇒ di direzione  $\vec{\omega}_i$  e intensità  $(L_R, L_G, L_B)$
- ✓ Passiamo a  $n$  sorgenti di luce:  
 ogni luce  $k$ -esima avrà la propria direzione  $\vec{\omega}_{ik}$  e intensità  $(L_{Rk}, L_{Gk}, L_{Bk})$   
 ⇒ Maggiore il numero di sorgenti, più complesso l'ambiente di illuminazione, ma più caro il rendering
- ✓ Il lighting è additivo: si sommano i contributi di ciascuna luce
- ✓ Esempio con due luci:

$$\begin{pmatrix} P_R \\ P_G \\ P_B \end{pmatrix} = (\vec{n} \cdot \vec{\omega}_{i1}) \begin{pmatrix} D_R \\ D_G \\ D_B \end{pmatrix} \otimes \begin{pmatrix} L_{R1} \\ L_{G1} \\ L_{B1} \end{pmatrix} + (\vec{n} \cdot \vec{\omega}_{i2}) \begin{pmatrix} D_R \\ D_G \\ D_B \end{pmatrix} \otimes \begin{pmatrix} L_{R2} \\ L_{G2} \\ L_{B2} \end{pmatrix}$$

Base color

pixel finale      Direzione luce 1      Intensità / colore della luce 1      Direzione luce 2      Intensità / colore della luce 2

42

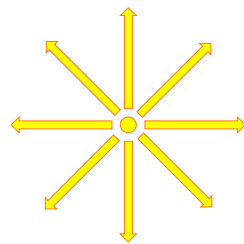
## Lighting con più sorgenti di luce: osservazioni

- ✓ L'aggiunta di ogni luce comporta un aggravio di computazione del lighting  
 ⇒ Dobbiamo computare un altro termine, su ogni frammento della scena
- ✓ In assenza di qualsiasi fonte di luce, il lighting degli oggetti ha zero addendi, e il colore risultante è  $(0,0,0)$   
 ⇒ Com'è sensato che sia:  
 al buio, tutti gli oggetti appaiono neri!
- ✓ All'aggiungere fonti di luce, il lighting si satura verso il bianco, dato che il pixel finale al massimo vale  $(1,1,1)$   
 ⇒ Quindi se vogliamo aggiungere molte luci, ciascuna di loro deve essere fiavole

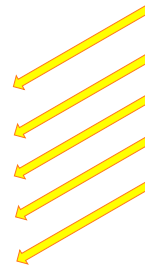
43

## Modellazione della sorgente di luce

- ✓ Possiamo considerare due tipi di sorgenti di luce:
  - ⇒ "direzionali" : modella fonti di luce grandi e molto distanti, per es, il sole (il tipo considerato fino a questo momento)
  - ⇒ "posizionali" : modella fonti di luci vicine e vicine, per es, lampadine
- ✓ Una luce direzionale rappresenta una luce posizionale posta molto lontana (arriva dalla stessa direzione in ogni punto della scena)



LUCE POSIZIONALE



LUCE DIREZIONALE



44

## Luci posizionali e direzionali

- ✓ Per la luce **direzionale** , la direzione di luce incidente  $\vec{\omega}_i$  è una costante, e non varia nella scena
  - ⇒ Il vettore unitario  $\vec{\omega}_i$  è un parametro settato per ogni luce
- ✓ Una luce **posizionale** è invece descritta da una posizione nello spazio  $\mathbf{p}_L$ 
  - ⇒ la direzione di luce incidente  $\vec{\omega}_i$  sarà diversa per ogni punto  $\mathbf{p}$  da illuminare, e sarà data da:

$$\vec{\omega}_i = \frac{\mathbf{p}_L - \mathbf{p}}{\|\mathbf{p}_L - \mathbf{p}\|}$$



45

## Luci posizionali: affievolimento con la distanza (in inglese: falloff oppure decay)

- ✓ Una luce **direzionale** ha un'intensità/colore  $(L_R, L_G, L_B)$  costante su tutta la scena
- ✓ In una luce **posizionale**, maggiore è la distanza dalla fonte di luce, minore l'intensità della luce ricevuta.
- ✓ l'intensità della luce può essere attenuata, per ogni punto illuminato  $\mathbf{p}$ , da un fattore di affievolimento calcolato in funzione della sua distanza dalla luce  $d = \|\mathbf{p}_L - \mathbf{p}\|$

