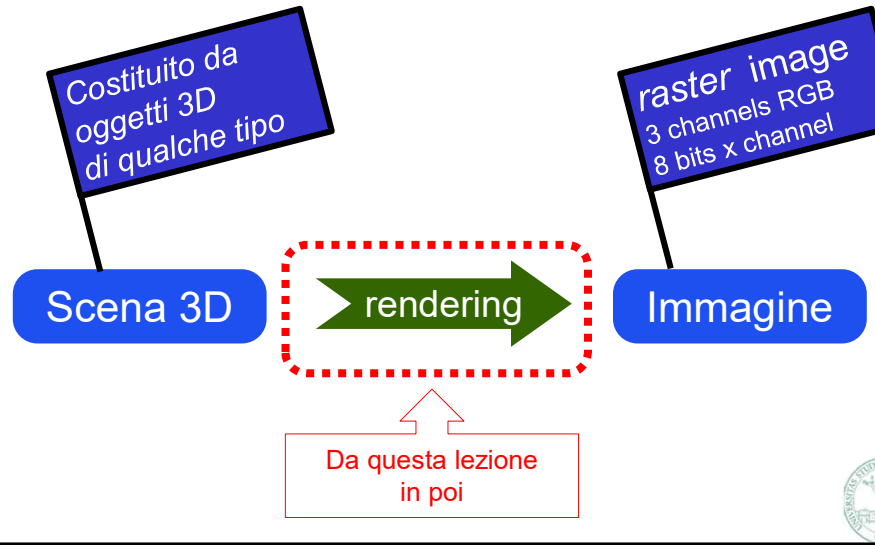
Marco Tarini - Computer Graphics 2022/2023
Università degli Studi di Milano

Rendering: ray-tracing

1

Rendering



2

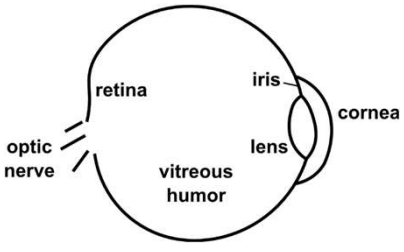
Rendering

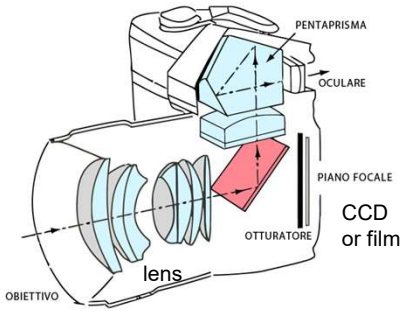
- ✓ Una classe di algoritmi di rendering prende diretta ispirazione dal modo in cui un dispositivo reale di cattura di immagini produce un'immagine
  - ⇒ Dispositivi come:
    una macchina fotografica (analogica o digitale), una telecamera, un occhio umano, una camera oscura
- ✓ A questo fine, possiamo rimuovere tutti gli elementi non necessari di tali dispositivi, per ridurli ad un modello più semplice possibile
  - ⇒ Questo modello è detto Pin-hole camera.

