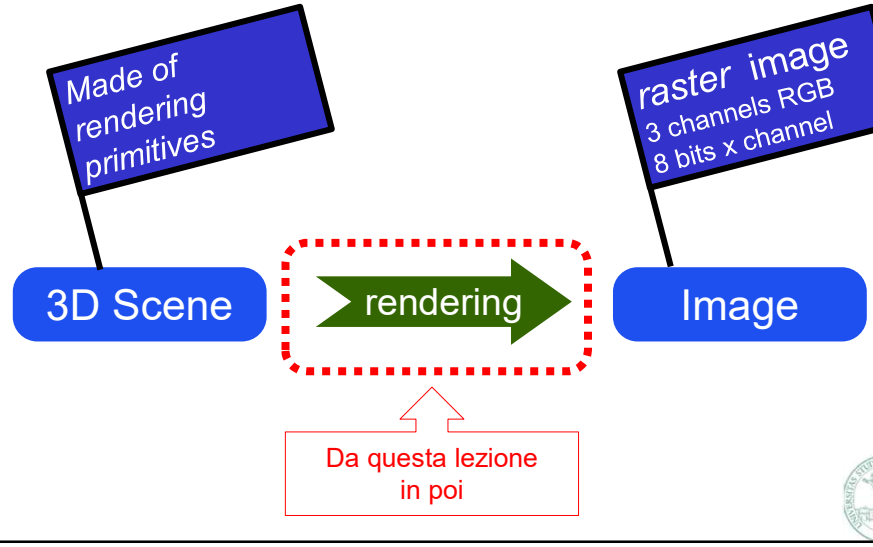
Marco Tarini - Computer Graphics 2021/2022
Università degli Studi di Milano

Rendering: ray-tracing

1

Rendering



2

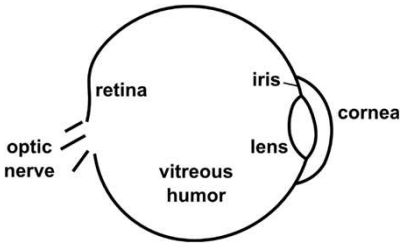
Rendering

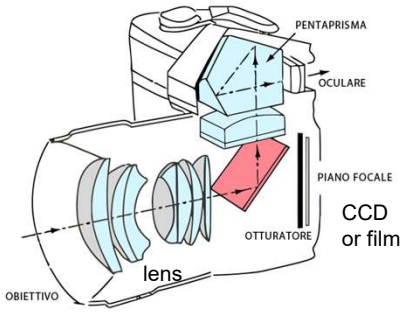
- ✓ Molti algoritmi di rendering prendono diretta ispirazione dal modo in cui un dispositivo reale di sintesi di immagini produce un'immagine
  - ⇒ Dispositivi come:
    una macchina fotografica (analogica o digitale),
    una telecamera,
    un occhio umano
- ✓ A questo fine, possiamo rimuovere tutti 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

