Marco Tarini - Computer Graphics 2023/2024
Università degli Studi di Milano

Approcci al rendering 1/2: ray-tracing

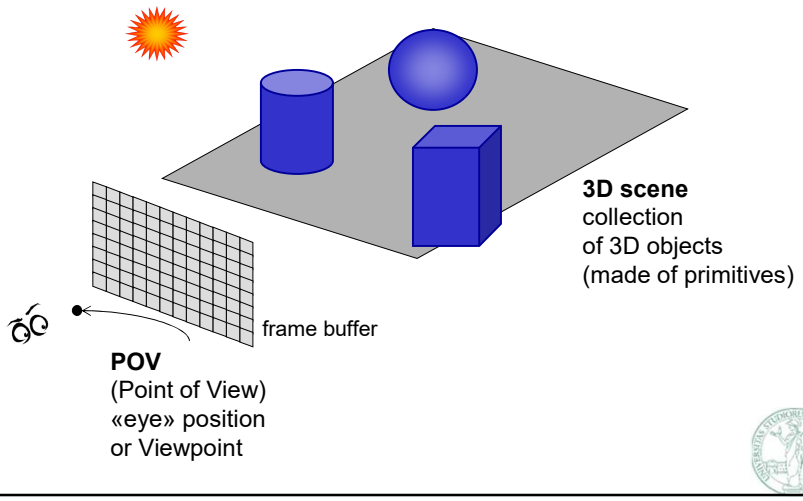
16

Ray-Casting

- ✓ Idea: seguire *a ritroso* i fotoni che raggiungono il POV
  - ⇒ per questo, detto anche : *"backward" ray tracing*
- ✓ per ogni pixel sul'immagine da produrre:
  - ⇒ produco un raggio che parte dal POV e attraversa quell pixel (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)

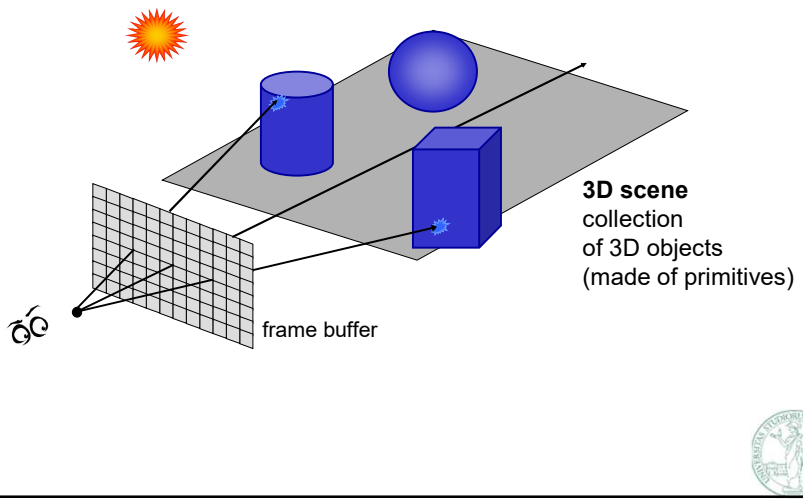
17

Ray casting



18

Ray casting



19

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
```

20

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
- ✓ Per 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!)