4

Apparati reali che sintentizzano immagini

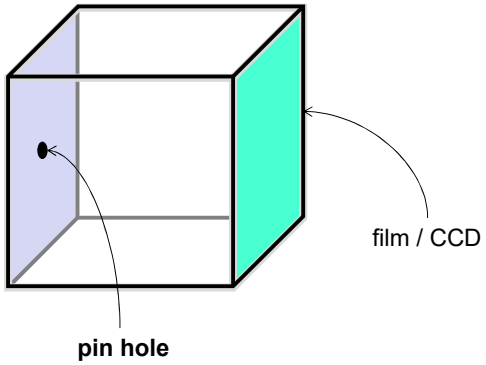
- ✓ Human Visual System
- ✓ Fotocamera





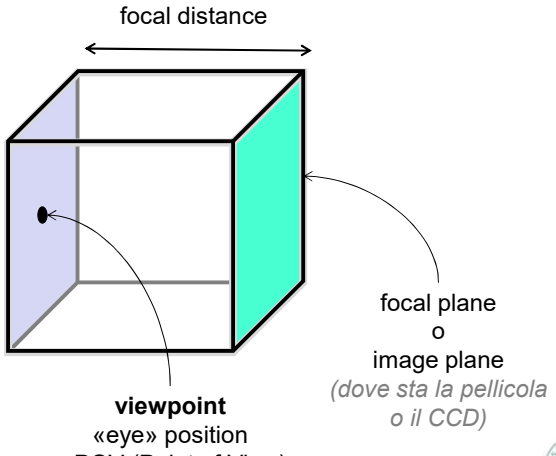
5

Pin-Hole camera



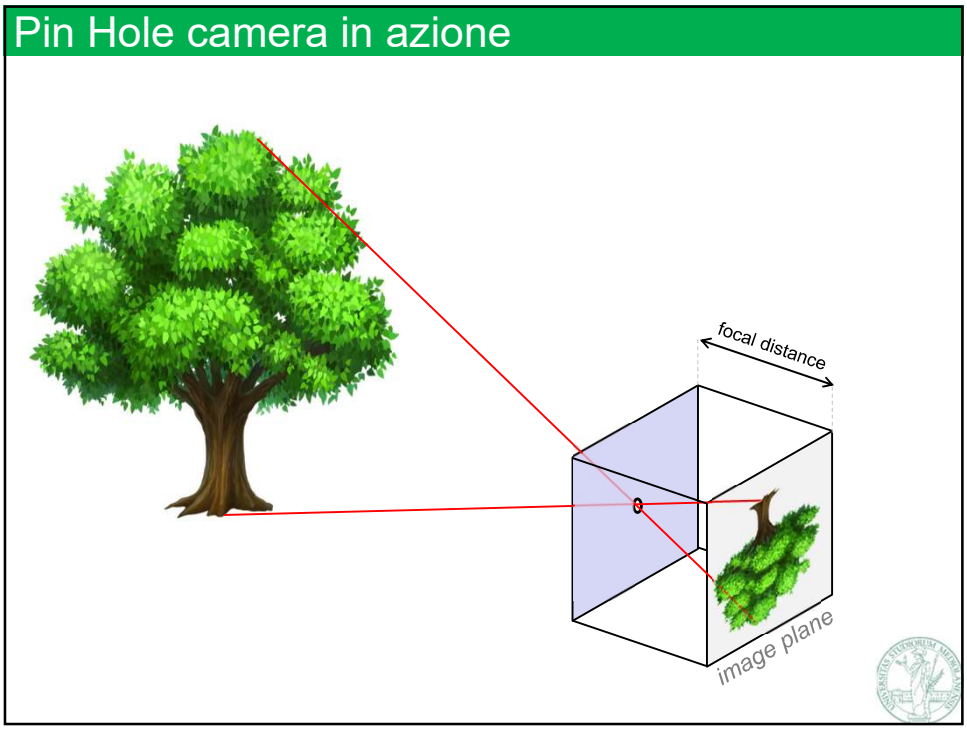
6

Pin-Hole camera



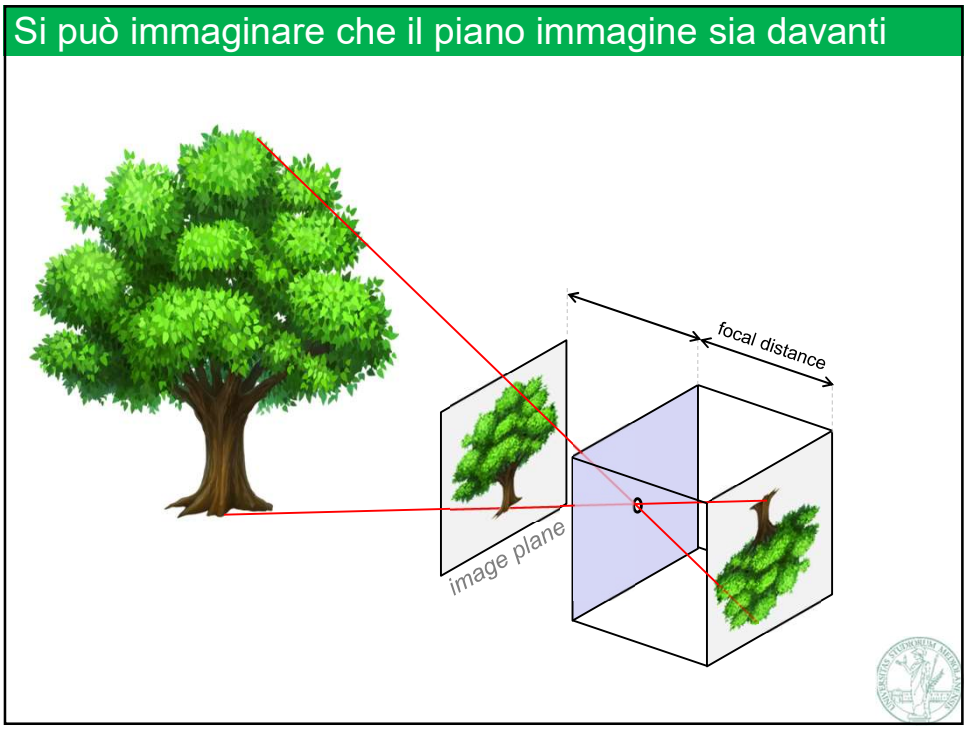
8

Pin Hole camera in azione



9

Si può immaginare che il piano immagine sia davanti



10

Pin-hole camera

- ✓ La pin-hole camera cattura la luce che passa per il POV
  - ⇒ la luce = i fotoni
- ✓ La pin-hole camera misura quanta di questa luce arriva da ogni direzione
  - ⇒ ogni pixel rilevato p[i,j] = quantità di luce che è passata dal POV in direzione (simile a) (POV – P(i,j)) durante il tempo di esposizione

