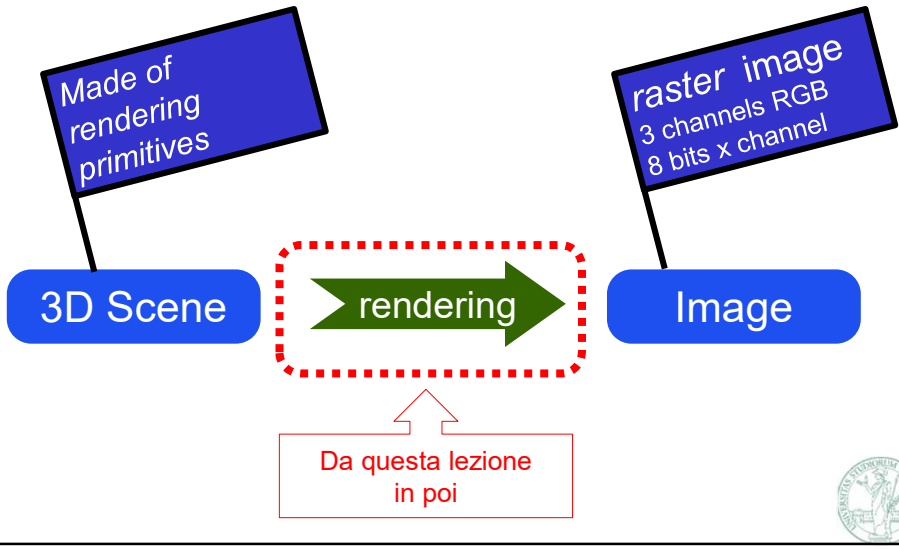
Marco Tarini - Computer Graphics 2019/2020
Università degli Studi di Milano

Rendering: ray-tracing

1

Rendering



2

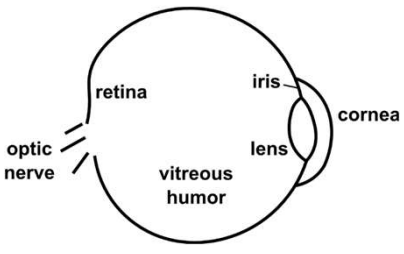
Rendering

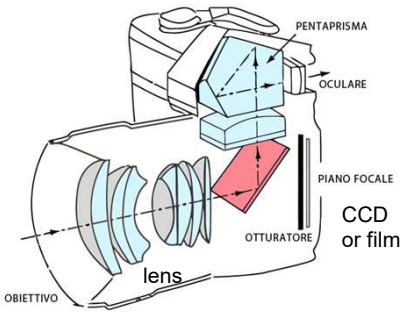
- ✓ Molti algoritmi di rendering prendono diretta ispirazione dal modo in cui un dispositivo reale di sintesi di immagini produce un'immagine
  - ⇒ Dispositivi come: macchina fotografica, una telecamera, un occhio umano
- ✓ A questo fine, possiamo rimuovere tutti gli elementi non necessari di tali dispositivi,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