  - ⇒ A questa considerazione si deve la predominanza delle tri-mesh come modelli 3D

21

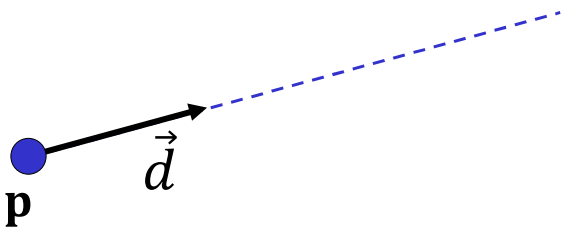
Primitive di rendering, nel caso del Ray-casting

- ✓ Con un algoritmo di Ray-casting possiamo renderizzare qualsiasi cosa per la quale siamo in grado di computare l'intersezione con un raggio
- ✓ Esistono quindi ray-caster per il rendering di :
  - ⇒ Mesh triangolari (di gran lunga il caso più rilevante, industrialmente)
  - ⇒ 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
  - ⇒ Vediamo alcuni casi per sempici modelli impliciti

22

Rappresentazione di un «raggio»

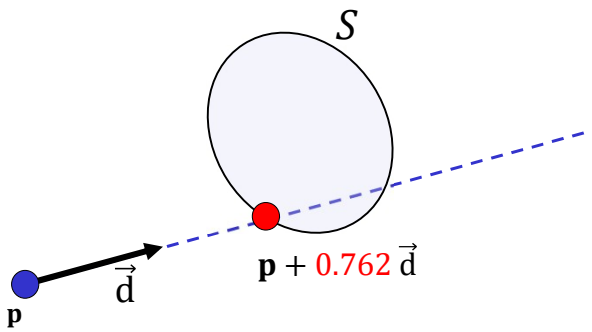
- ✓ Un raggio = una semiretta (nota, in inglese: «ray» non «radius»)
- ✓ Rappresentabile come
  - ⇒ Un dato punto $\mathbf{p}$ di partenza del raggio
  - ⇒ Un dato vettore $\vec{d}$ direzione (unitario o no, è una scelta implementativa)
- ✓ Il raggio è dato da tutti i punti esprimibili come $\mathbf{p} + k\,\vec{\mathrm{d}}$
  - ⇒ con k scalare non-negativo, $k \geq 0$ (perché il raggio è una ***semi***-retta)
  - ⇒ cioè «i punti raggiungibili a partire da $\mathbf{p}$ facendo k passi in direzione $\vec{\mathrm{d}}$ »



24

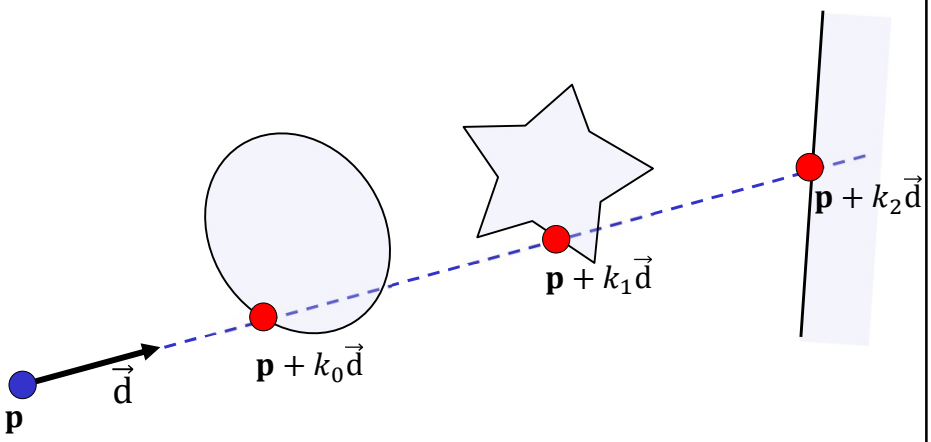
Intersezione raggio – primitiva

- ✓ Trovare l'intersezione con una data superficie S significa trovare uno scalare k tale che $\mathbf{p} + k\vec{\mathrm{d}}$ stia su S



25

Interesezione raggio-scena (cioè tante primitive)



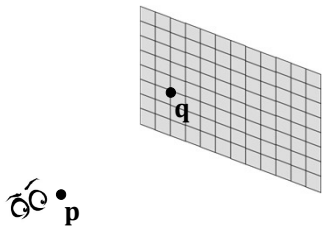
Basta prendere la primitiva che interseca col k **minore** fra $k_{0,1,2}$

26

Costruzione del raggio primario

- ✓ Dato un pixel Q, quale è il suo raggio primario?
