

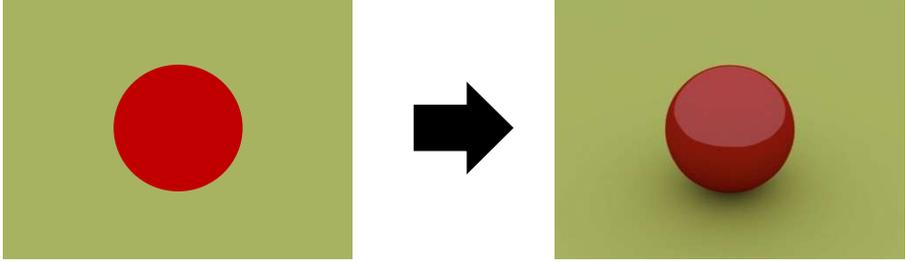
Marco Tarini - Computer Graphics 2024/2025
Università degli Studi di Milano

Lighting 1/2: intro e Lambertian



1

Lighting



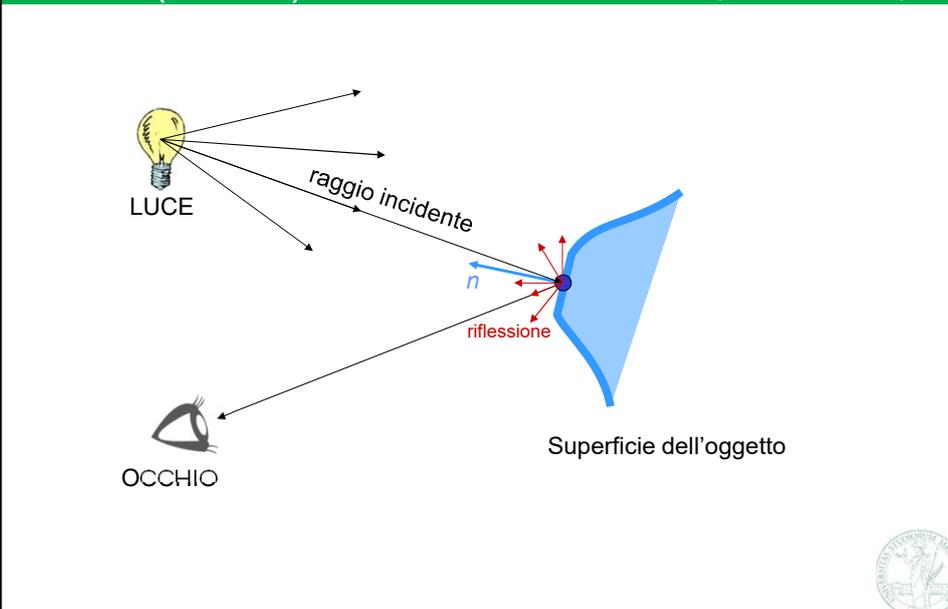
2

Lighting: intro

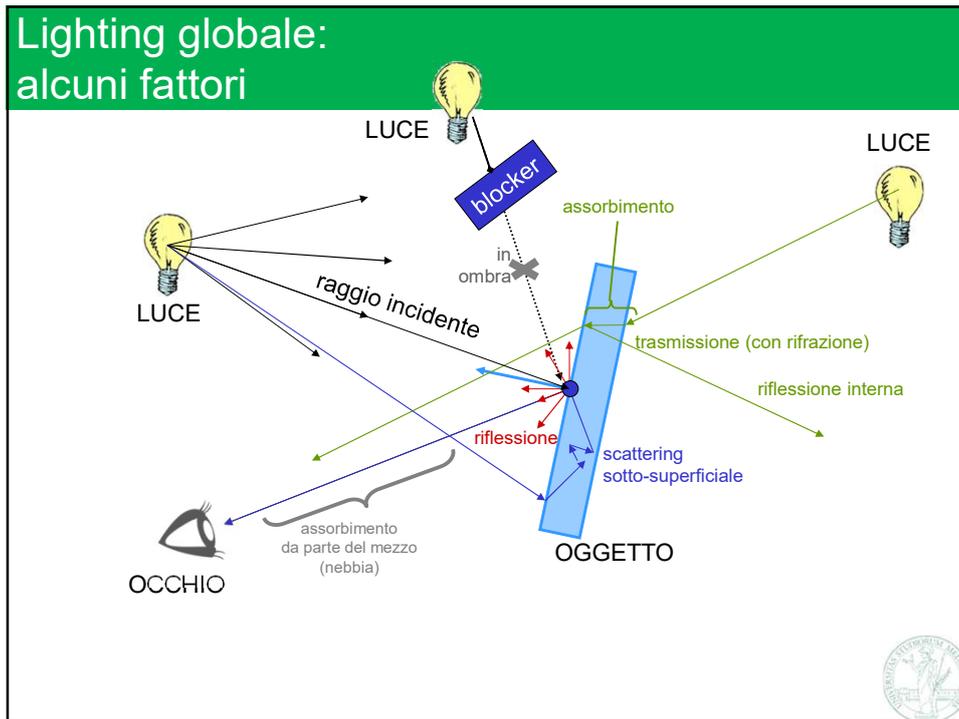
- ✓ Dopo la transform, il task principale del rendering
- ✓ Ci proponiamo di simulare una serie di fenomeni reali di interazione fra **luce** (radiazione elettromagnetica) e **scena** (come insieme di superfici)
- ✓ Il task, in concreto: determinare il **colore RGB** del pixel da generare ad ogni frammento
 - ⇒ In funzione di dati come:
 1. la normale locale della superficie (l'orientamento del pezzetto di superficie che rappresenta)
 2. L'illuminazione presente nella scena (come insieme discreto di luci, vedremo in che modo)
 3. Il «materiale» di cui è composto questo pezzetto di sup

3

Lighting locale: emitter (di luce) → surface → occhio (e nient'altro!)



5



6

Lighting: globale VS locale

Illuminazione locale	Illuminazione globale