4

Modelling the picture making apparatus

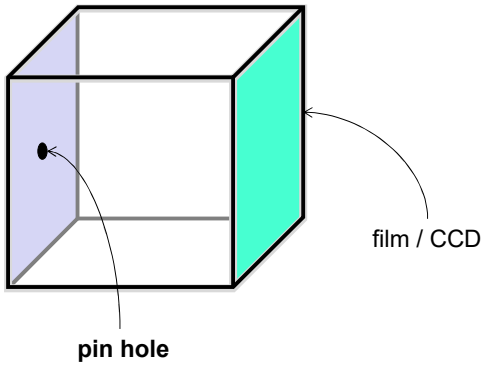
✓ Human Visual System ✓ Camera





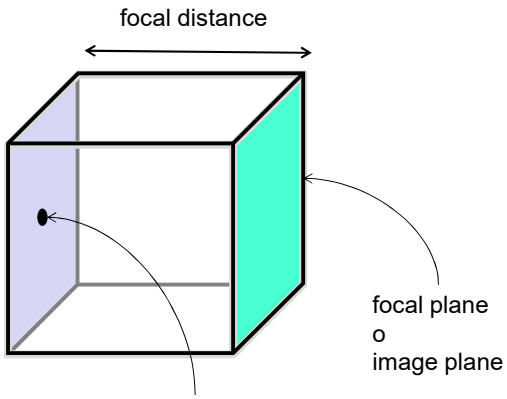
5

Pin-Hole camera



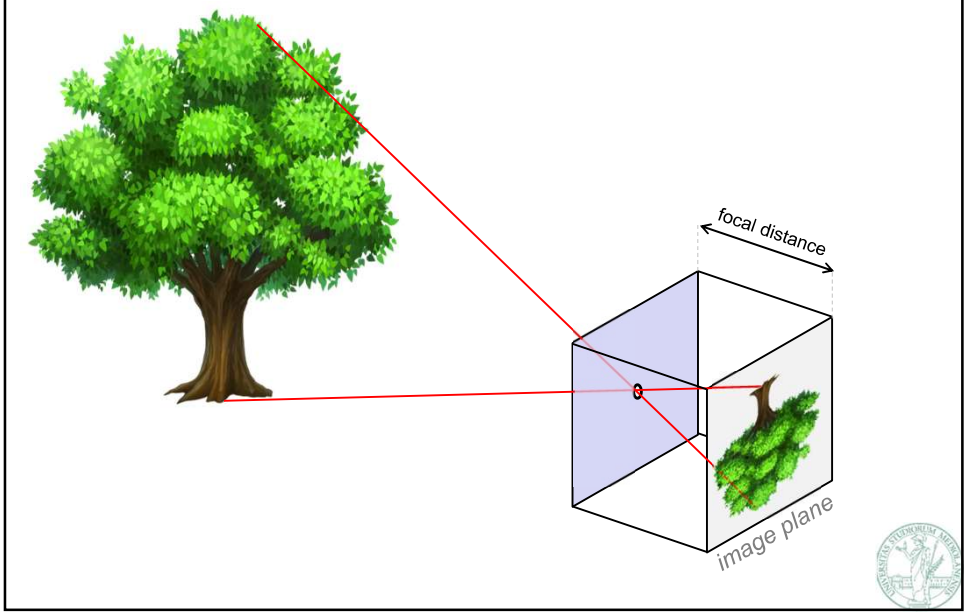
6

Pin-Hole camera



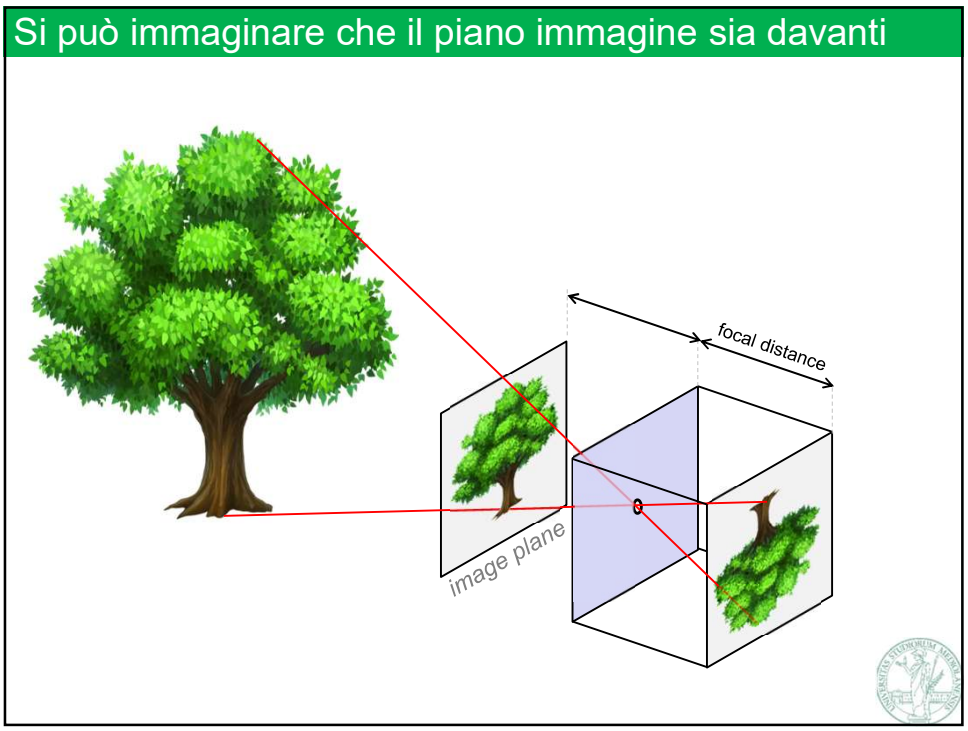
7

Pin Hole camera in azione



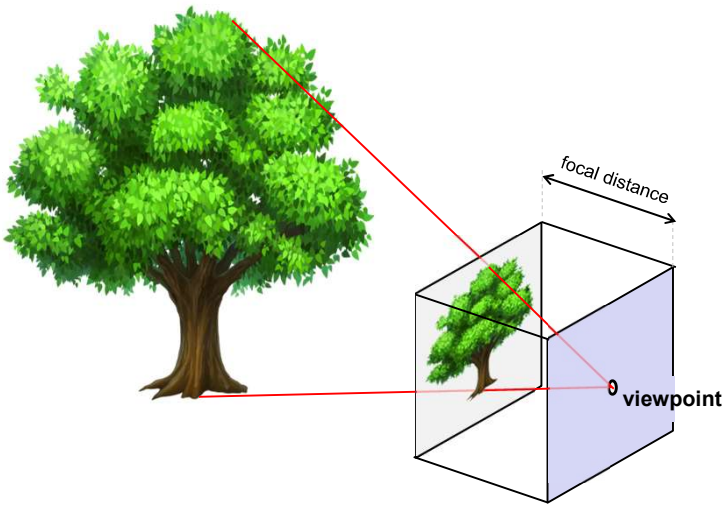