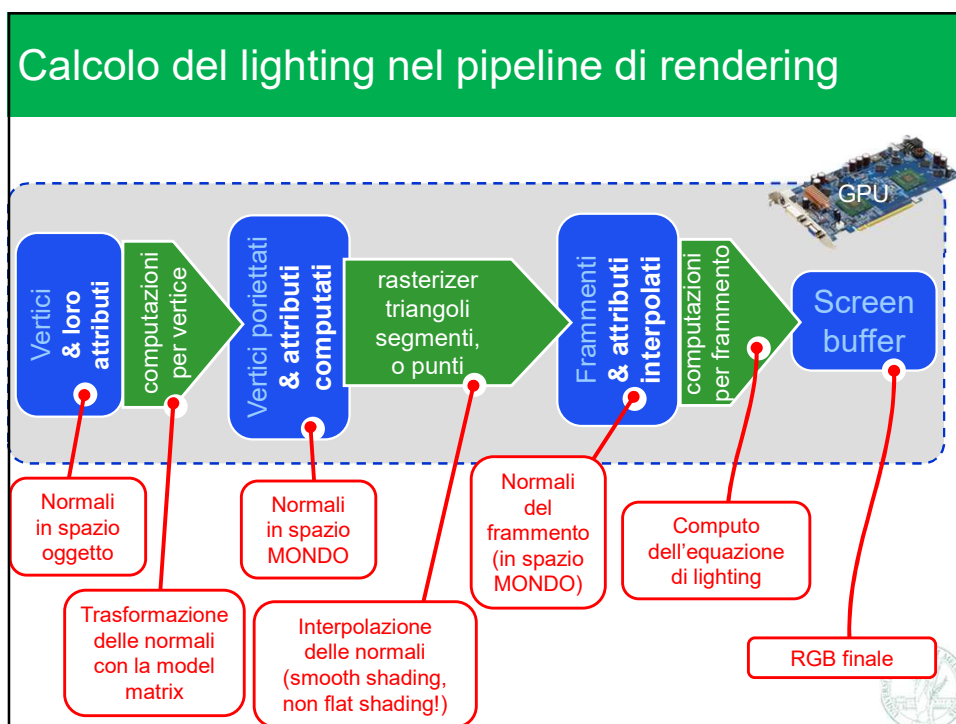
⇒ La formula fisica è

$$f_{\text{affievolimento}} = \frac{1}{d^2}$$

⇒ a volte si usa un esponente minore di 2, per es 1  
per rendere artificialmente meno decisa l'attenuazione

⇒ (se esponente scelto è 0, non c'è attenuazione – perché?)

46



47

## Modelli (o equazioni) di lighting

- ✓ L'equazione di lighting vista è molto semplice
  - ⇒ consiste esclusivamente di una componente di riflessione diffusa
  - ⇒ però è physically based – il fenomeno ottico modellato è accuratamente riprodotto (eccetto che per la tricromia RGB)
- ✓ E' un esempio di modello di illuminazione locale
- ✓ Il questo modello:
  - ⇒ il «materiale» è descritto unicamente dal base color (detto anche diffuse color -- o albedo se in bianco e nero)
  - ⇒ L'unico aspetto geometrico che è rilevante è la normale del punto illuminato. In particolare, la direzione di osservazione non conta.
  - ⇒ Per questo, il modello è detto «view independent».
  - ⇒ Cioè: il chiaroscuro non dipende dalla direzione di osservatore: non abbiamo «riflessi» vividi (infatti materiali come gesso ne sono privi)
- ✓ Vedremo in seguito altri modelli, in grado di produrre «riflessi»
  - ⇒ In inglese: highlights, glossy reflections,