- ✓ Risposta:
  - ⇒ $\mathbf{p}$ (punto di partenza del raggio): il POV
  - ⇒ $\vec{d}$ (direzione del raggio): $(\mathbf{q} - \mathbf{p})$
    (eventualmente, normalizzato, cioè scalato di $\frac{1}{\|\mathbf{q}-\mathbf{p}\|}$)

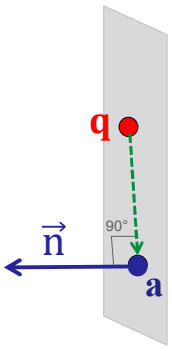
    dove $\mathbf{q}$ è la posizione 3D del pixel Q sul piano immagine



30

Esempio: Intersezione Raggio – Piano 1/3

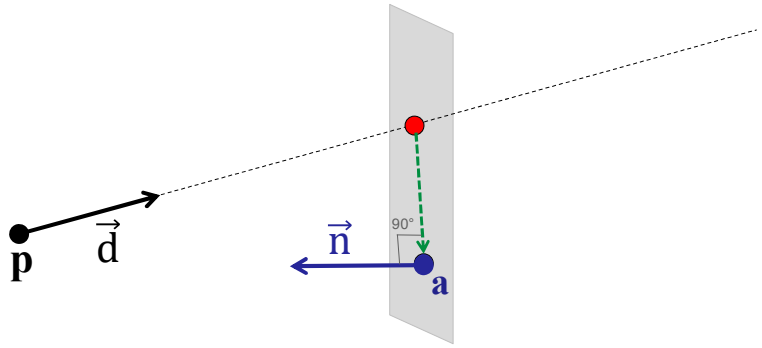
- ✓ piano definito da: un punto su di esso $\mathbf{a}$ e il suo vettore normale $\vec{n}$ (l'orientamento del piano)
- ✓ punti su piano: cioè tutti i punti $\mathbf{q}$ tali che il vettore $(\mathbf{a} - \mathbf{q})$ è ortogonale ad $\vec{n}$, cioè $(\mathbf{a} - \mathbf{q}) \cdot \vec{n} = 0$
- ✓ intersecare il piano con un raggio dato $(\mathbf{p}, \vec{d})$ = trovare un k (se ∃) t.c. $(\mathbf{p} + k\vec{d})$ sia un punto sul piano



Piano passante per il punto **a** e avente normale $\vec{n}$

32

Esempio: Intersezione raggio piano 2/3



intersecare raggio con sfera = trovare un $k \geq 0$ (se $\exists$) tale che

Il vettore verde

$$\left(\mathbf{a} - \left(\mathbf{p} + k\vec{\mathrm{d}}\right)\right) \cdot \vec{\mathrm{n}} = 0$$

Il punto rosso

perché so che il vettore verde deve essere ortogonale ad $\vec{\mathrm{n}}$

33

Esempio: Intersezione raggio piano 3/3

Trovo k (che è l'unica incognita) manipolando l'equazione

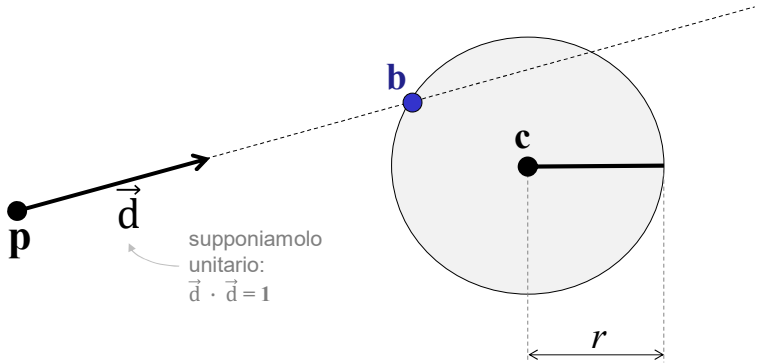
$$\left(\mathbf{a} - \left(\mathbf{p} + k\vec{\mathrm{d}}\right)\right) \cdot \vec{\mathrm{n}} = 0$$
$$\Leftrightarrow$$