Consideriamo solo: <ul style="list-style-type: none"> ⇒ il frammento di superficie da illuminare ⇒ a sua posizione, orientamento (normale) e materiale ⇒ la direzione di osservaz ⇒ la fonte (o le fonti) di luce 	Comprende fenomeni come <ul style="list-style-type: none"> ⇒ riflessioni multiple ⇒ ombre portate ⇒ scattering sottosuperficiale ⇒ rifrazione ⇒ ...
Caso particolare e semplificato, ma facile da ottenere anche con l'approccio rasterization-based	Realistica, ma richiede (nell'approccio rasterization-based) algoritmi specializzati (che non vedremo)

8

Materiali in CG

- ✓ In CG, il “materiale” è un modello matematico che descrive in che modo una data superficie reagisce alla luce, e comprende cose come
 - ⇒ Quanto appare chiaro oppure scuro
 - ⇒ Di che colore si presenta
 - ⇒ Quanto è lucido, oppure opaco (nel senso di «privo di riflessi lucidi», detti highlight)
 - ⇒ Quanto è trasparente, semitrasparente, traslucido, o permeabile alla luce
 - ⇒ Se e quanto il materiale è “cangiante”
 - ⇒ Se è fosforescente, se emette luce propria (oltre a riflettere quella di cui è investito), etc
- ✓ (nota: in alcune librerie ad alto livello, come three.js o unity, il materiale è una struttura che descrive quanto sopra, ma anche alcuni settaggi di rendering come quelli relativi a back-face culling, depth test, e l'uso di tessiture etc)



