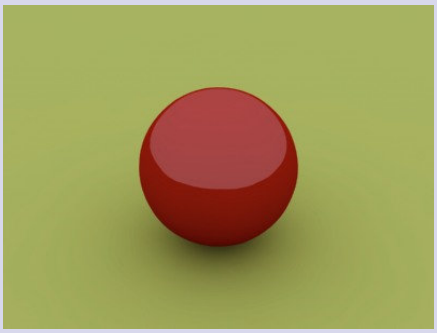
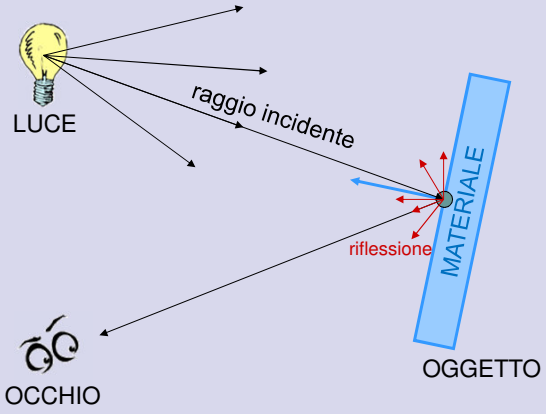
Local Lighting



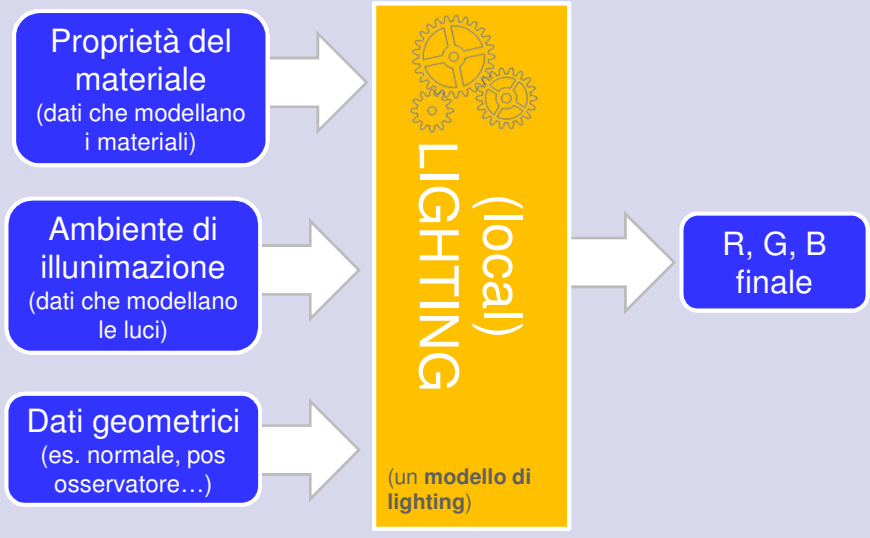
(recall?) Game Engine

- Parte del game che si occupa di alcuni dei task “comuni”
  - Scena / livello
  - Renderer
    - Real time transofrm + lighting
    - Models, materials …
  - Physics engine
    - (soft real-time) newtonian physical simulations
    - Collision detection + response
  - Networking
    - (LAN – es tramite UTP)
  - Sound mixer e “sound-renderer”
  - Gestore unificato HCI devices
  - Main event loop, timers, windows manager…
  - Memory management
  - Artificial intelligence module
    - Soluz dei sotto task comuni AI
  - Supporto alla localizzazione
  - Scripting
  - GUI (HUD)