$$\left((\mathbf{a} - \mathbf{p}) - k\vec{\mathrm{d}}\right) \cdot \vec{\mathrm{n}} = 0$$
$$\Leftrightarrow$$
$$(\mathbf{a} - \mathbf{p}) \cdot \vec{\mathrm{n}} - k\left(\vec{\mathrm{d}} \cdot \vec{\mathrm{n}}\right) = 0$$
$$\Leftrightarrow$$
$$k = \frac{(\mathbf{a} - \mathbf{p}) \cdot \vec{\mathrm{n}}}{\vec{\mathrm{d}} \cdot \vec{\mathrm{n}}}$$

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

Se $k > 0$, allora il raggio interseca il piano, e l'intersezione è alla posizione $\mathbf{a} + k\vec{\mathrm{d}}$
altrimenti, il raggio non interseca il piano

34

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

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

35

Esempio: Intersezione raggio sfera 2/3

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

$$\Leftrightarrow$$

$$\left\|\mathbf{p} + k\vec{d} - \mathbf{c}\right\|^2 - r^2 = 0$$

$$\Leftrightarrow$$

$$\left\|(\mathbf{p} - \mathbf{c}) + k\vec{d}\right\|^2 - r^2 = 0$$

$$\Leftrightarrow \quad \|\vec{v}\|^2 = \vec{v} \cdot \vec{v}$$

$$\left((\mathbf{p} - \mathbf{c}) + k\vec{d}\right) \cdot \left((\mathbf{p} - \mathbf{c}) + k\vec{d}\right) - r^2 = 0$$

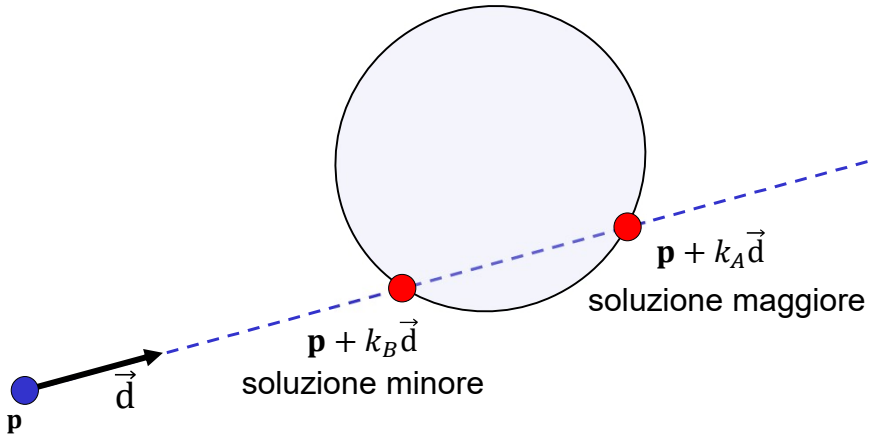
$$\Leftrightarrow$$

$$k^2 \underbrace{\left\|\vec{d}\right\|^2}_{a} + k \underbrace{2\,(\mathbf{p} - \mathbf{c}) \cdot \vec{d}}_{b} + \underbrace{\|\mathbf{p} - \mathbf{c}\|^2 - r^2}_{c} = 0$$

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

36

Esempio: Intersezione raggio sfera 3/3



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