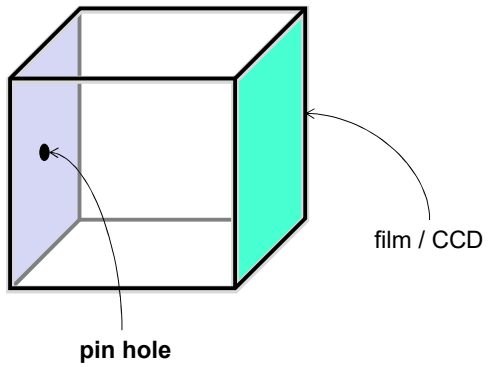
- ✓ Human Visual System
- ✓ Camera





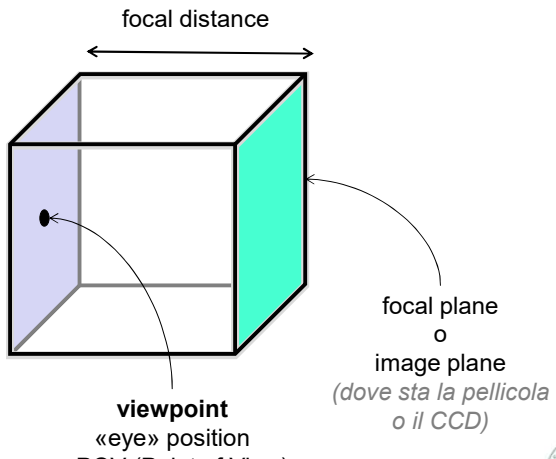
5

Pin-Hole camera



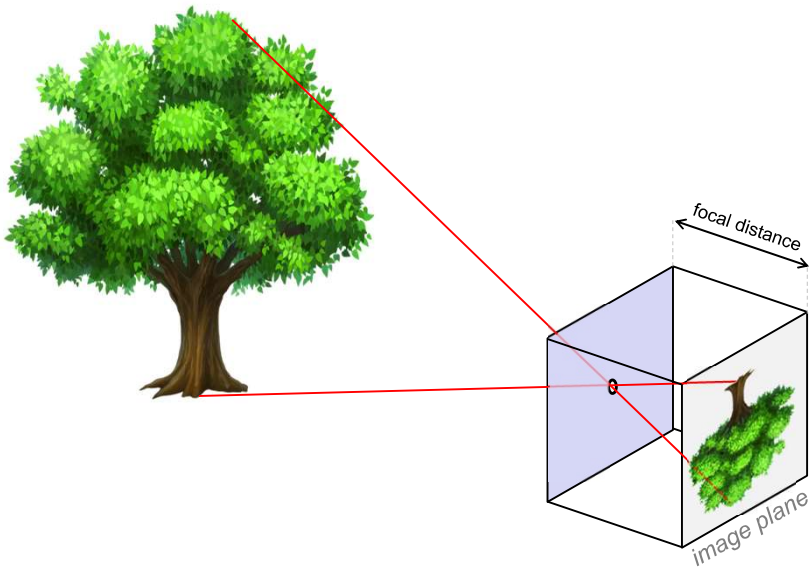
6

Pin-Hole camera



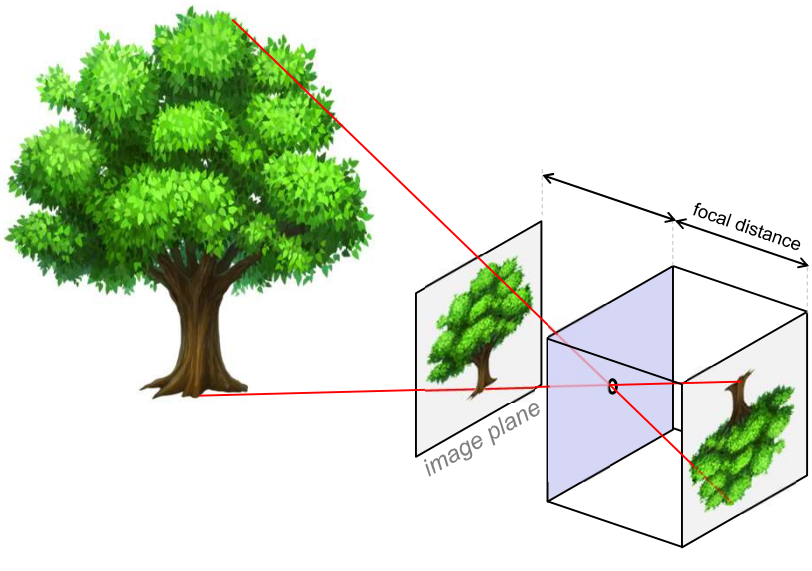
8

Pin Hole camera in azione



9

Si può immaginare che il piano immagine sia davanti



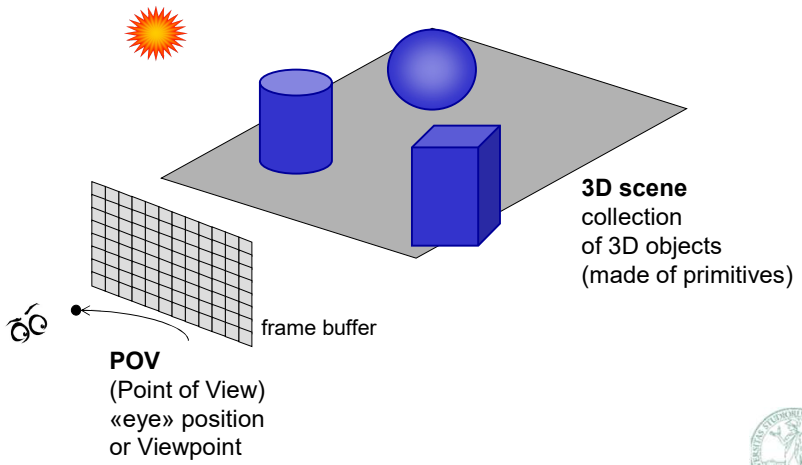
10

Pin-hole camera

- ✓La pin-hole camera cattura 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!