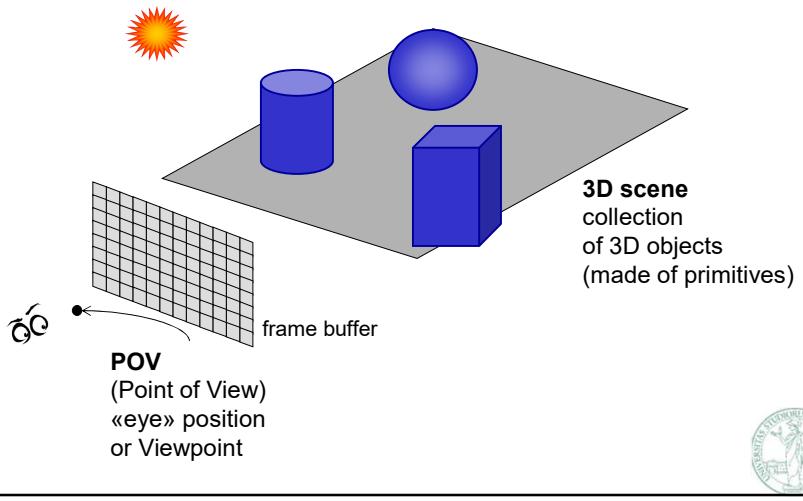
11

Ray-Casting

- ✓ Idea: seguire *a ritroso* i fotoni che raggiungono il POV
  - ⇒ per questo, detto anche : *"backward" ray tracing*
- ✓ per ogni pixel sull'immagine da produrre:
  - ⇒ costruisco un raggio (detto il "raggio primario" di quel pixel)
  - ⇒ trovo le sue intersezioni con gli oggetti della scena
  - ⇒ scelgo l'intersezione più vicina al POV
- ✓ Implementazione:
  - ⇒ Numero di raggi da processare = numero di pixel dell'immagine da produrre
  - ⇒ Per ogni raggio, è necessario computare la sua intersezione con gli oggetti che costituiscono la scena (cioè le "primitive") (questa operazione deve quindi deve essere ottimizzata)