devo ovviamente considerare la soluzione minore
(quella scegliendo il segno meno nel $\pm$)

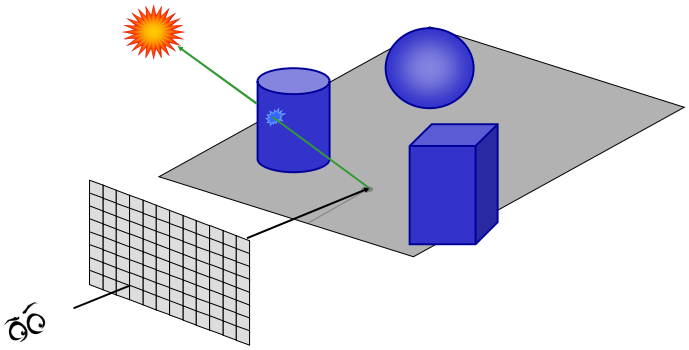
37

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)

38

Ray-Tracing: ombre portate (nette)

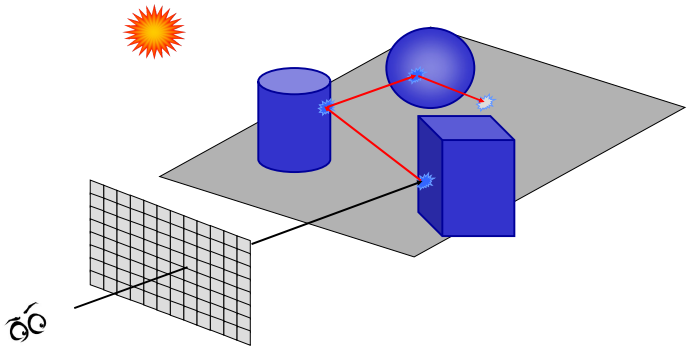


raggio primario

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

39

Ray-Tracing: riflessioni speculari

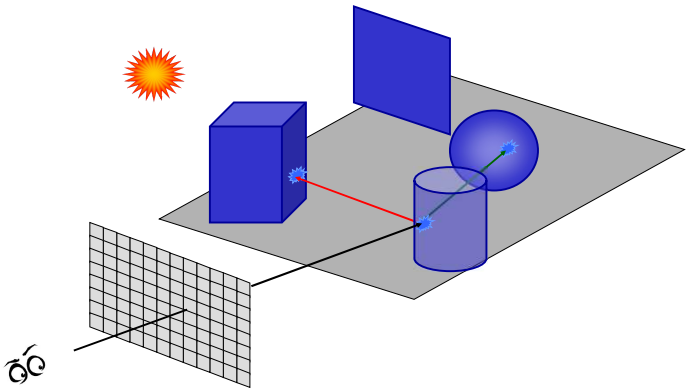


raggio primario

raggi secondari di riflessione

40

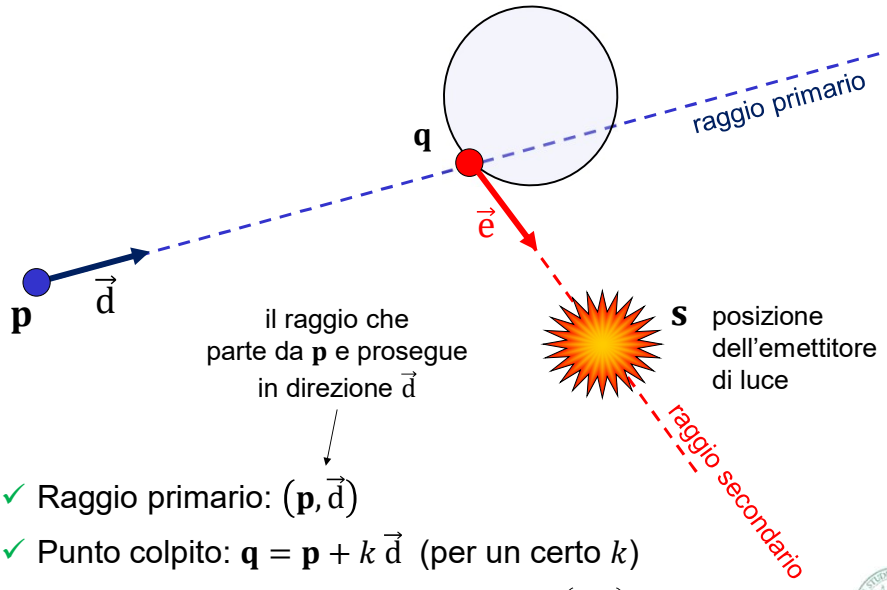
Ray-Tracing: semitrasparenze con rifrazioni



raggio primario
raggio secondario di riflessione
raggio secondario di rifrazione

41

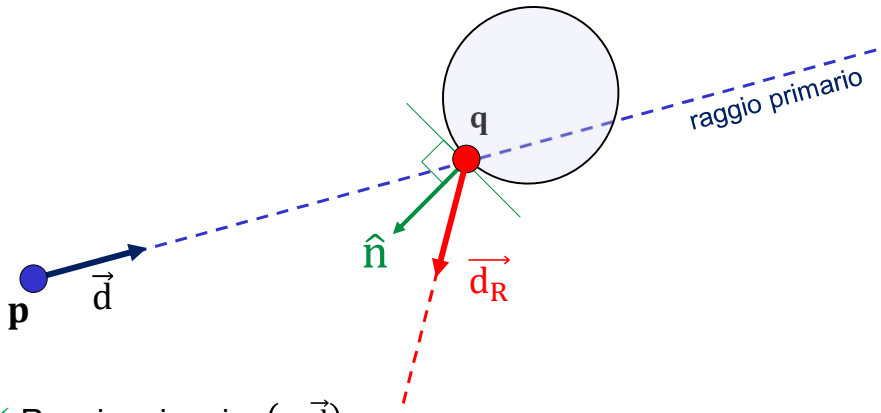
Computo del raggio secondario di shadowing



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

42

Computo del raggio secondario di riflessione

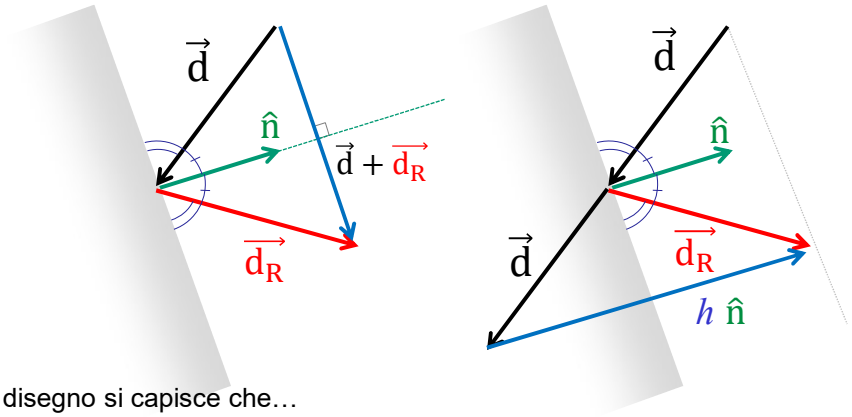


- ✓ Raggio primario: $(\mathbf{p}, \vec{d})$
- ✓ Punto colpito: $\mathbf{q} = \mathbf{p} + k\,\vec{d}$ (per un certo k)
- ✓ Raggio di riflessione: $(\mathbf{q}, \vec{d}_R)$, dove $\vec{d}_R$ è il vettore $\vec{d}$ riflesso da un piano di normale $\hat{n}$
- ✓ Vediamo come calcolarlo (semplice "ottica meccanica")

Normale della superficie in $\mathbf{q}$

43

Calcolo della direzione riflessa $\vec{d}_R$ (1/2)



Dal disegno si capisce che…

(1) Il vettore $\vec{d} + \vec{d}_R$ è ortogonale ad $\hat{n}$
Cioè:

$$(\vec{d} + \vec{d}_R) \cdot \hat{n} = 0$$

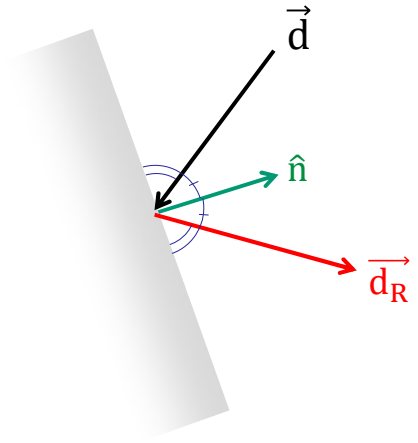