9

Materiali in CG, nel lighting locale

- ✓ Nel lighting locale, il materiale è esaustivamente descritto da una funzione che descrive in che modo la superficie riflette (rimbalza) la luce che la raggiunge da ogni possibile direzione
- ✓ Cioè *quanta* della luce che investe un pezzetto di superficie (da una certa direzione) viene riflessa dalla superficie (in ogni possibile direzione)
 - ⇒ E' una questione probabilistica perché il percorso di ogni singolo fotone non è prevedibile
 - ⇒ Nota: direzione = vettore unitario (come sappiamo)
- ✓ Si tratta quindi una funzione che prende in input:
 - ⇒ Una possibile *direzione* di provenienza della luce $\vec{\omega}_i$ (vettore unitario)
 - ⇒ Una possibile *direzione* di uscita della luce $\vec{\omega}_r$ (vettore unitario)e restituisce in output
 - ⇒ Quanta della luce proveniente da $\vec{\omega}_i$ verrà riflessa in $\vec{\omega}_r$
 - ⇒ Settata una direzione, è una distribuzione di probabilità sull'altra



10

La BRDF

- ✓ Questa funzione viene detta **BRDF**
 - ⇒ **B**idirectional (*poiché prende due direzioni, che sono intercambiabili*)
 - ⇒ **R**eflectance (*riflettanza, abilità di riflettere la luce*)
 - ⇒ **D**istribution (*perché è una distribuzione di riflettanza*)
 - ⇒ **F**unction
- ✓ La BRDF descrive in modo esaustivo un materiale (almeno per quello che riguarda il lighting locale)
- ✓ Ad esempio, materiali come...

⇒ Legno	⇒ Oro	⇒ Plastica levigata
⇒ Raso	⇒ Acciaio	⇒ Plastica ruvida
⇒ Stoffa	⇒ Argento	⇒ Vetro
⇒ Pietra	⇒ Cobalto	⇒ Diamante
⇒ Ferro arrugginito	⇒ Seta	⇒ Catarifrangenti
⇒ Velluto	⇒ (superficie dell') Acqua	⇒ etc

sono caratterizzati da BRDF specifiche

11

Descrizione di un materiale nel lighting locale: la BRDF

$f_r =$

fotone =

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

su 100 che arrivano a
 dalla dir $\vec{\omega}_i$,
 quante verranno da lui rimbalzate
 proprio nella dir $\vec{\omega}_r$?
 (al variare di $\vec{\omega}_i$ e $\vec{\omega}_r$)

Ogni materiale è descritto dalla sua BRDF.
 Ad esempio

- metallo cromato
- raso
- legno
- stoffa
- gesso
- carta
- specchio...

Funzione di 4 dimensioni!

 (una direzione = 2 dimensioni)

12

Materiali non uniformi

- ✓ In un **materiale non uniforme**, la funzione BRDF è diversa per ogni punto \mathbf{p} sulla superficie:
$$f_r(\mathbf{p}, \vec{\omega}_i, \vec{\omega}_r)$$
- ✓ Per ora occupiamoci solo di **materiali uniformi**, per i quali possiamo omettere il punto \mathbf{p} che stiamo illuminando:
$$f_r(\vec{\omega}_i, \vec{\omega}_r)$$
- ✓ Nella pratica, possiamo ottenere non uniformità di materiale assegnando valori che codificano la BRDF diversi (come il base color, vedi dopo)...
 - ⇒ per ogni **vertice** della **mesh**, come **attributo**, oppure
 - ⇒ ogni **texel** di una **tessitura** associata alla mesh



14

Alcuni esempi “estremi” di BRDF (nessun materiale reale esibisce davvero questo comportamento)