Lighting locale

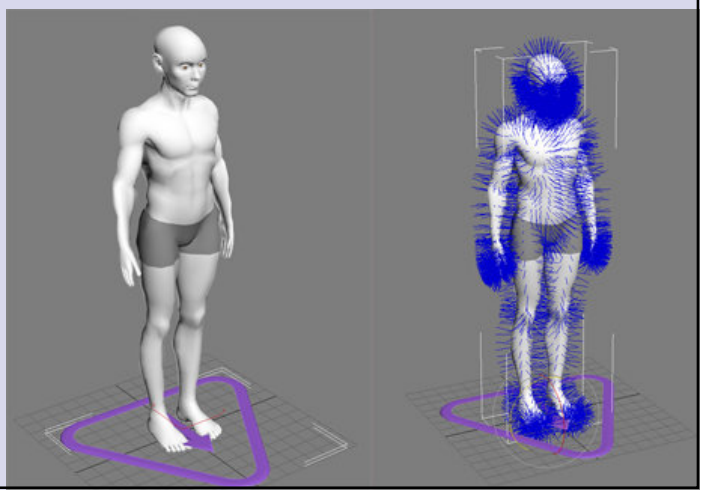


Lighting in breve

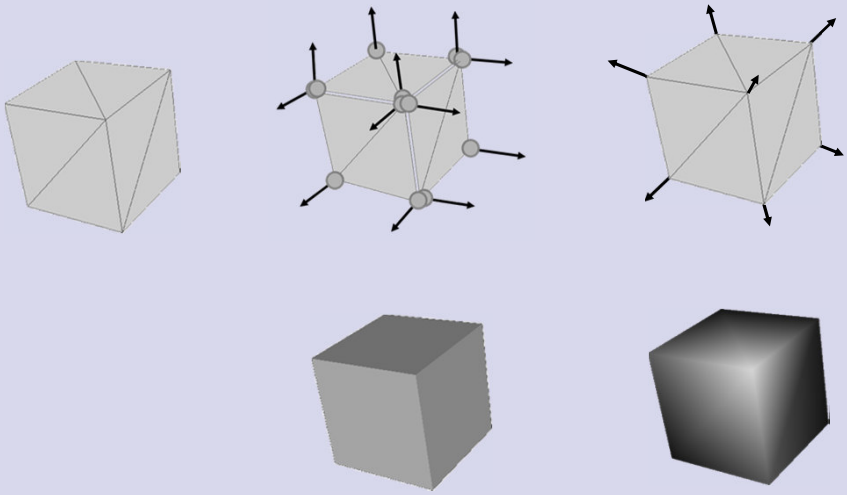


Reminder: le normali

- Attributo per vertice delle mesh…

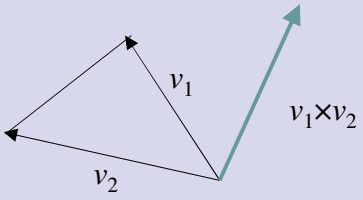


Reminder: seams per hard edges



normali: es. di una possibile pre-computazione

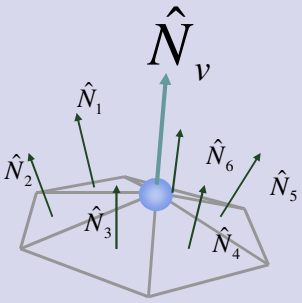
Normale di un Triangolo:



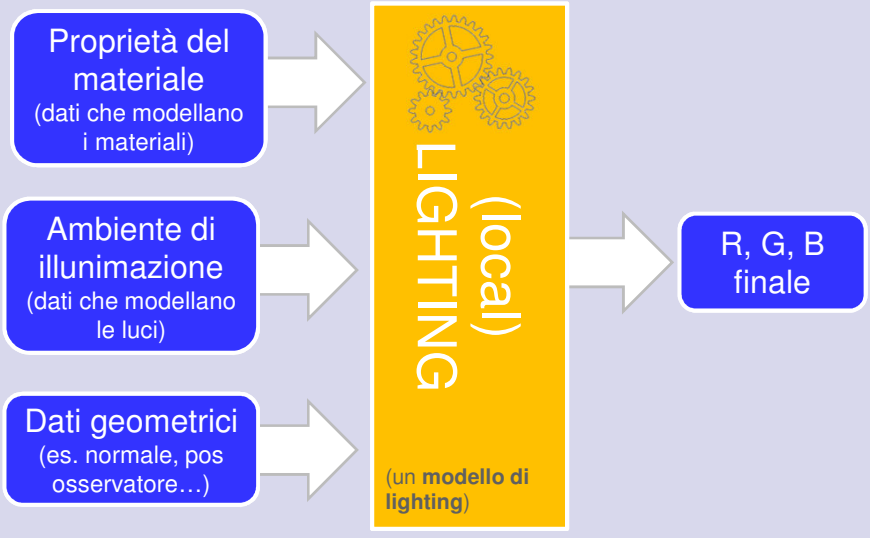
Normale di un vertice (condiviso da *n* triangoli):