(2) Il vettore $\vec{d}_R$ è ottenibile sommando a $\vec{d}$ un vettore allineato ad $\hat{n}$
Cioè:

$$\vec{d}_R = \vec{d} + h\,\hat{n}$$

(per un dato scalare h, da trovare)

45

Calcolo della direzione riflessa $\overrightarrow{d_R}$ (2/2)



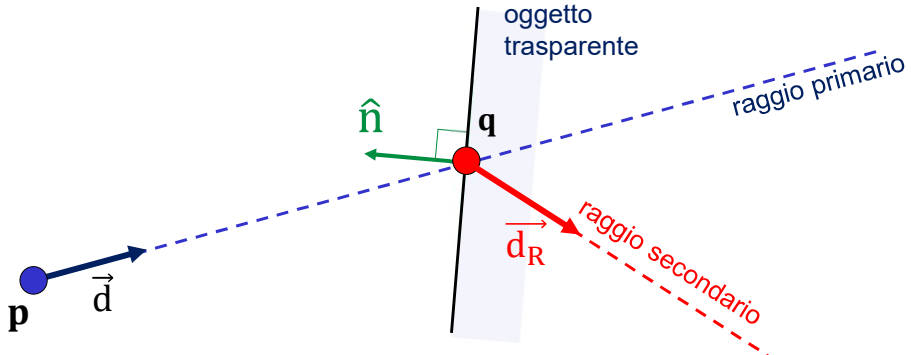
Sostituedo, trovo

$\overrightarrow{d_R} = \vec{d} + h\,\hat{n}$

con $h = -2\,\vec{d} \cdot \hat{n}$

46

Computo del raggio secondario di rifrazione

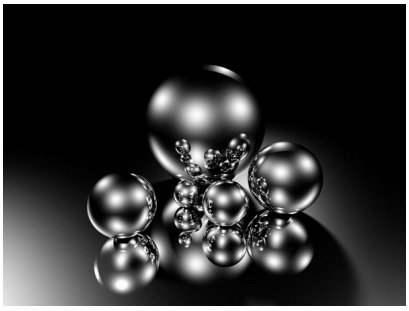


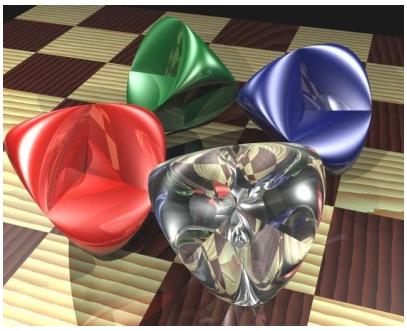
- ✓ Raggio primario: $(\mathbf{p}, \vec{d})$
- ✓ Punto colpito: $\mathbf{q} = \mathbf{p} + k\,\vec{d}$ (per un certo k)
- ✓ Raggio di rifrazione: $(\mathbf{q}, \overrightarrow{d_F})$,
  con $\overrightarrow{d_F}$ il vettore $\vec{d}$ rifratto da un piano di normale $\hat{n}$

  Una formula di meccanica ottica, (che non vediamo) descrive come trovare la direzione rifratta $\overrightarrow{d_F}$ dato l'«*indice di refrazione*» del materiale

47

Ray-Tracing : tipici esempi di risultati






(Advanced Rendering Toolkit - Alexander Wilkie - 1999)

(with CUDA)

52

Algoritmi di Ray-tracing: efficienza

- ✓ Un'implementazione **triviale** per
  - ⇒ una scena con M «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 troppo 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 molte ottimizzazioni…
  - ⇒ per testare solo un sottoinsieme delle primitive
  - ⇒ **Implementazioni parallelizzate** attraverso GPU (infatti ogni pixel può essere computato indipendentemente dagli altri: elevato livello di **parallelismo implicito**)