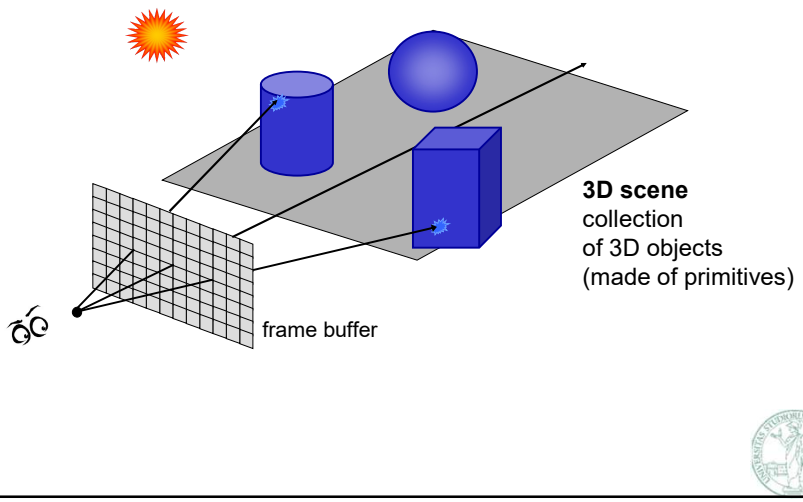
15

Ray casting



16

Ray casting



17

Ray Casting pseudo-code

```
For each pixel p:
   make a ray r (POV to p)
   for each primitive o in scene:
      find intersect(r,o)
   keep closest intersection o_j
   find color of o_j at p
```

18

Primitive di rendering (precisazione)

- ✓ Per “primitiva di rendering” si intende una descrizione di un entità (3D o 2D) che un dato algoritmo di rendering può processare direttamente
  - ⇒ Diversi algoritmi di rendering prevedono diverse primitive
- ✓ Ad esempio:
  - ⇒ Se disponiamo di un algoritmo in grado di processare… triangoli 3D, ma non quadrilateri 3D,
    (primitiva di rendering = triangolo)
    allora possiamo renderizzare direttamente una tri-mesh,
    ma una quad-mesh deve prima essere convertita
    in una tri-mesh (attraverso diagonal split)
  - ⇒ Se disponiamo di un algoritmo di rendering in grado di processare… campi di altezza, allora non è necessario convertire il campo di altezza in una rappresentazione poligonale
- ✓ Come vedremo, la primitiva prevista dagli algoritmi di rendering più diffusi è il triangolo (ma non è l’unica!)
  - ⇒ Da questo discende la predominanza delle tri-mesh come modelli 3D

19

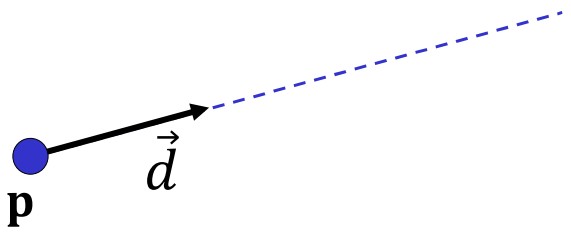
Primitive di rendering del Ray-casting

- ✓ Con un algoritmo di Ray-casting possiamo renderizzare qualsiasi cosa per la quale siamo in grado di computare l'intersezione con un raggio
  - ⇒ Vale anche per ualsiasi algoritmo di Ray-tracing – vedi sotto
- ✓ Esistono quindi ray-caster per il rendering di :
  - ⇒ Mesh poligonali
  - ⇒ Campi d'altezza
  - ⇒ Modelli impliciti
  - ⇒ Superfici parametriche, come patch di Bezier
  - ⇒ Etc
- ✓ In ciascun caso, è necessario implementare la procedura di intersezione raggio-oggetto
  - ⇒ Vedremo alcuni casi per sempici modelli impliciti

20

Rappresentazioni di un «raggio»