$$N = \hat{N}_1 + \hat{N}_2 + \ldots + \hat{N}_n$$

$$\hat{N} = \frac{N}{|N|}$$

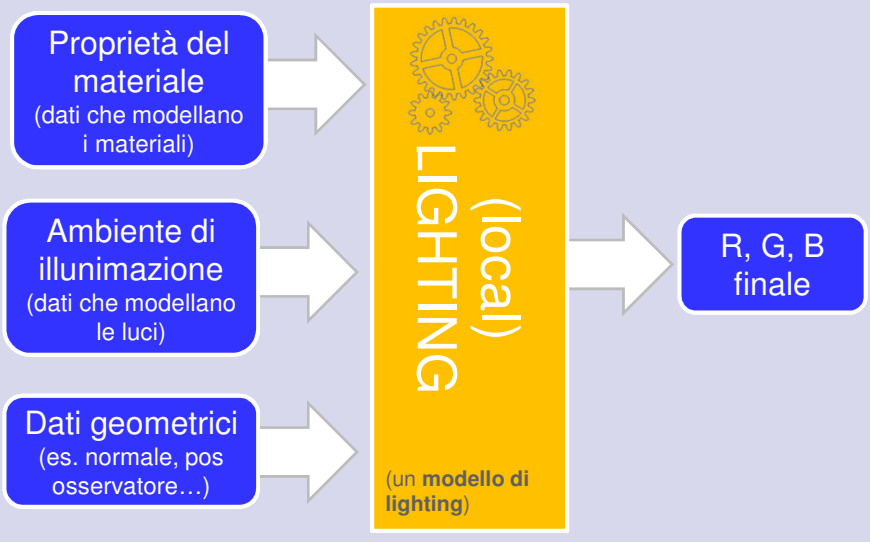


Lighting in breve



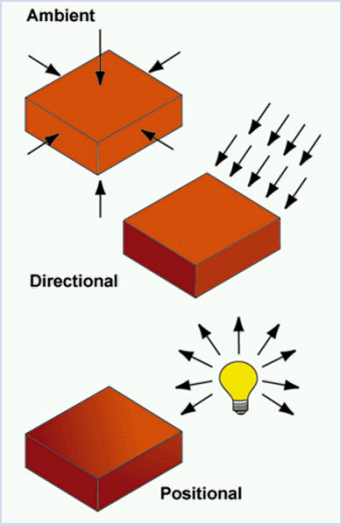
Materiale

Lighting in breve

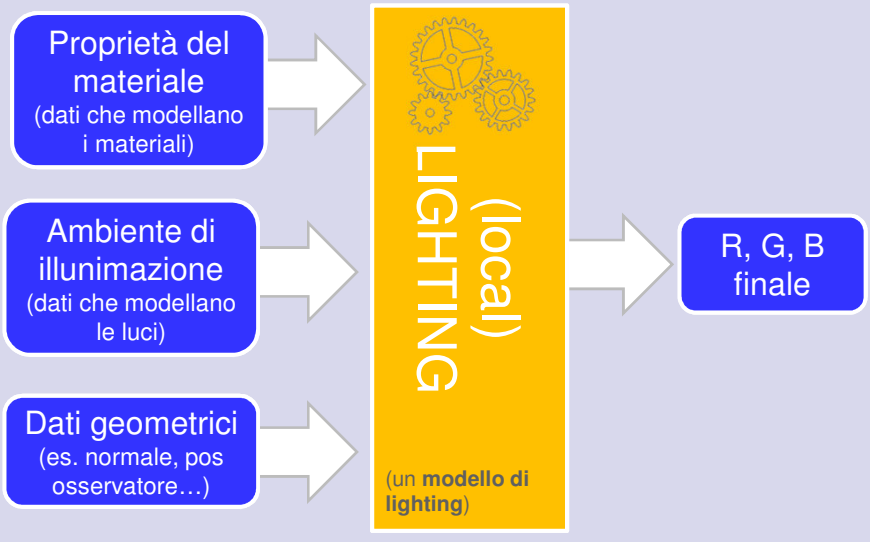


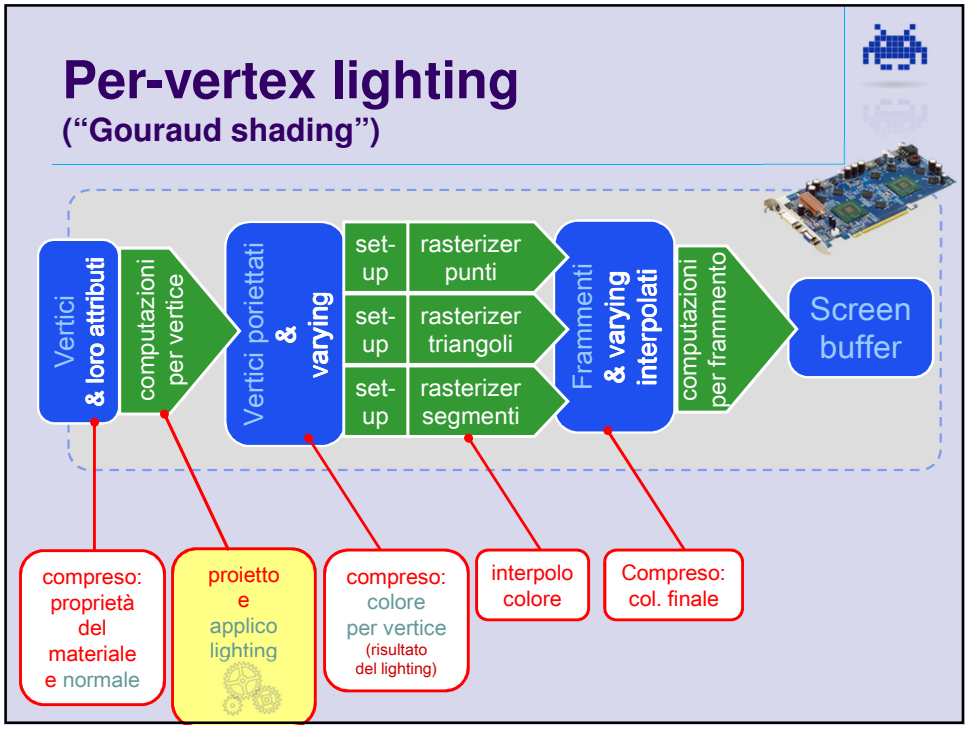
Ambiente di illuminazione

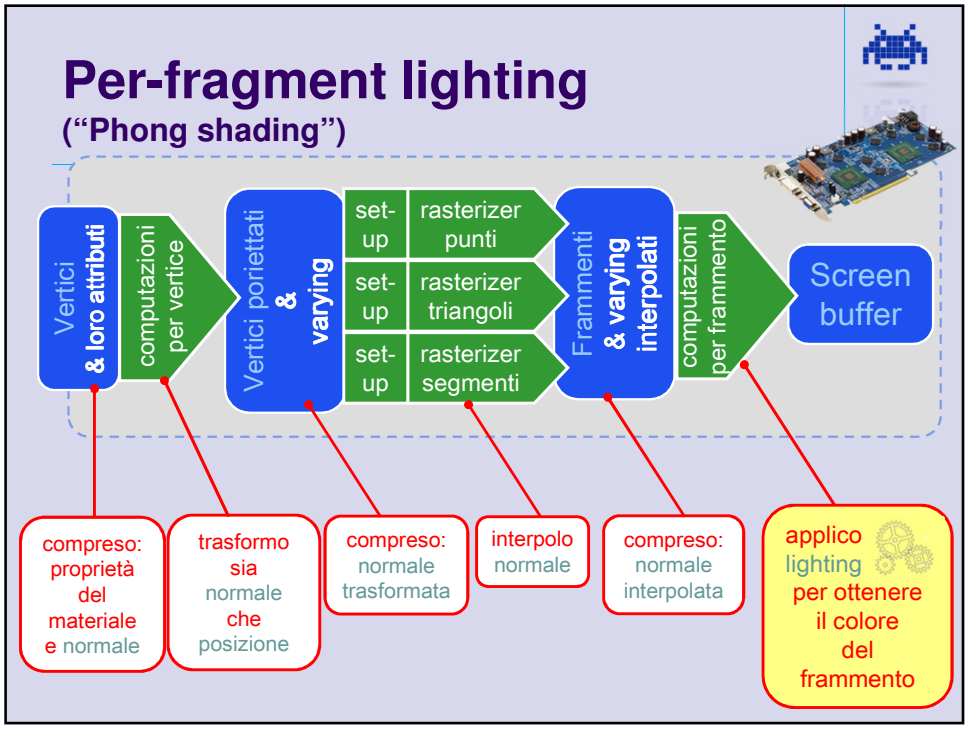
- Insieme (discreto) di emettitori di luce
  - + componente ambient
- x ogni luce:
  - geometria: puntiforme o direzio
  - intensità / colore
  - area affected:
    - area light
    - spot light



Lighting in breve

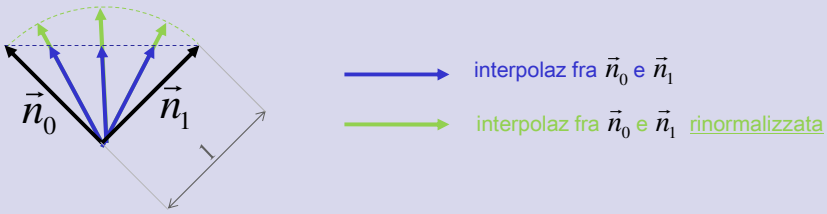



Per-vertex lighting
("Gouraud shading")
Vertici & loro attributi
computazioni per vertice
Vertici proiettati & varying
set-up
rasterizer punti
set-up
rasterizer triangoli
set-up
rasterizer segmenti
Frammenti & varying interpolati
computazioni per frammento
Screen buffer
compreso: proprietà del materiale e normale
proietto e applico lighting
compreso: colore per vertice (risultato del lighting)
interpolo colore
Compreso: col. finale


Per-fragment lighting
("Phong shading")
Vertici & loro attributi
computazioni per vertice
Vertici proiettati & varying
set-up
rasterizer punti
set-up
rasterizer triangoli
set-up
rasterizer segmenti
Frammenti & varying interpolati
computazioni per frammento
Screen buffer
compreso: proprietà del materiale e normale
trasformo sia normale che posizione
compreso: normale trasformata
interpolo normale
compreso: normale interpolata
applico lighting per ottenere il colore del frammento