53

Esempio di RayTracing su GPU (qui: con CUDA)

54

Ray tracing: alcune varianti

Nota: questa terminologia può variare leggermente da una fonte all'altra

- ✓ Molti algoritmi e varianti si basano su principi simili.
- ✓ Alcuni di questi:
  - ⇒ Ray-casting:
    - un solo raggio (primario) viene "mandato" (*cast*) per pixel
  - ⇒ Ray-tracing:
    - Ogni raggio viene "tracciato" (*traced*) attraverso vari fenomeni successivi riflessione o rifrazione, attraverso raggi secondari
    - Nota: in ordine inverso rispetto alla realtà, come al solito
    - (è anche il usato come nome della classe generale di questi algoritmi)
  - ⇒ Path-tracing:
    - Per ogni pixel, lanciare molti raggi, che seguono i path pseudo-casuali ("monte carlo sampling") con una distribuzione che riflette quella reale
    - Mediare i risultati per ottenere il pixel finale
    - Più raggi vengono mandati, maggiore il realismo
  - ⇒ Ray-marching:
    - Una classe di algoritmi per accelerare l'intersezione raggio-primitiva
    - Principio: raggio viene spezzato in una serie di passi (segmenti) successivi, per ciascuno dei quali si devono testare solo le primitive a lui vicine (invece di tutte)

55

Esempi: Ray tracing su alcuni tipi di modelli visti

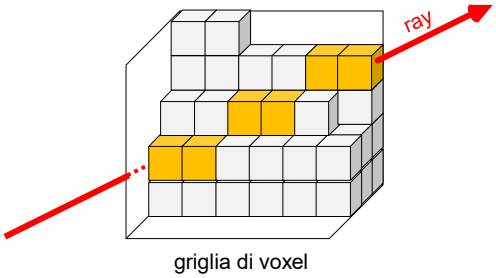
✓ Raytracing di volume di voxel:
(con ciascun voxel = valore di densità)

⇒ È una forma commune di **direct volume rendering**

⇒ Per ogni raggio primario, identifico i voxel attraversati

⇒ Assegno il pixel corrispondente ad una funzione di tutti valori di intesità dei voxel attraversati