- ✓ La BRDF «tutta 0», perfettamente nera
$$f_r(\vec{\omega}_i, \vec{\omega}_r) = 0$$
 - ⇒ «da qualsiasi direzione provenga la luce, esattamente 0 di essa verrà riflessa verso ogni possibile direzione»
 - ⇒ Un materiale che ingoia *tutti* i fotoni che riceve
 - ⇒ Materiale reale che si avvicina: «vantablack» →

questa è una foto reale



- ✓ La BRDF perfettamente speculare, che riflette *tutta* la luce che proviene da una direzione $\vec{\omega}_i$ nella direzione $\vec{\omega}_r$, se $\vec{\omega}_r$ è il raggio riflesso $\vec{\omega}_i$ dalla normale
 - ⇒ Vedi lezione sul raytracing per determinare $\vec{\omega}_r$
 - ⇒ Materiale reale che si avvicina: una specchio molto buono →



17

La BRDF costanti: materiali "Lambertiani"

- ✓ Per ora occupiamoci di una semplice classe delle BRDF: quelle costanti
 - ⇒ dette puramente diffuse, o Lambertiane (da Lambert, v. dopo)
- ✓ I materiali che esibiscono questa BRDF sono detti diffusivi o Lambertiani
 - ⇒ Esempio di materiali che si avvicinano a questo: gesso, carta ruvida
 - ⇒ (in realtà è sempre solo un'approssimazione)
- ✓ Un materiale Lambertiano riflette la stessa percentuale di luce in tutte le direzioni possibili di uscita, indipendentemente da quale sia la direzione di arrivo della luce
 - ⇒ Quindi è privo di riflessi, esibisce lo stesso colore visto da tutti gli angoli
 - ⇒ Il lighting è *view-independent*: non dipende dalla direzione di luce



18

Materiale Lambertiano (o puramente diffusivo)

- ✓ E' definito un materiale in cui la BRDF è costante

$$f_r(\vec{\omega}_i, \vec{\omega}_r) = \text{const}$$

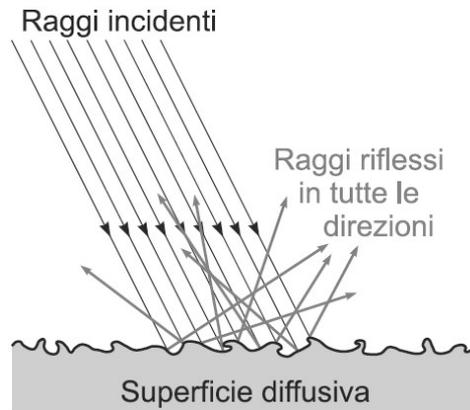
- ✓ Detto anche materiale (puramente) diffusivo
 - ⇒ Una superficie di questo materiale riflette la luce da cui è raggiunta *diffondendola uniformemente*
- ✓ La costante è detta l'**albedo** del materiale: il rapporto fra la luce incidente e la luce riflessa
 - ⇒ da 0, materiale perfettamente nero, assorbe tutta la luce (la BRDF «perfettamente nera», quella del «vanta black»)
 - ⇒ a 1, materiale perfettamente bianco, diffonde tutta la luce, senza assorbirne alcuna



19

Materiali «lambertiano» (o diffusivi) reali

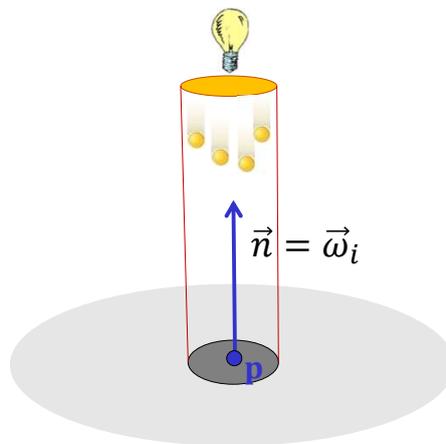
- ✓ Si verificano quando...
 - ⇒ a livello **microscopico**, la superficie presenta microsfaccettature disposte in modo molto irregolare, caotico
 - ⇒ altri fenomeni di interazioni fra fotone e superficie
- ✓ Nota:
 - ⇒ le BRDF dei materiali dipendono dalle forma microscopica della superficie (e altro, come le proprietà elettromagnetiche)