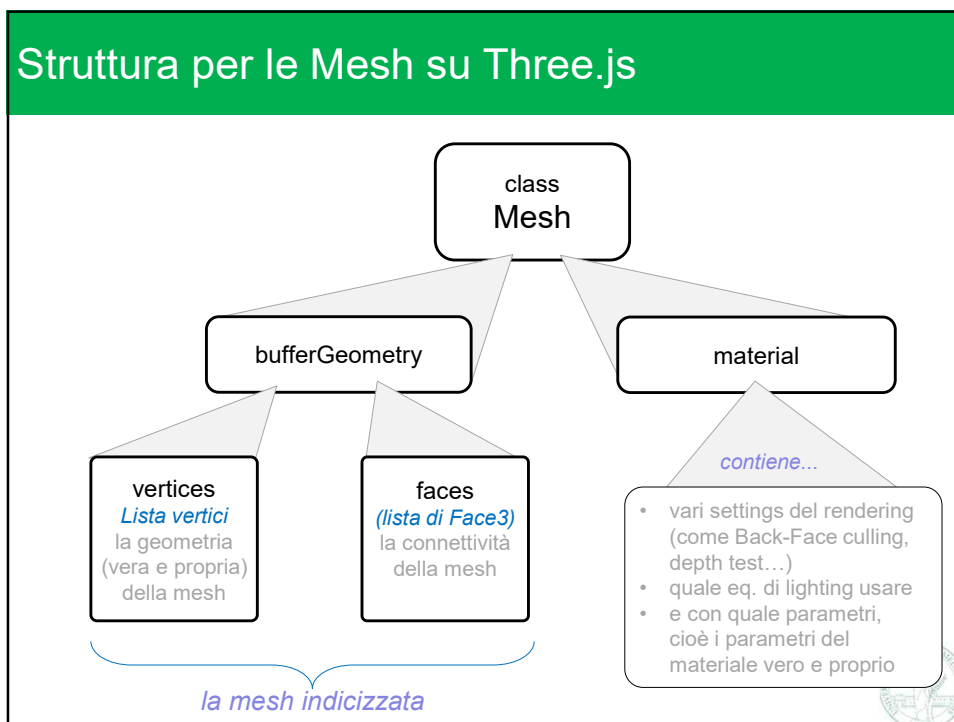
48

## Calcolo del lighting nel pipeline di rendering

- ✓ il computo del lighting tipicamente viene svolto nella fase per frammento
  - ⇒ Tecnica nota come “per-pixel lighting”, calcolo dell'illuminazione per ogni pixel
  - ⇒ (ma esistono anche varianti e ottimizzazioni)
  - ⇒ Dunque, il lighting avviene dopo la trasformazione dei vertici
  - ⇒ L'equazione di lighting è calcolata in un fragment-shader
  - ⇒ Usando three.js, stiamo usando un fragment-shader fornito dalla libreria
- ✓ I punti e vettori usati nell'equazione di lighting devono, ovviamente, essere tutti espressi nello stesso spazio
  - ⇒ direzione luce, direzione vista, normale, posizione luce...
  - ⇒ Ad esempio, si può scegliere lo spazio di vista: allora, le normali della mesh (originalmente espresse, ovviamente, in spazio oggetto) devono essere trasformate in spazio vista, nella fase di trasformazione per vertice, moltiplicandole per la matrice di model-view
  - ⇒ Anche lo spazio mondo può essere usato (quale matrice occorre usare, allora?)
  - ⇒ (nota: NON in spazio clip: la matrice di proiezione prospettica non può essere usata per trasformare vettori, ma solo punti)
  - ⇒ Le direzioni della luce vanno tutte espresse nello stesso spazio in cui è espressa la normale



49



50

### Sperimentiamo il lighting di materiali Lambertiani in three.js (note 1/2)

- ✓ 1: costruiamo il materiale e usiamolo per la mesh

```
var mioMat = new THREE.MeshLambertMaterial();
var miaMesh = new THREE.Mesh( buffers, mioMat );
```

(questo istruisce three.js ad usare il pipeline come descritto sopra)
- ✓ 2: settiamo il base color (detto anche diffuse color) del materiale ad esempio, ad un blu chiaro

```
mioMat.color.set(0xFF8855);
```
- ✓ 3: creiamo le luci e aggiungiamole alla scena ad esempio, una luce direzionale

```
var miaLuce = new THREE.DirectionalLight();
miaScena.add(miaLuce);
```
- ✓ 4: settiamo la direzione (chiamata da three.js, per confondere, "position") e l'intensità-colore della luce (chiamato da three.js color), ad esempio, una luce bianca (il default)

```
miaLuce.color.set(0xFFFFFFFF);
miaLuce.position.set(0,3,-2); // viene normalizzato
```