⇒ per esempio, banalmente, il tono di grigio corrispondente al valore max



griglia di voxel

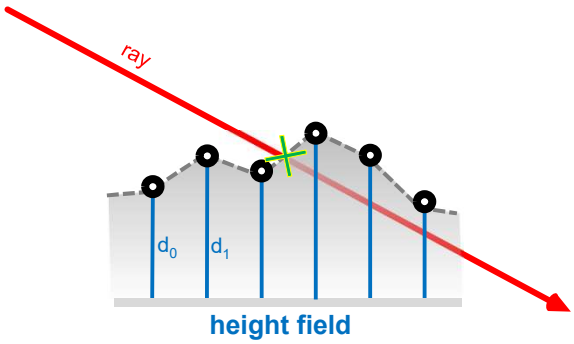
56

Esempi: Ray tracing su alcuni tipi di modelli visti

✓ Raytracing di Campi di Altezza (schema)

⇒ Tracciare il raggio sulla griglia 2D di altezze x, y

⇒ Trovare la prima posizione in cui la z del raggio cade sotto l'Altezza a coordinate $[x][y]$



height field
(vista laterale)

57

Esempi: Ray tracing su alcuni tipi di modelli visti

✓ Raytracing di CSM (schema):

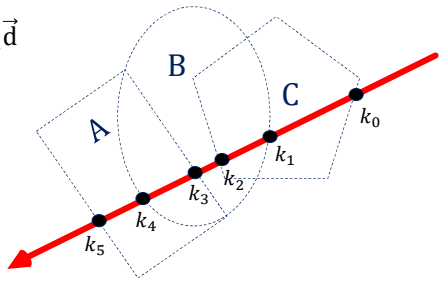
⇒ Trovare *tutte* le intersezioni $\mathbf{p} + k_i\vec{\mathrm{d}}$ con *tutte* le foglie della funz F

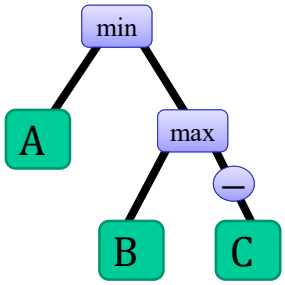
⇒ Ordinarle per k_i

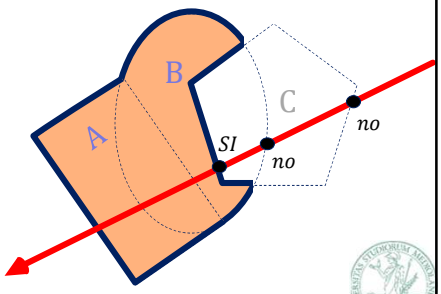
⇒ Trovare la prima intersezione in cui $F(\mathbf{p} + k_i\vec{\mathrm{d}}) = 0$

⇒ Esempio con:

$$F = \min(\mathrm{A}, \max(\mathrm{B}, -\mathrm{C}))$$







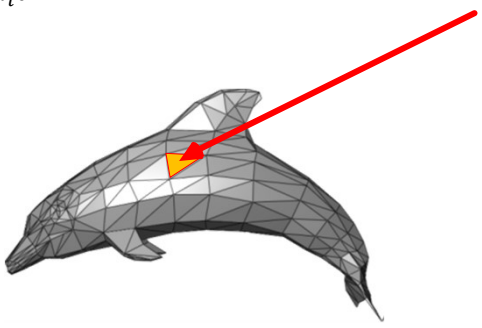
58

Esempi: Ray tracing su alcuni tipi di modelli visti

✓ Raytracing di tri-mesh

⇒ Per ogni triangolo: trovare l'intersezione $\mathbf{p} + k_i\vec{\mathrm{d}}$ raggio-triangolo, se esiste (come implementare questo calcolo?)

⇒ Prendere l'intersezione con il k_i minore



E' una soluzione molto generale, perché è sempre possibile tradurre i nostri modelli 3D in una mesh.
Ma: può essere una soluzione molto onerosa, (molte primitive sono necessarie per approssimare la forma), e la discretizzazione introduce errori

59