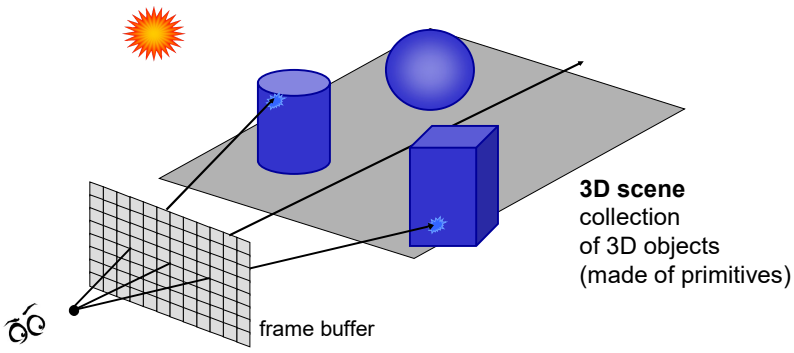
11

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

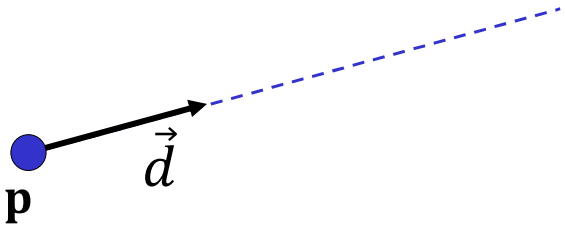
Cosa possiamo renderizzare

- ✓ 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

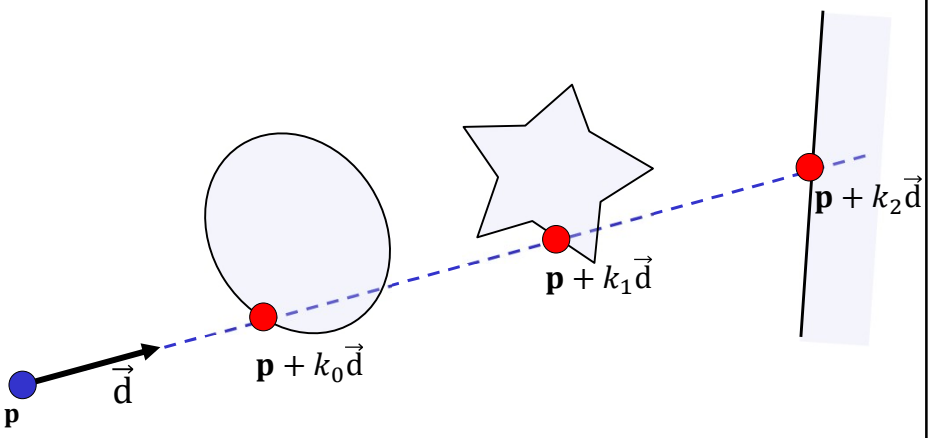
Rappresentazioni di un «raggio»

- ✓ 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

Interesezione raggio con più primitive



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

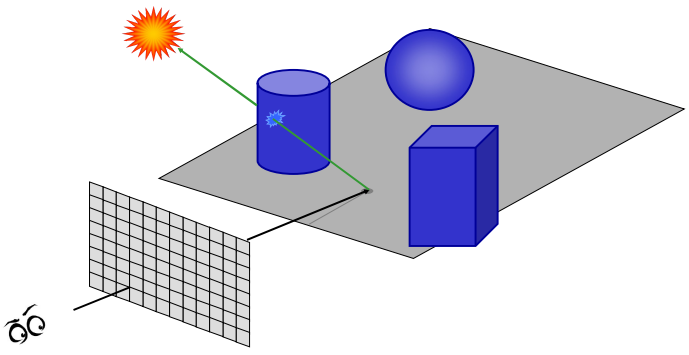
25

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 che ha emesso quel fotone
- ✓ Questo richiede di computare le intersezioni su un ulteriore raggio
  - ⇒ Detto di shadowing
  - ⇒ E' un esempio di raggio «secondario» (ne vedremo altri)
  - ⇒ Nota: è la *stessa* procedura, «intersezione raggio-primitiva», eseguita più volte per uno stesso pixel.
- ✓ Algoritmi basati su tracciamento di raggi sono detti di Ray-Tracing (compreso il semplice ray-casting)

26

Ray-Tracing: ombre portate (nette)