20

Sottoproblema

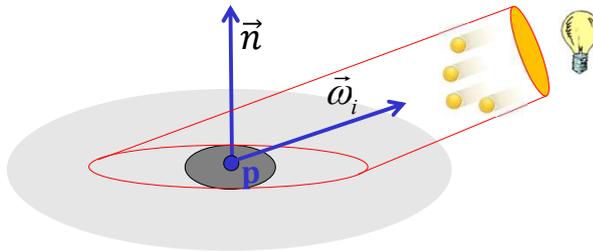
- ✓ Se dalla dir $\vec{\omega}_i$ arrivano L lumens , quante ne riceve un intorno di \mathbf{p} con normale \vec{n} ?



22

Sottoproblema

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

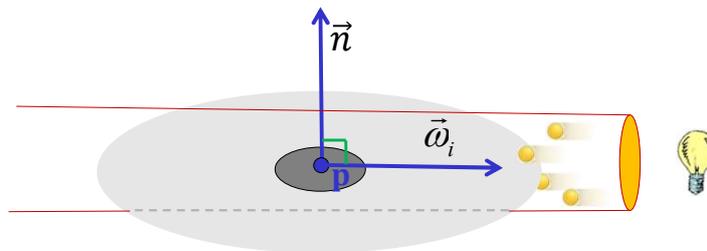


24

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



26

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")
 $\cos(\alpha)L$
=
 $(\vec{\omega}_i \cdot \vec{n})L$

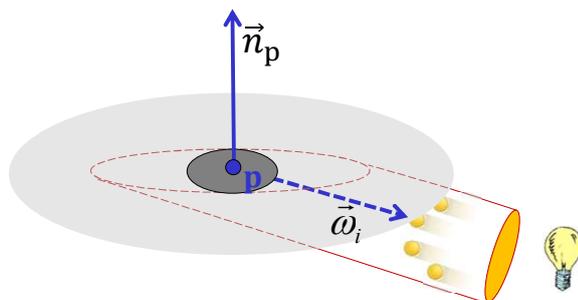
Johann
Heinrich
Lambert
1728 - 1777

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 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)



29

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. $\vec{\omega}_r$

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

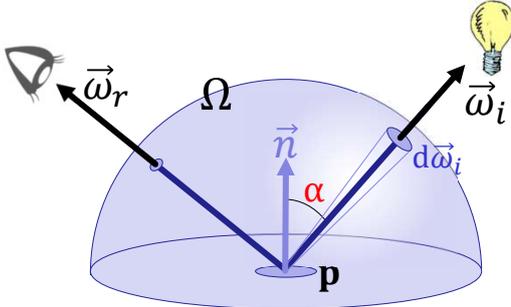
30

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

31

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



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

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

32

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}
 Ω : 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})

33

Ambiente di illuminazione generico: la luce incidente (da ogni direzione)

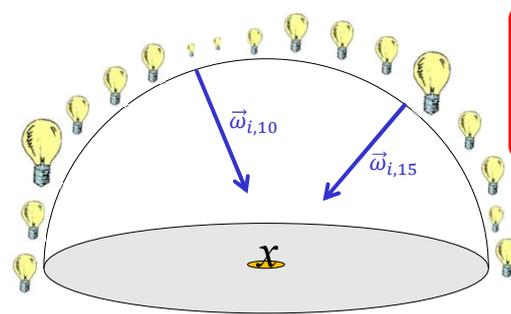
✓ Per ogni posizione \mathbf{p}
 $L_i(\mathbf{p}, \vec{\omega}_i)$ modella
la distribuz di luce incidente

$L_i(\mathbf{p}, \vec{\omega}_i) =$ 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
- gioranta coperta
- una discoteca

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



34

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 poche direzione costanti $\vec{\omega}_i$ (un parametro; per esempio, verso il sole), con intensità L_i dalla direzione i . Allora da integrale passo a sommatoria:

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



38

Equazione di lighting per materiali lambertiani (o puramente diffusivi)

- ✓ L'equazione di lighting risultante per una luce è

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

Somma su un piccolo numero di luci (discrete) → \sum_i
 Prodotto dot, ma zero se negativo → $(\vec{\omega}_i \cdot \vec{n})$
 Luminosità finale del pixel → P
 direzione di arrivo della luce i (vettore unitario, va verso la luce) → $\vec{\omega}_i$
 normale della sup → \vec{n}
 intensità della luce i → L_i
 albedo del materiale (quanto è chiaro o scuro il suo aspetto) → D



39

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

$$P_R = \sum (\vec{n} \cdot \vec{\omega}_i) D_R L_{i,R}$$

$$P_G = \sum (\vec{n} \cdot \vec{\omega}_i) D_G L_{i,G}$$

$$P_B = \sum (\vec{n} \cdot \vec{\omega}_i) D_B L_{i,B}$$

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

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

Idem, per il verde

Idem, per il blue

40

Lighting lambertiano... a colori (una riscrittura)

Scalatura del vettore RGB (nessun simbolo)

$$\begin{pmatrix} P_R \\ P_G \\ P_B \end{pmatrix} = \sum_i (\vec{n} \cdot \vec{\omega}_i) \begin{pmatrix} D_R \\ D_G \\ D_B \end{pmatrix} \otimes \begin{pmatrix} L_{i,R} \\ L_{i,G} \\ L_{i,B} \end{pmatrix}$$

Questo simbolo denota qui il "prodotto componente per componente"

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 i -esima, su ciascun canale (in sostanza: colore e intensità della luce i)

41

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.»



42

Esempio: lighting lambertiano con due luci

- ✓ Esempio 2 sorgenti di luce:
 ogni luce i -esima avrà la propria direzione $\vec{\omega}_{ik}$ e intensità (L_{Ri}, L_{Gi}, L_{Bi})
 ⇒ 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 luci

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

Diagram illustrating the additive lighting model for two light sources. The final pixel color vector $\begin{pmatrix} P_R \\ P_G \\ P_B \end{pmatrix}$ is the sum of two terms. Each term consists of the cosine of the angle between the surface normal \vec{n} and the light direction $\vec{\omega}_{ik}$, multiplied by the base color vector $\begin{pmatrix} D_R \\ D_G \\ D_B \end{pmatrix}$ and the light intensity/color vector $\begin{pmatrix} L_{Ri} \\ L_{Gi} \\ L_{Bi} \end{pmatrix}$. The base color vector is shared between both light sources. Labels with arrows point to each component: 'pixel finale' points to the left vector, 'Direzione luce 1' points to $\vec{\omega}_{i1}$, 'Intensità / colore della luce 1' points to $\begin{pmatrix} L_{R1} \\ L_{G1} \\ L_{B1} \end{pmatrix}$, 'Direzione luce 2' points to $\vec{\omega}_{i2}$, and 'Intensità / colore della luce 2' points to $\begin{pmatrix} L_{R2} \\ L_{G2} \\ L_{B2} \end{pmatrix}$. A 'Base color' label with a double-headed arrow spans the two base color vectors.



43

Lighting con più sorgenti di luce: osservazioni pratiche

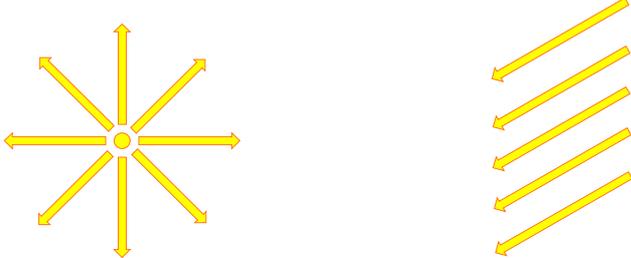
- ✓ L'aggiunta di ogni luce comporta un aggravio di computazione del lighting
 - ⇒ Dobbiamo computare un altro termine, su ogni frammento della scena
- ✓ In assenza di qualsiasi fonte di luce, il lighting degli oggetti ha zero addendi, e il colore risultante è $(0,0,0)$
 - ⇒ Com'è sensato che sia:
al 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 fievole