Vedere il progetto [lab05.html](#)

52

## Sperimentiamo il lighting di materiali Lambertiani in three.js (note 2/2)

- ✓ possiamo aggiungere una seconda, ad esempio, questa volta una luce posizionale

```
var luce2 = new THREE.PointLight();  
miaScena.add(luca2);
```

- ✓ settiamone la posizione (per es, x=2,y=1.5,z=5) e l'intensità-colore della luce (chiamato color), ad esempio, una luce verdognola

```
luca2.color.set(0x66DD66);  
luca2.position.set(2,1.5,5);
```

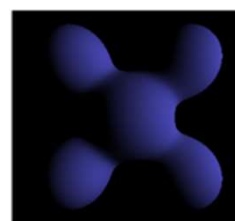
Vedere il progetto [lab06.html](#)



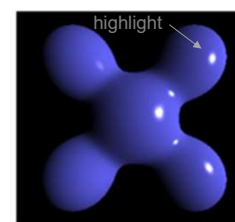
53

## Limiti del modello di Lambert

- ✓ Un materiale Lambertiano riproduce solo l'aspetto materiali opachi (dull) privi di riflessi luminosi
  - ⇒ ma, almeno, lo fa in modo fisicamente corretto
- ✓ Di seguito, vediamo il modello di lighting detto di Phong, capace di riprodurre anche l'aspetto di superfici lucide
  - ⇒ come quelle bagnate, levigate, incerate, etc,
  - ⇒ aggiungendo al lighting labertiano riflessi lucidi (detti highlight, specular reflections, o glossy reflections)
- ✓ Il modello di Phong, tuttavia...
  - ⇒ non è ispirato da considerazioni fisiche
  - ⇒ non è confermato da misurazione radiometriche.
  - ⇒ è basato su una costruzione geometrica molto approssimata (ma efficiente)



Lambert



Phong

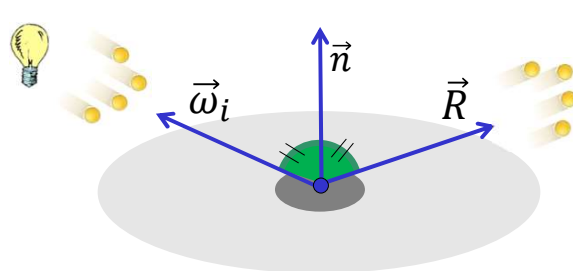


54



### Modello di illuminazione di Phong: la spiegazione spaziale intuitiva

- ✓ In presenza di un materiale molto levigato / liscio / lucido, i fotoni tenderanno a rimbalzare sulla superficie (riflessione) in modo simile a quello di una pallina da ping-pong su un tavolo



Nota:  
I tre vettori  
mostrati  
sono  
co-planari

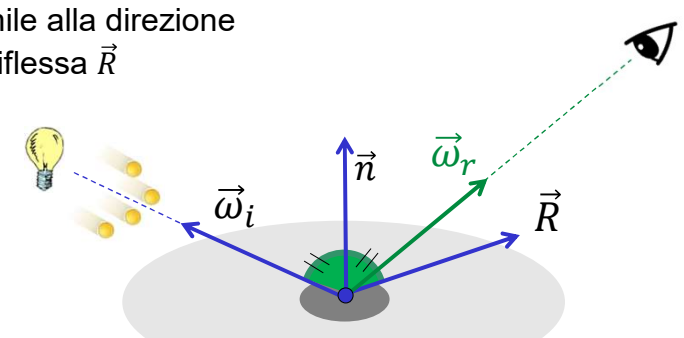
- ✓ Intuizione: la BRDF di un materiale lucido manderà un numero maggiore dei fotoni provenienti dalla direzione  $\vec{\omega}_i$  nella direzione riflessa  $\vec{R}$ , che non nelle altre direzioni  
⇒ a differenza di quella di un materiale diffusivo / lambertiano, che li rimbalza in qualsiasi direzione con uguale probabilità