8

Si può immaginare che il piano immagine sia davanti



9

Pin Hole camera (variante)



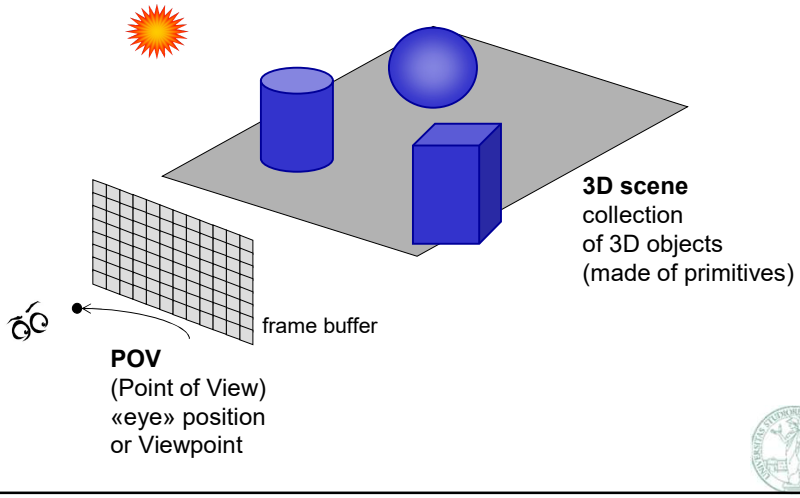
10

Pin-hole camera

- ✓La pin-hole camera cattura tutti 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
- ✓La luce è partita da degli emettitori (fonti di luce)
  - ⇒prima di arrivare al POV, ha interagito con la scena!