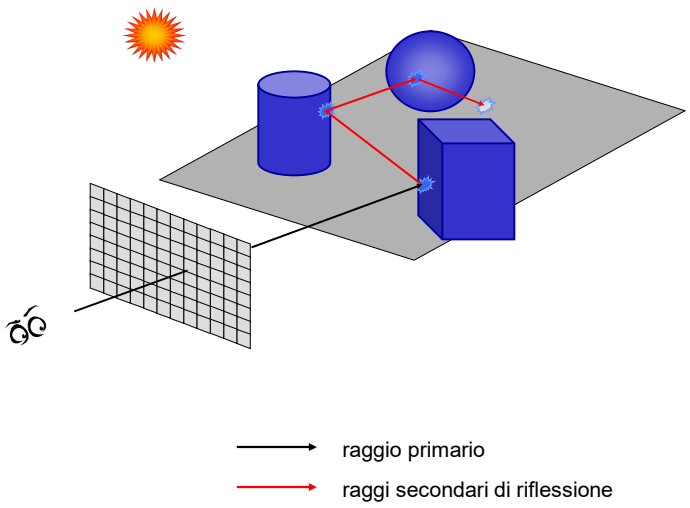


raggio primario

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

27

Ray-Tracing: riflessioni speculari



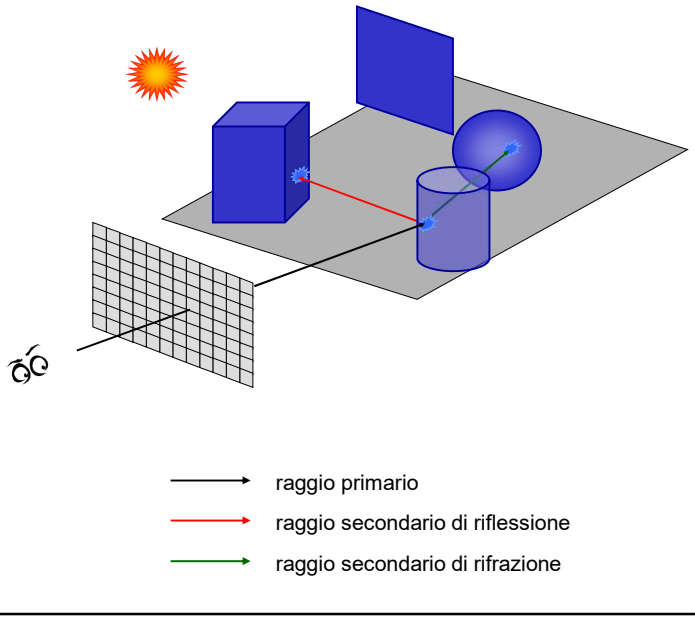
34

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)