Per vertex lighting

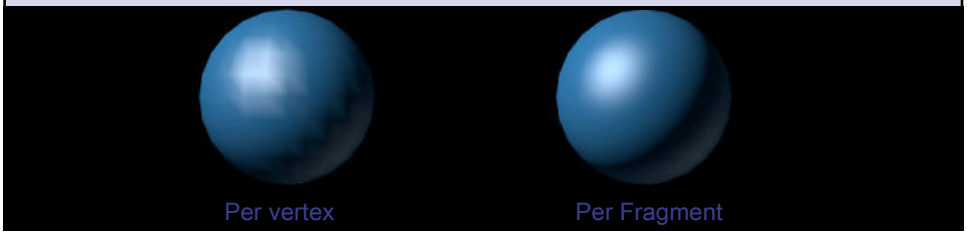
- Invece di interpolare il colore *dopo* il lighting.
  interpolo la normale *prima* del lighting!
- dettaglio: interpolando due vettori normali,
  non ottengo un vettore normale:
  (bisogna rinormalizzare dopo l'interpolazione)

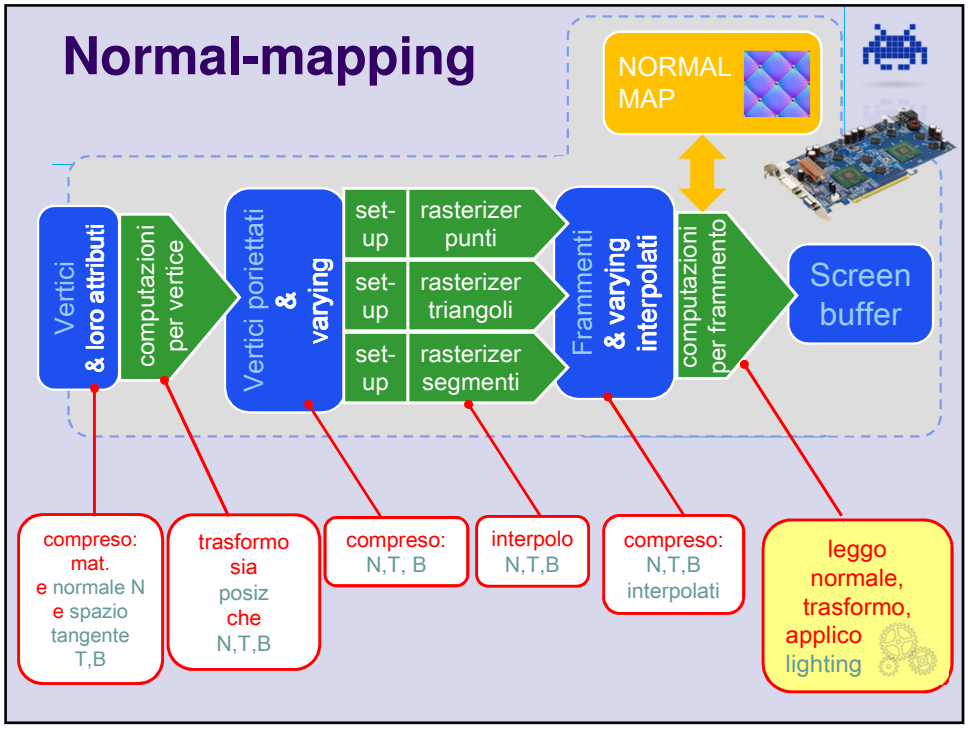


Per vertex VS Per fragment

Per-vertex Lighting - (a.k.a. "Gouraud Shading")
meno oneroso:
applico il lighting una volta per vertice – di solito #vertici << #frammenti

Per-fragment* Lighting – (a.k.a. " Phong Shading")
risultati migliori (*) talvolta (ma impropriamente) detto: per-pixel lighting
specialmente con i riflessi luminosi e piccoli (alta "glossiness")





Computazione lighting

- Scritta nel *vertex shader*
  o nel *fragment shader*