55

### Modello di illuminazione di Phong: componente speculare: spiegazione intuitiva

- ✓ Il modello di Phong somma alla componente di riflesso **diffusiva** (quella già discussa) una componente di riflesso **speculare**, che sarà tanto maggiore tanto più la direzione di vista  $\vec{\omega}_r$  sarà simile alla direzione di luce riflessa  $\vec{R}$



56

### Modello di illuminazione di Phong: componente speculare: un calcolo preliminare

calcolo di  $\vec{R}$  a partire da  $\vec{\omega}_i$  e  $\vec{n}$  (vedi lez algebra punti e vettori)

$$\vec{R} = -\vec{\omega}_i + 2 (\vec{\omega}_i \cdot \vec{n}) \vec{n}$$

57

### Modello di illuminazione di Phong: componente speculare: calcolo

Prodotto dot, ma 0 se negative. Misura la similarità fra le due direzioni!

“Specular exponent” o “shininess” (vedi lucido successivo)

Intensità della luce (RGB)

$$(\vec{\omega}_r \cdot \vec{R})^s \begin{pmatrix} H_R \\ H_G \\ H_B \end{pmatrix} \otimes \begin{pmatrix} L_R \\ L_G \\ L_B \end{pmatrix}$$

intensità e colore RGB degli highlights, scelto a piacere per un dato materiale

58

## Modello di illuminazione di Phong: shininess (o specular exponent)

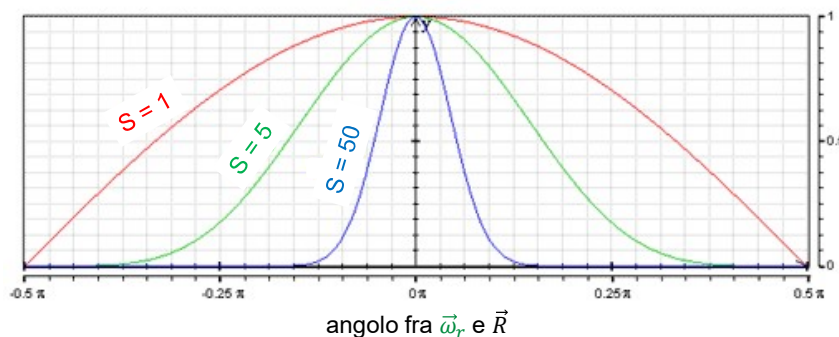
- ✓ La formula  $(\vec{\omega}_r \cdot \vec{R})^S$  (fra 1 e 0) produrrebbe riflessi troppo grandi, perché decresce troppo lentamente al discostarsi di  $\vec{\omega}_r$  da  $\vec{R}$
- ✓ Per ovviare, basta elevare ad una potenza  $S > 1$ 
  - ⇒ Elevando un numero minore di 1 ad un a  $S > 1$ , lo si avvicina allo 0
- ✓ Il fattore  $S$  è detto specular exponent, o shininess, o glossiness.
  - ⇒ Possiamo scegliere ad esempio valori come 1, 10, 100, o 1000
- ✓ Tanto maggiore vale  $S$ , tanto più simile dovranno essere  $\vec{\omega}_r$  e  $\vec{R}$  per avere un riflesso visibile, cioè tanto più piccoli e concentrati saranno gli *highlights*.
  - ⇒ Nuovamente, questo passaggio non ha nessun preciso significato fisico, ed è solo un espediente matematico usato per ottenere l'effetto desiderato (in modo controllabile tramite il parametro  $S$ )



59

## Componente *riflessione speculare*


- ✓ Elevando il coefficiente ad una potenza  $S > 1$ , si ottengono highlights più piccoli e concentrati



60

### Effetto della scelta dello specular exponent


$S = 2$        $S = 5$        $S = 10$        $S = 100$



61

### Calcolo della componente speculare con Blinn-Phong 1/2

- ✓ Blinn-Phong è una variante (quasi equivalente) per calcolare la similarità fra  $\vec{\omega}_r$  e  $\vec{R}$  senza dover calcolare esplicitamente  $\vec{R}$
- ✓ Intuizione:  $\vec{\omega}_r$  coincide con  $\vec{R}$  **sse** la normale  $\vec{n}$  è esattamente a metà strada dei due vettori  $\vec{\omega}_r$  e  $\vec{\omega}_i$  (verifica in un disegno!)



