Marco Tarini - Computer Graphics 2025/2026
Università degli Studi di Milano

Approcci al rendering 1/2: ray-tracing

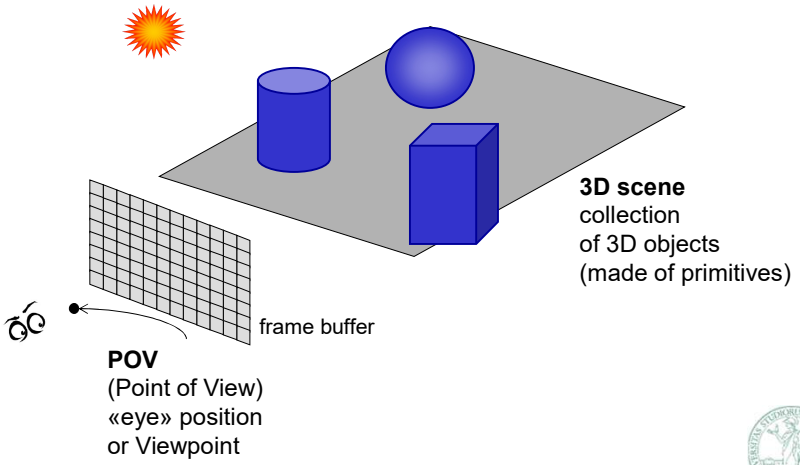
14

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:
  - ⇒ 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") (quest'operazione deve quindi deve essere ottimizzata)

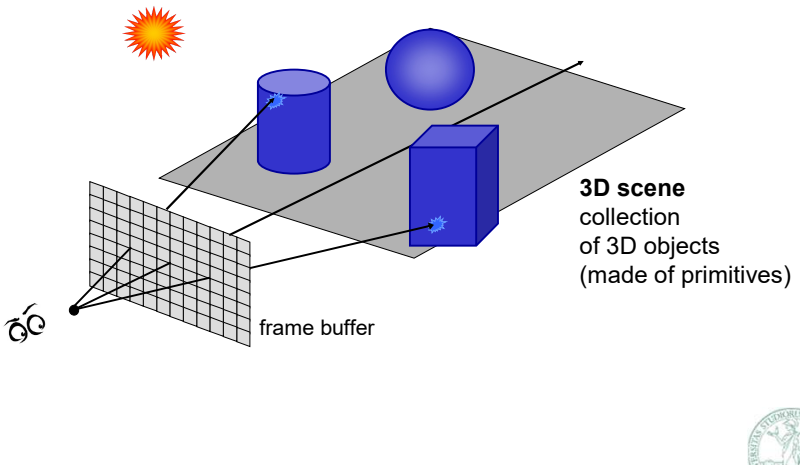
15

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
- ✓ 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

19

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 generiche (incluso quad mesh)
  - ⇒ Campi d'Altezza (che non abbiamo visto)
  - ⇒ Modelli impliciti (che non abbiamo visto)
  - ⇒ Superfici parametriche, come patch di Bezier
  - ⇒ Etc
- ✓ In ciascun primitiva, è necessario implementare la procedura di intersezione raggio-primitiva
  - ⇒ Vediamo alcuni casi per sempici modelli impliciti

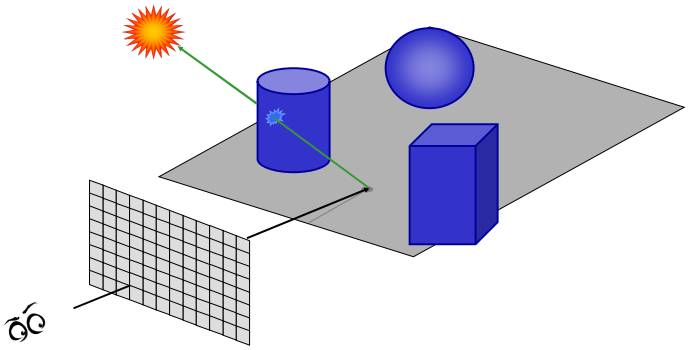
20

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)