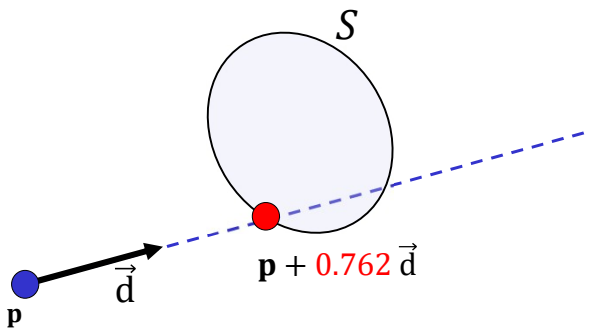
- ✓ Un raggio (ray) è una semiretta
- ✓ La possiamo modellare come
  - ⇒ Un dato punto $\mathbf{p}$ di partenza del raggio
  - ⇒ Un dato vettore $\vec{d}$ direzione (unitario o no, è una scelta implementativa)
- ✓ I punti sul raggio sono i punti esprimibili come $\mathbf{p} + k\vec{d}$
  - ⇒ per un qualche scalare k
  - ⇒ con $k \geq 0$ (perché il raggio è una SEMI retta)
  - ⇒ cioè «i punti raggiungibili a partire da $\mathbf{p}$ facendo k passi in direzione $\vec{d}$ »



21

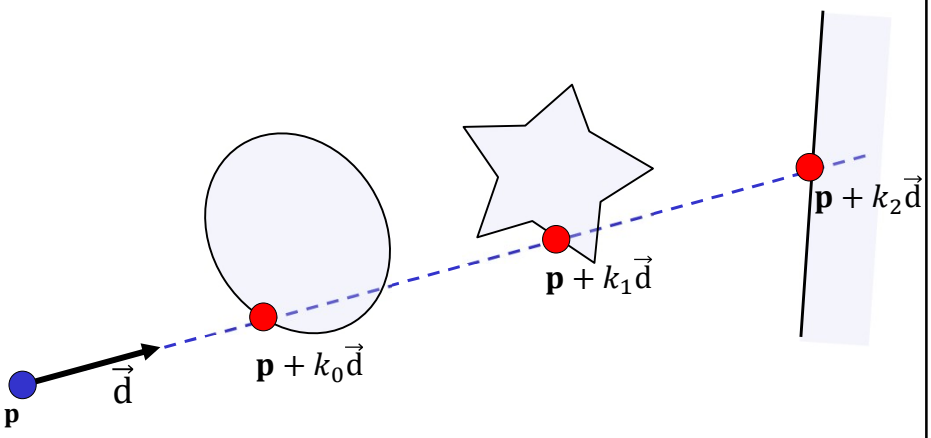
Intersezione raggio – oggetto singolo

✓ Trovare l'intersezione con una data superficie S significa trovare uno scalare k t.c. $\mathbf{p} + k\vec{d}$ stia su S



22

Interesezione raggio-scena (cioè tante primitive)



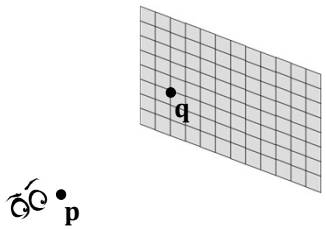
Basta prendere la primitiva che interseca col k **minore** fra $k_{0,1,2}$

23

Costruzione del raggio primario

- ✓ Dato un pixel Q, quale è il suo raggio primario?
- ✓ Risposta:
  - ⇒ $\mathbf{p}$ (punto di partenza del raggio): il POV
  - ⇒ $\vec{\mathrm{d}}$ (direzione del raggio): $(\mathbf{q} - \mathbf{p})$ (eventualmente, normalizzato)

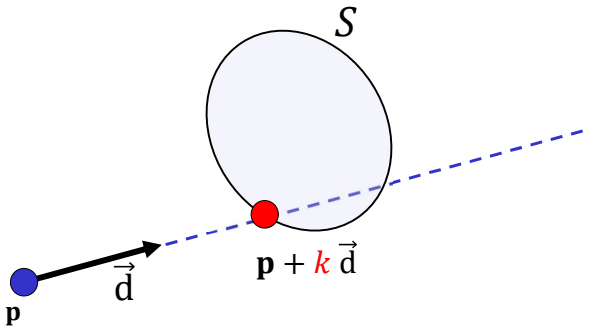
  dove $\mathbf{q}$ è la posizione 3D del pixel Q sul piano immagine



24

Intersezione raggio – oggetto singolo

- ✓ Vedremo nelle prossime lezioni alcuni casi di computo dell'intersezione di un raggio con alcune semplici primitive
- ✓ Prima, sarà necessaria una breve digressione matematica sul prodotto dot



25