62

### Calcolo della componente speculare con Blinn-Phong 2/2

- ✓ Invece che calcolare  $\vec{R}$  (a partire da  $\vec{\omega}_i$  e  $\vec{n}$ ) e poi chiedersi quanto sia simile a  $\vec{\omega}_r$ , la formula Blinn-Phong calcola la direzione media fra  $\vec{\omega}_i$  e  $\vec{\omega}_r$ , detta half-way vector, e si chiede quanto sia simile ad  $\vec{n}$
- ✓ Questo produce un risultato molto simile ma meno oneroso calcolare

63

### Modello di illuminazione di Blinn-Phong: componente speculare: calcolo

Prodotto dot, ma 0 se negativo. Misura la similarità fra i due vettori

**Half-way vector:**  
la media (rinormalizzata) fra direzione di vista e direzione di luce

Intensità della luce (RGB)

$$(\vec{n} \cdot \vec{\omega}_h)^s \begin{pmatrix} H_R \\ H_G \\ H_B \end{pmatrix} \otimes \begin{pmatrix} L_R \\ L_G \\ L_B \end{pmatrix}$$

intensità e colore RGB degli highlights, scelto a piacere per un dato materiale

Specular exponent o shininess (come in precedenza)

64

### Modello di illuminazione di Phong: in totale

✓ Il modello di illuminazione di Phong aggiunge, (per ogni luce) alla componente diffusiva (di Lambert) la componente Speculare (qui, calcolata con Blinn-Phong)

65

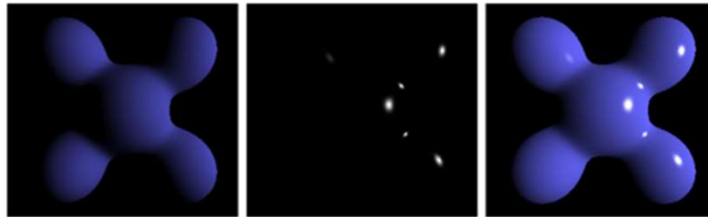
### Modello di illuminazione di Phong: riassunto degli input del calcolo

- ✓ la direzione di luce  $\vec{w}_i$  è un modello *view-dependent* !
- ✓ la direzione di vista  $\vec{w}_r$
- ✓ Intensità/colore della luce  $\begin{pmatrix} L_R \\ L_G \\ L_B \end{pmatrix}$
- ✓ Base-color o diffuse-color  $\begin{pmatrix} D_R \\ D_G \\ D_B \end{pmatrix}$  «di che colore è» l'oggetto
- ✓ Intensità/colore degli highlights  $\begin{pmatrix} H_R \\ H_G \\ H_B \end{pmatrix}$  Nuove caratteristiche del materiale scegliibili a piacere, che concorrono a differenziare un materiale da un altro
- «quanto sono intensi / di che colore sono i riflessi»
- ✓ Esponente di specularità, o shininess, o glossiness  $S$  «quanto sono piccoli i riflessi»

66

## Modello di Phong

Il risultato finale è ottenuto sommando al termine diffusivo (di Lambert) il nuovo componente speculare (gli highlight)



diffusivo + speculare = totale  
(highlight)



67

## Sperimentiamo il lighting di materiali Phong con three.js (note)

- ✓ 1: costruiamo il materiale di tipo Phong

```
var mioMat = new THREE.MeshPhongMaterial();  
var miaMesh = new THREE.Mesh( buffers, mioMat );
```
- ✓ 2: come prima, questo materiale prevede un base color base (o diffuse color). Ad esempio, scegliamo un azzurro chiaro

```
mioMat.color.set(0xFF8855);
```
- ✓ 3: il materiale di Phong prevede anche il colore speculare (colore degli highlights, chiamato da three.js "specular")

```
mioMat.specular.set(0x222222);
```

e un coefficiente speculare, chiamato da three.js "shininess"

```
mioMat.shininess = 100;
```

Vedere il progetto [lab06.html](#)



68