22

Ray-Tracing: ombre portate (nette)

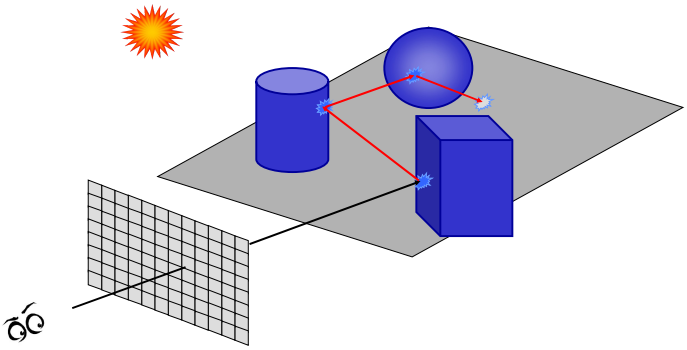


raggio primario

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

23

Ray-Tracing: riflessioni speculari

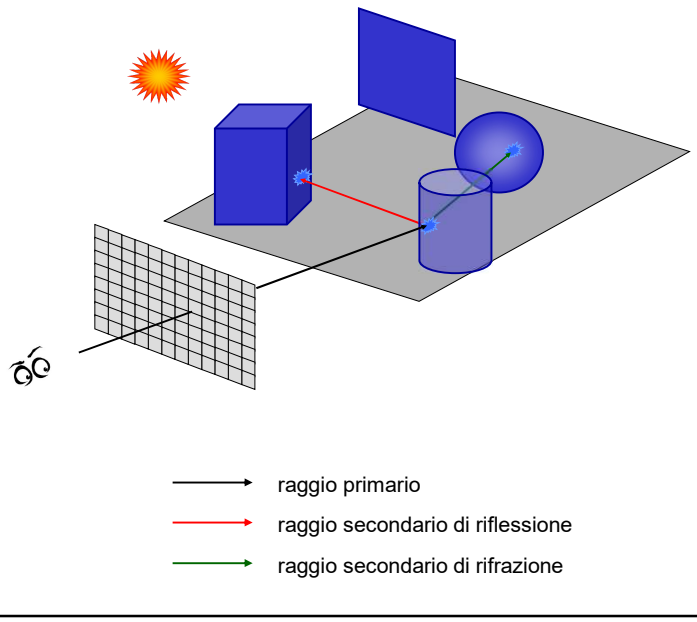


raggio primario

raggi secondari di riflessione

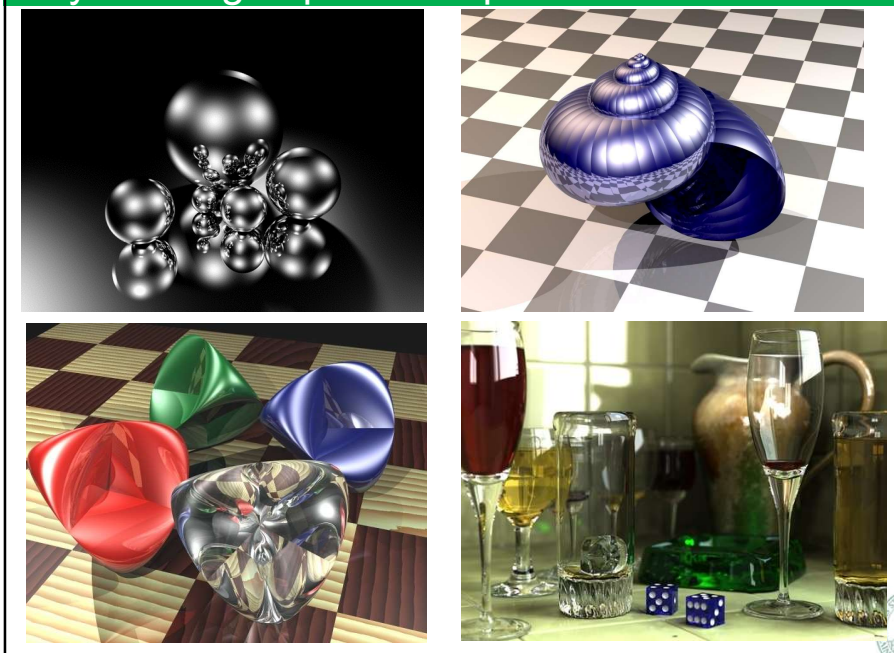
24

Ray-Tracing: semitrasparenze con rifrazioni



25

Ray-Tracing : tipici esempi di risultati



(Advanced Rendering Toolkit - Alexander Wilkie - 1999)