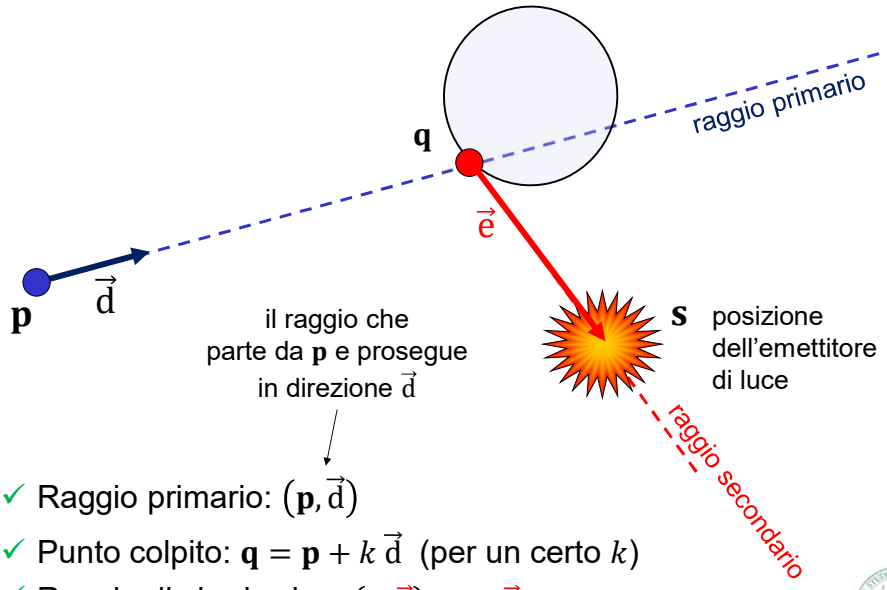
35

Ray-Tracing: semitrasparenze con rifrazioni



36

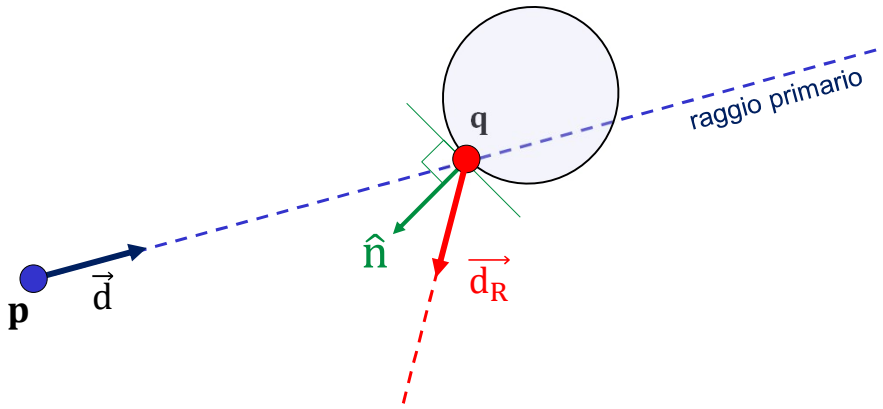
Computo del raggio secondario di shadowing



- ✓ Raggio primario: $(\mathbf{p}, \vec{\mathrm{d}})$
- ✓ Punto colpito: $\mathbf{q} = \mathbf{p} + k\,\vec{\mathrm{d}}$ (per un certo k)
- ✓ Raggio di shadowing: $(\mathbf{q}, \vec{\mathrm{e}})$, con $\vec{\mathrm{e}} = \mathbf{s} - \mathbf{q}$

37

Computo del raggio secondario di shadowing



- ✓ Raggio primario: $(\mathbf{p}, \vec{\mathrm{d}})$
- ✓ Punto colpito: $\mathbf{q} = \mathbf{p} + k\,\vec{\mathrm{d}}$ (per un certo k)
- ✓ Raggio di shadowing: $(\mathbf{q}, \overrightarrow{\mathrm{d_R}})$, con $\overrightarrow{\mathrm{d_R}}$ il vettore $\vec{\mathrm{d}}$ riflesso da un piano di normale $\hat{\mathrm{n}}$

Normale della superficie in $\mathbf{q}$

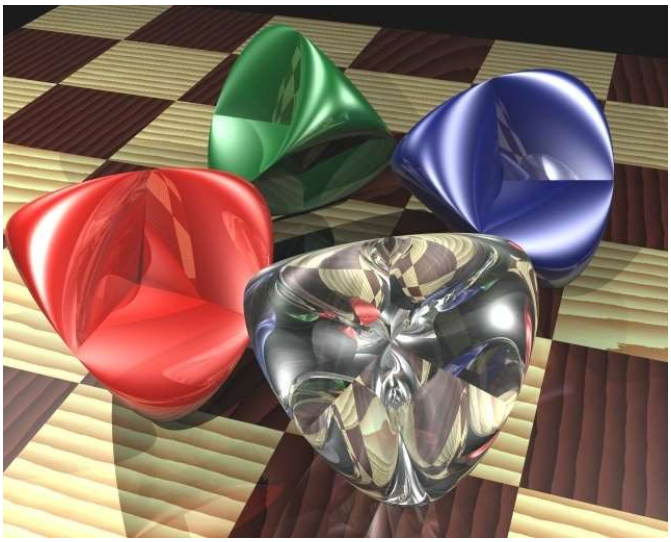
38

Ray-Tracing: tipici esempi di risultati

(Advanced Rendering Toolkit - Alexander Wilkie - 1999)

40

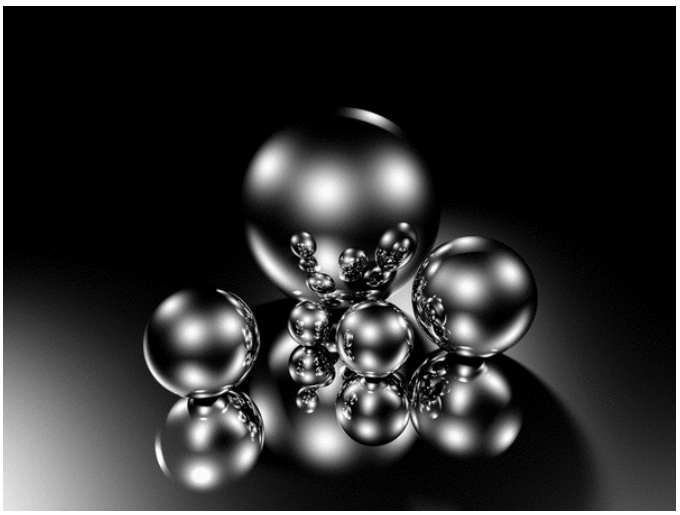
Ray-Tracing: tipici esempi di risultati



(Advanced Rendering Toolkit - Alexander Wilkie - 1999)

41

Ray-Tracing : tipici esempi di risultati



42