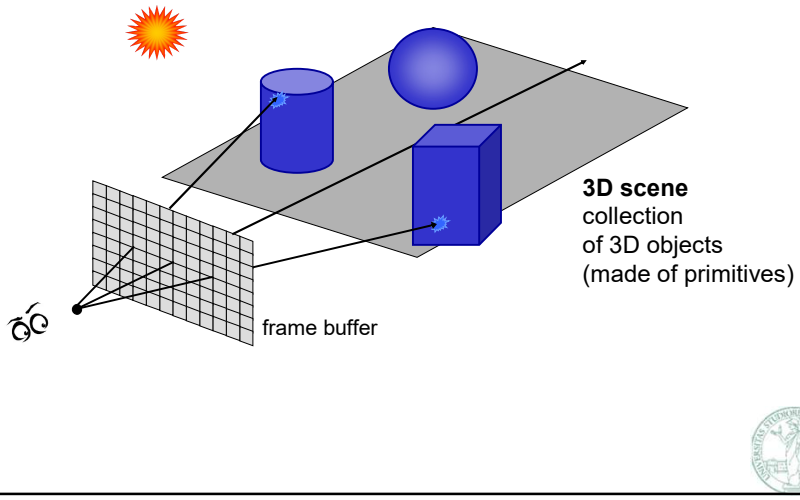
11

Ray casting



15

Ray casting



16

Ray-Casting

- ✓ Idea: seguire *a ritroso* i fotoni che raggiungono il POV
  - ⇒ per questo, detto anche :*"backwards" ray tracing*
- ✓ per ogni pixel sullo schermo:
  - ⇒ 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:
  - ⇒ basata sul computo dell'intersezione raggio-primitive (che quindi deve essere ottimizzata)
  - ⇒ Numero di intersezioni da calcolare = numero di pixe