(with CUDA)

53

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**)

54

Ray tracing: alcune varianti

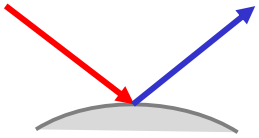
Nota: questa terminologia può variare leggermente da una fonte all'altra

- ✓ 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)

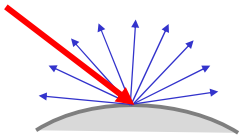
55

Modelli *semplificati* di riflessione della luce

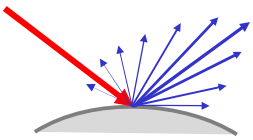
- ✓ Riflessioni speculari
  - ⇒ luce (fotoni) rimbalza proprio come una pallina da ping-pong su di un tavolo



- ✓ Riflessioni diffusive
  - ⇒ luce riflessa in ogni direzione possibile in egual misura (i fotoni vengono riflessi in tutte le direzioni)



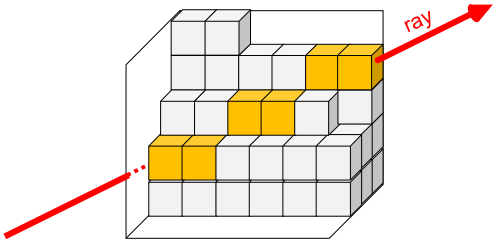
- ✓ Miste (“glossy”)
  - ⇒ luce riflessa in ogni direzione ma maggiormente in quelle simili alla direzione speculare



56

Esempi: Ray tracing su alcuni tipi di modelli visti

- ✓ Ray-tracing di volume di voxel:
  (con ciascun voxel = valore di densità)
  - ⇒ È una forma comune 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

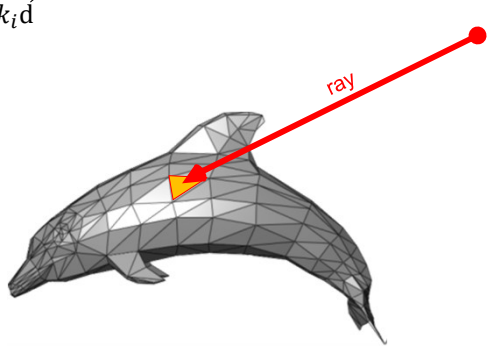


griglia di voxel

59

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

62

Nota: esempio di codice di ray-tracing

- ✓ L'algoritmo di ray-tracing, è concettualmente semplice
  - ⇒ Una versione di base (non ottimizzata, e su poche primitive semplici) può essere implementata in poche centinaia di righe di codice
- ✓ Nota «storica»: nelle edizioni precedenti di questo corso (di 9 CFU), dedicando 4-5 lezioni a questo, era stato possibile implementare in classe, da zero, un ray-tracer giocattolo
  - ⇒ In un linguaggio chiamato GLSL
- ✓ Il codice, disponibile in rete, e può essere un esempio utile:
  - ⇒ https://tarini.di.unimi.it/glsltoy/?raytracer_riflessi_materiale.glsl
  - ⇒ Il link è anche sul sito del corso
- ✓ Questo codice (che viene eseguito su ogni pixel) contiene:
  - ⇒ Computo del raggio primario per il pixel processato
  - ⇒ Computo intersezione raggio-piano
  - ⇒ Computo intersezione raggio-sfera
  - ⇒ Computo della PRIMA intersezione con le primitive (occlusione)
  - ⇒ Computo del raggio secondario di shadowing
  - ⇒ Computo del raggio secondario di riflessione

63