44

Modellazione delle sorgenti di luce

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



LUCE POSIZIONALE

LUCE DIREZIONALE



45

Luci posizionali e direzionali

- ✓ Per la luce **direzionale**, la direzione di luce incidente $\vec{\omega}_i$ è una costante, e non varia nella scena
 - ⇒ $\vec{\omega}_i$ (vettore unitario) 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:

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



46

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 viene 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

$$f_{affievolimen} = \frac{1}{d^2}$$

⇒ 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 – verifica



47

Modelli (o equazioni) di lighting

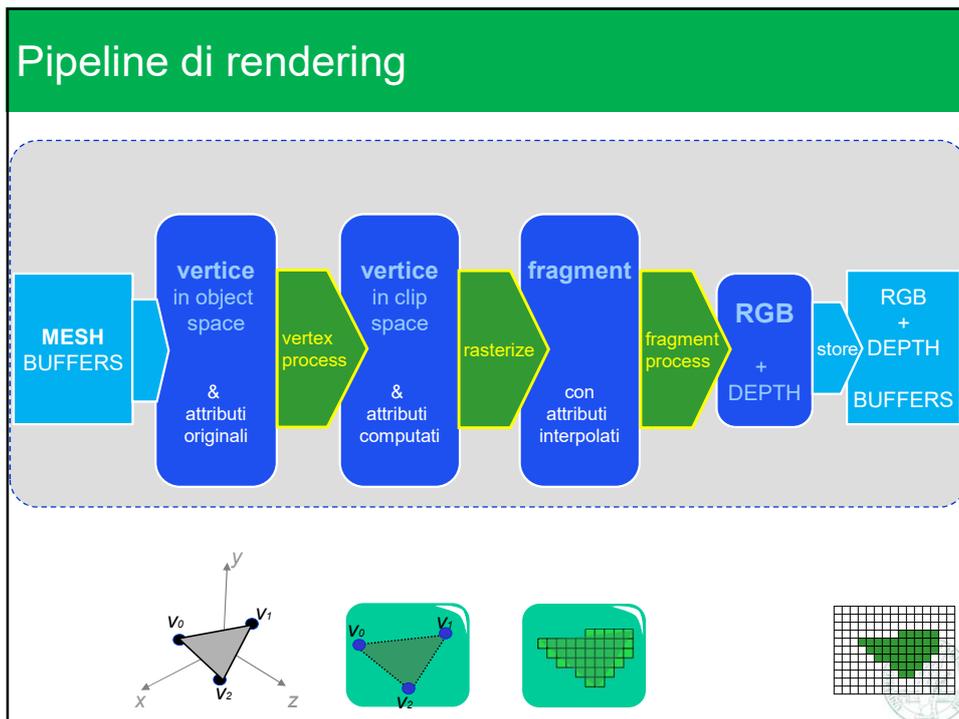
- ✓ L'equazione di lighting vista è molto semplice
 - ⇒ consiste esclusivamente di una componente di riflessione diffusa
 - ⇒ però è physically based – il fenomeno ottico modellato è accuratamente riprodotto (eccetto che per la tricromia RGB)
- ✓ E' un esempio di modello di illuminazione locale
- ✓ Il questo modello:
 - ⇒ il «materiale» viene descritto interamente dal base color (detto anche diffuse color -- o albedo se in bianco e nero)
 - ⇒ L'unico aspetto geometrico 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 osservazione: 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