17

Ray Casting pseudo-code

```
For each pixel p:
  make a ray r (viewpoint 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

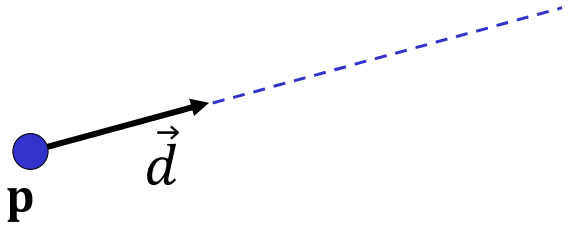
Cosa possiamo renderizzare

- ✓ Con un algoritmo di Ray-casting possiamo renderizzare qualsiasi cosa per la quale siamo 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, serve di sapere implementare la procedura di intersezione raggio-oggetto
  - ⇒ Vediamo alcuni casi per sempici modelli impliciti

19

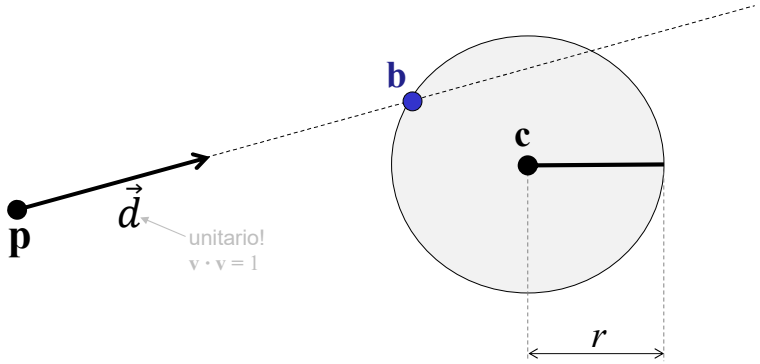
Rappresentazioni di un «raggio»

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



20

Esempio: Intersezione raggio sfera 1/3



sfera (come superficie implicita):

$$f(\mathbf{q}) = \mathbf{0} \quad \text{con} \quad f(\mathbf{q}) = \|\mathbf{q} - \mathbf{c}\|^2 - r^2$$

intersecare raggio con sfera = trovare un $k \geq 0$ (se ∃) t.c.

$$f(\mathbf{p} + k\vec{d}) = \mathbf{0}$$

21

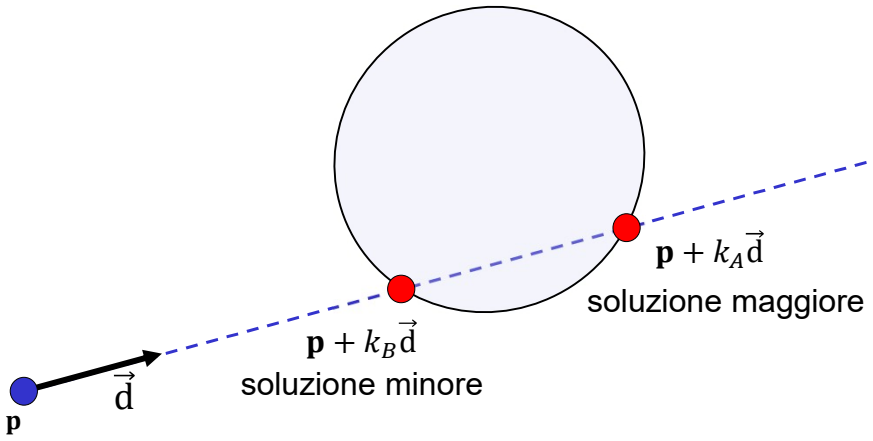
Esempio: Intersezione raggio sfera 2/3

$$f(\mathbf{p} + k\vec{d}) = 0$$
$$\Leftrightarrow$$
$$\|\mathbf{p} + k\vec{d} - \mathbf{c}\|^2 - r^2 = 0$$
$$\Leftrightarrow \quad \|\vec{v}\|^2 = \vec{v} \cdot \vec{v}$$
$$(\mathbf{p} + k\vec{d} - \mathbf{c}) \cdot (\mathbf{p} + k\vec{d} - \mathbf{c}) - r^2 = 0$$
$$\Leftrightarrow$$
$$((\mathbf{p} - \mathbf{c}) + k\vec{d}) \cdot ((\mathbf{p} - \mathbf{c}) + k\vec{d}) - r^2 = 0$$
$$\Leftrightarrow$$
$$k^2 \|\vec{d}\|^2 + k\, 2\,(\mathbf{p} - \mathbf{c}) \cdot \vec{d} + \|\mathbf{p} - \mathbf{c}\|^2 - r^2 = 0$$

Equazione 2° grando in k Prendo la soluzione minore. Se è > 0 abbiamo trovarto k e quindi anche il punto di intersezione

22

Esempio: Intersezione raggio sfera 3/3



$$k_{A,B} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

devo ovviamente considerare la soluzione minore
(quella col segno –)

23

Esempio: Intersezione Raggio / Piano

- ✓ piano definito da: un punto sul piano $\mathbf{s}$ e la sua normale $\hat{n}$
- ✓ punti su piano: { tutti i punti $\mathbf{q}$ tali che $(\mathbf{q} - \mathbf{s}) \cdot \hat{n} = 0$ }
- ✓ intersecare quel piano con un raggio dato $(\mathbf{p}, \vec{d})$ = trovare un k (se ∃) t.c. $(\mathbf{p} + k\vec{d})$ è un punto sul piano

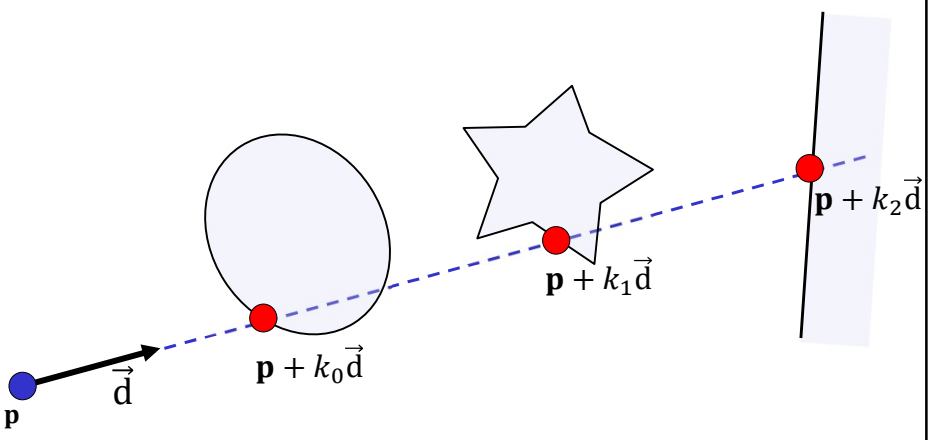
$$(\mathbf{p} + k\vec{d} - \mathbf{s}) \cdot \hat{n} = 0$$
$$\Leftrightarrow$$
$$(\mathbf{p} - \mathbf{s}) \cdot \hat{n} + k\vec{d} \cdot \hat{n} = 0$$
$$\Leftrightarrow$$
$$k\,(\vec{d} \cdot \hat{n}) = (\mathbf{s} - \mathbf{p}) \cdot \hat{n}$$
$$\Leftrightarrow$$
$$k = \frac{(\mathbf{s} - \mathbf{p}) \cdot \hat{n}}{\vec{d} \cdot \hat{n}}$$

Eq. 1mo grando in k
Se ha soluzione $k > 0$,
allora intersezione esiste.

Se $\vec{d} \cdot \hat{n} = 0$, div per zero.
Ma se è 0, allora
il raggio è ortogonale a $\hat{n}$
cioè corre parallelo
al piano: no intersez.

24

Interesezione raggio con più primitive



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

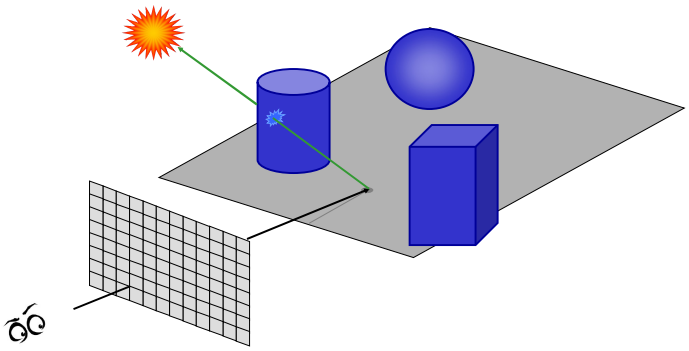
25

Ray Tracing (classe di algoritmi)

- ✓ Seguendo la stessa idea per un altro passo…
- ✓ **Tracciamo** a ritroso il percorso dei fotoni
  - ⇒ non solo dal POV ad un punto su un oggetto 3D,
  - ⇒ ma dall'oggetto 3D all'emettitore della luce
  - ⇒ seguendo vari passaggi
- ✓ Questo richiede di computare le intersezioni su altri raggi
  - ⇒ Detti raggi «secondari»
  - ⇒ Vediamo esempi per calcolo di ombre, di riflessioni, di rifrazioni (oggetti trasparenti)
  - ⇒ Nota: richiede solo di eseguire la stessa procedura, intersezione raggio primitiva, ancora altre volte per pixel.
- ✓ Algoritmi basati su tracciamento di raggi sono detti di Ray-Tracing (compreso il semplice ray-casting)

26

Ray-Tracing: ombre portate (nette)

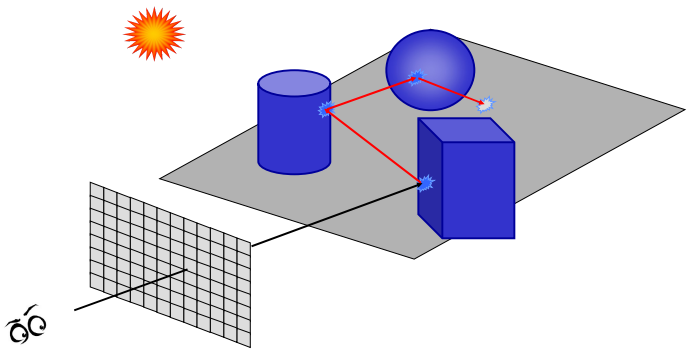


raggio primario

raggio secondario **di shadowing**
(qui: interseca, quindi il punto colpito dal raggio primario è in ombra)

27

Ray-Tracing: riflessioni speculari



raggio primario

raggi secondari di riflessione

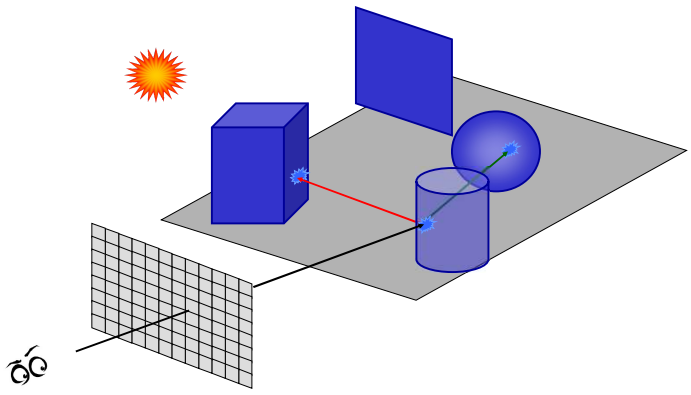
34

Come computare la direzione raggio riflesso?

- ✓Input:
  - ⇒Direzione del raggio primario
  - ⇒Normale della superficie nel punto colpito dal raggio primario
- ✓Output:
  - ⇒Direzione del raggio secondario
  - ⇒(insieme alla posizione colpita dal raggio primario, questo modella il raggio secondario di riflessione)
- ✓Vediamo la soluzione nella prossima lezione

35

Ray-Tracing: semitrasparenze con rifrazioni



36

Ray Tracing pseudo-code: secondary rays