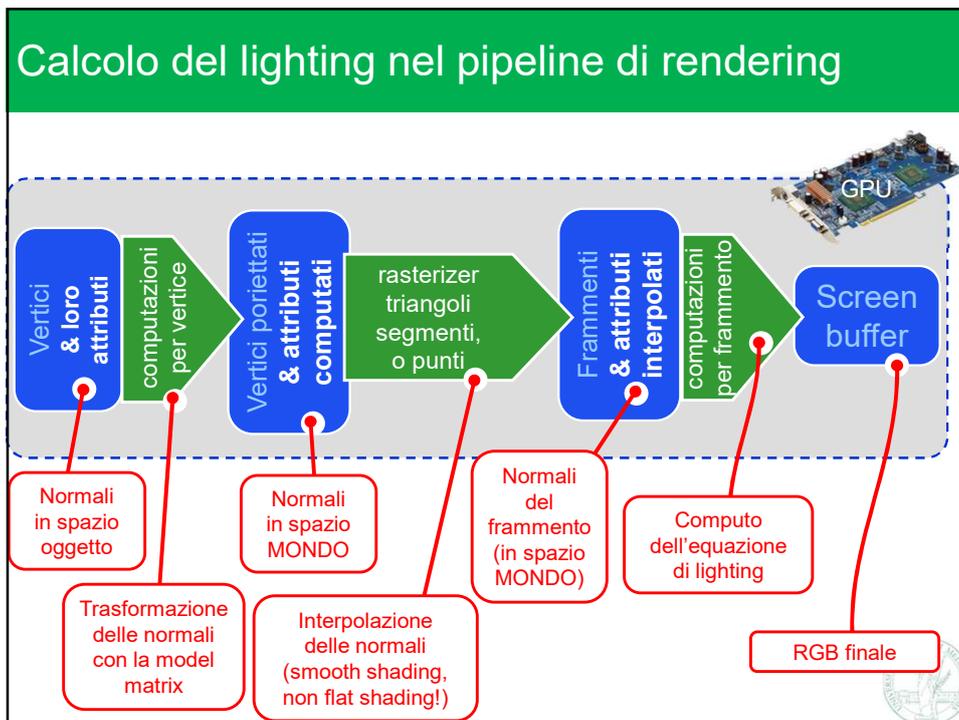
Calcolo del lighting nel pipeline di rendering

- ✓ 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



51

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

- ✓ 1: rendiamo il materiale lambertiano

```
var materiale = new THREE.MeshLambertMaterial();  
var unaMesh = new THREE.Mesh( ...geometria..., materiale );
```

(questo istruisce three.js ad usare il pipeline come descritto sopra)
- ✓ 2: il comando che prima determinava il colore finale, ora, con il lighting, determina settiamo il base color (o diffuse color) del materiale

```
materiale.color.setRGB( ... , ... , ... );
```
- ✓ 3: creiamo le luci e aggiungiamole alla scena
ad esempio, una luce direzionale

```
var luceA = new THREE.DirectionalLight();  
miaScena.add(luceA);
```
- ✓ 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)

```
luceA.color.setRGB(1,1,1);  
luceA.position.set(.4,1,.3); // viene normalizzato
```

Vedere il progetto [lab06.html](#)



53

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

Possiamo aggiungere una seconda luce:

```
var luceB = new THREE.DirectionalLight();  
luceB.color.setRGB( ... ); // intensità / colore  
luceB.position.set( ... ); // direzione  
miaScena.add(luceB);
```

- ✓ nota come la luce si somma alla precedente

Oppure, una luce *posizionale* (provare, come esercizio).

- ✓ settiamone la posizione (per es, x=2,y=1.5,z=5)

```
var luce2 = new THREE.PointLight();  
luce2.position.set(2,1.5,5); // la posizione!
```
- ✓ notare l'effetto affievolimento con la distanza alla n (default 2), che può essere controllato con
- ✓

```
luce2.decay = ...; // 0 = no decay. 2 = decay fisico
```

Vedere il progetto [lab06.html](#)



54