```
For each pixel p:

  make a ray r (from viewpoint to p)

  while (…)
    for each primitive o in scene:
       find if intersect(r,o)
    keep closest intersection;
    r = new_bounce( r );

  find color for p (according to all path)
```

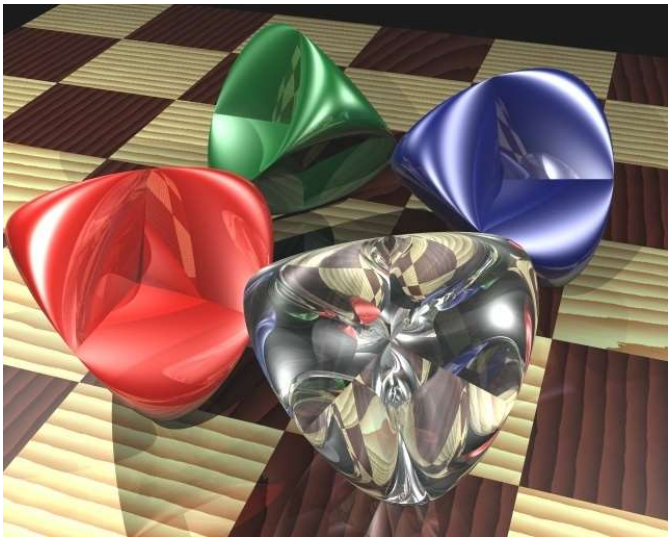
37

Ray-Tracing: tipici esempi di risultati

(Advanced Rendering Toolkit - Alexander Wilkie - 1999)

38

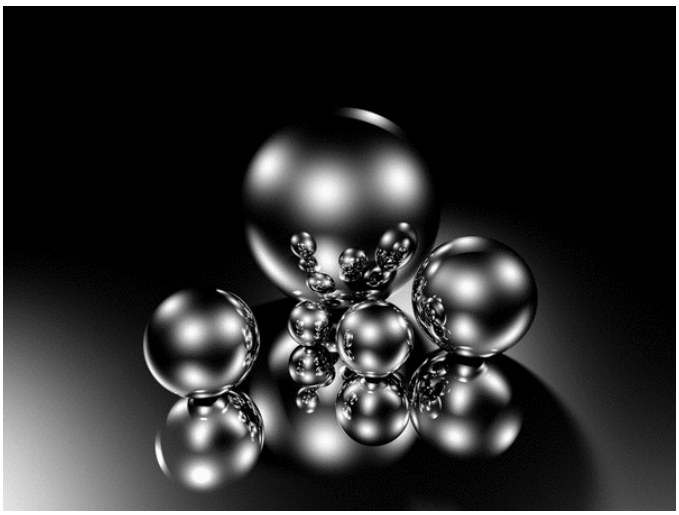
Ray-Tracing: tipici esempi di risultati



(Advanced Rendering Toolkit - Alexander Wilkie - 1999)

39

Ray-Tracing : tipici esempi di risultati



40

Algoritmi di Ray-tracing: efficienza

- ✓ Un'implementazione triviale per
  - ⇒ una scena con M oggetti («primitive»)
  - ⇒ immagini N×N pixel
  - ⇒ con K raggi per pixel (1 primario e K-1 secondari)

  richiede ben N^2 M K intersezioni raggio-primitiva!
  - ⇒ È un numero molto molto elevato
  - ⇒ Complessità temporale dell'algoritmo: $O(N^2 M K)$
  - ⇒ Per questo motivo, questo tipo di algoritmi è usato soprattutto per il **rendering offline**
- ✓ Sono possibili però ottimizzazioni…
  - ⇒ per testare solo un sottoinsieme delle primitive
  - ⇒ **Implementazioni parallelizzate** attraverso GPU (infatti ogni pixel può essere computato indipendentemete dagli altri: elevato livello di **parallelismo implicito**)

41

Esempio di RayTracing su GPU (qui: con CUDA)

42