



UNIVERSITÀ
DI TORINO

Corso di Laurea Magistrale in Informatica

Visualizzazione Immersiva e Interattiva di Volumi Medici attraverso Ray Marching

Relatore

Grangetto Marco

Candidato

Orrù Michael

Matricola 883850

Correlatore

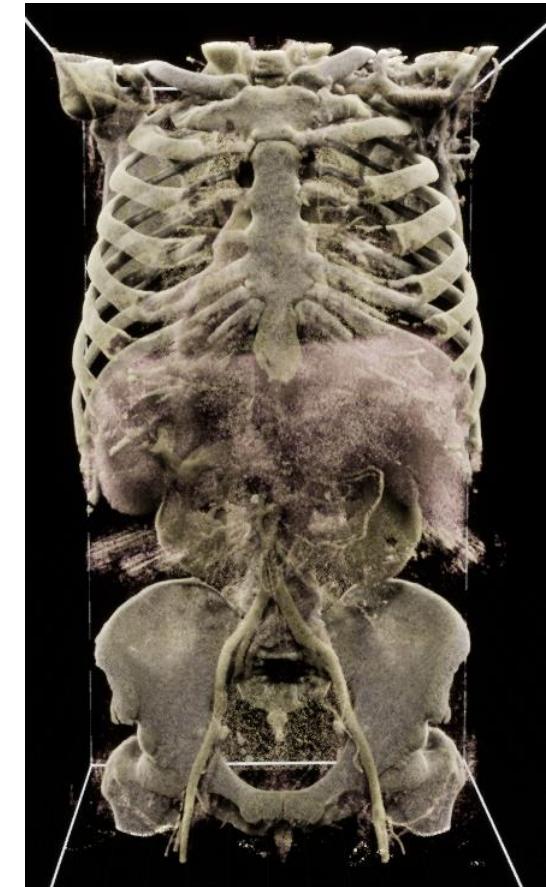
Lucenteforte Maurizio

A.A 2023/2024

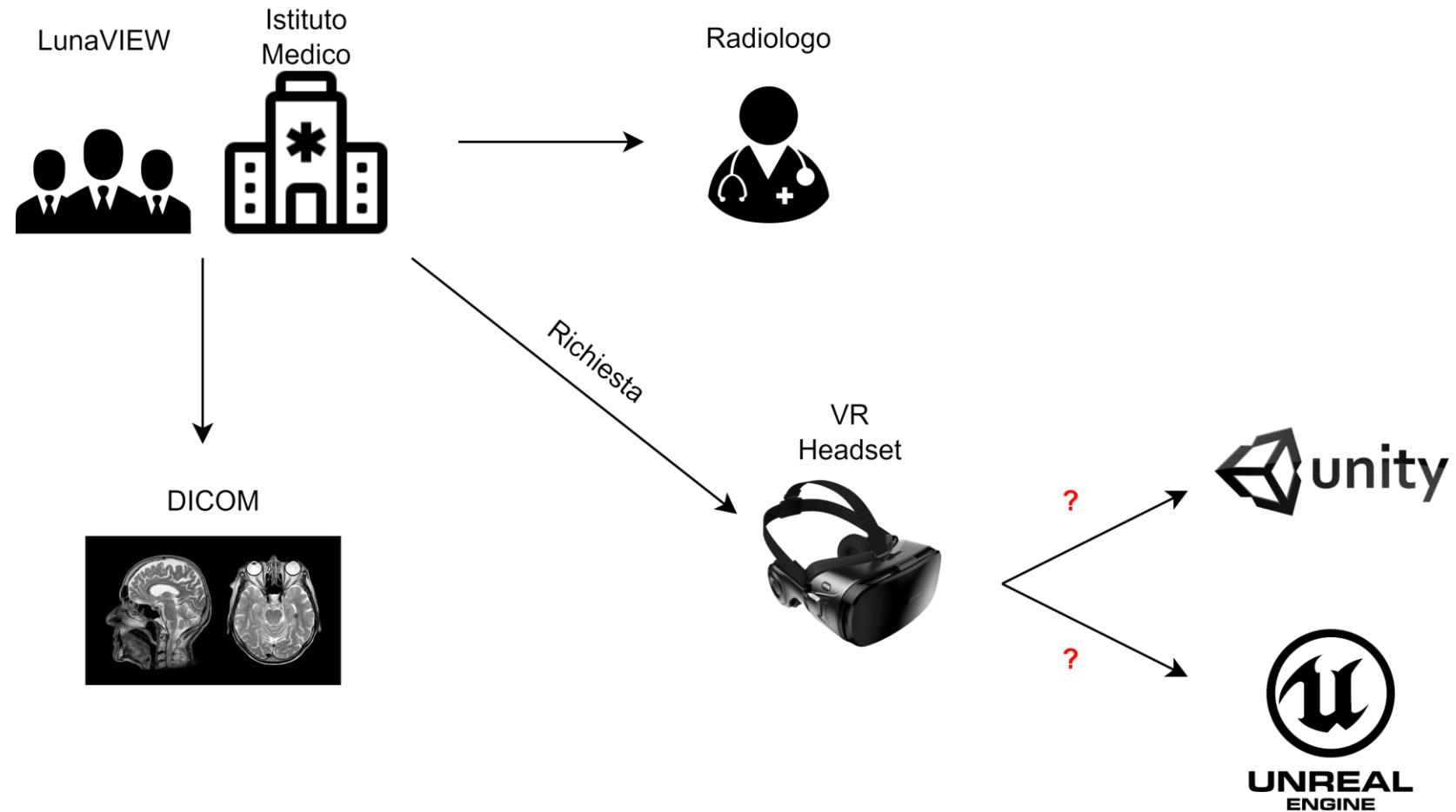
Il Progetto LunaVIEW

• Obiettivo

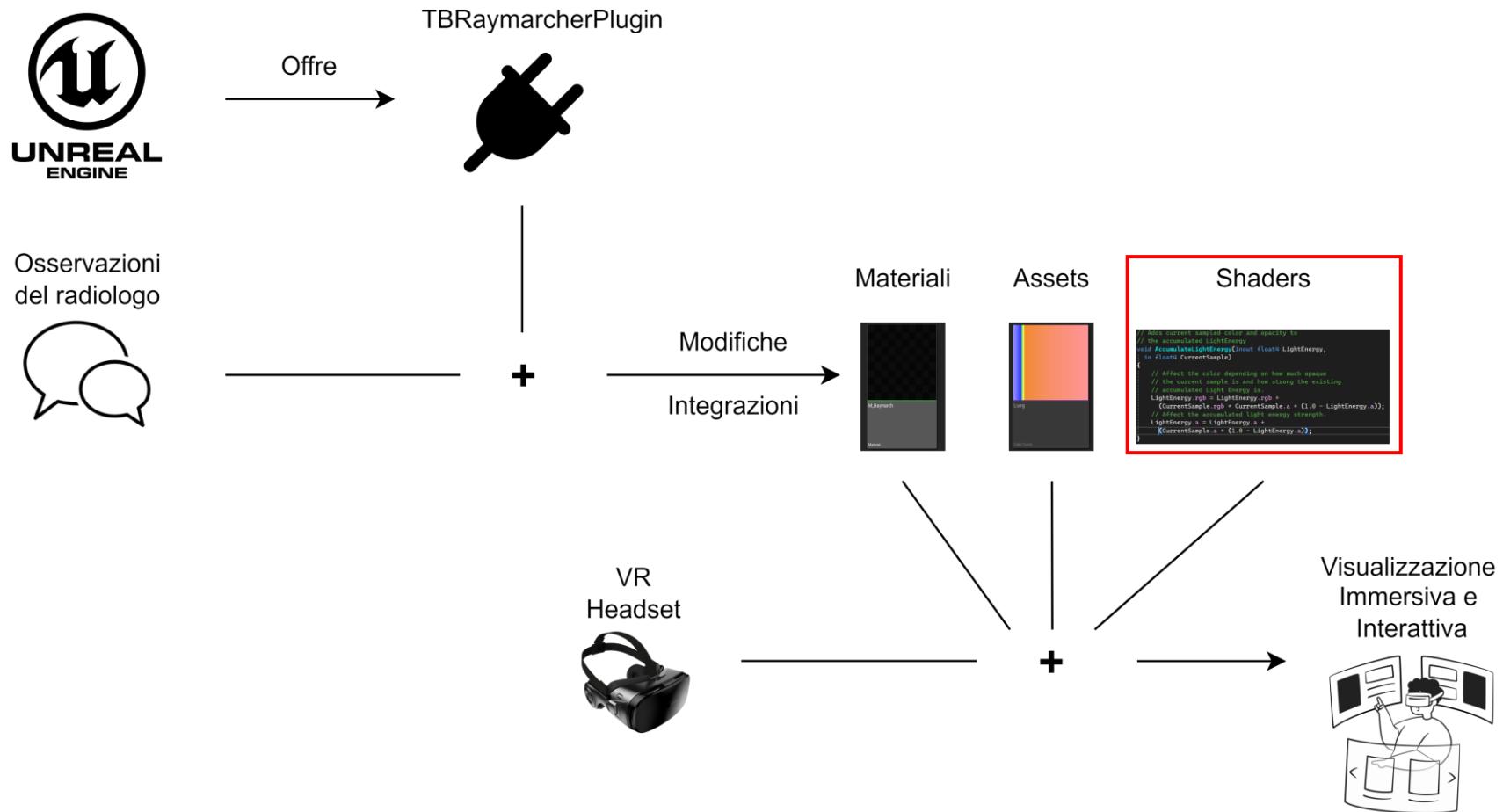
- Visualizzazione immersiva e interattiva di volumi medici (CT scan) a fini diagnostici.
- Le iterazioni sono state guidate dal Dottor Giovanni Musella, radiologo della Casa di Cura San Camillo di Cremona, in collaborazione con l'azienda LunaView.
- L'integrazione delle osservazioni del radiologo permetterà di creare un'applicazione su misura per la visualizzazione immersiva e interattiva di volumi medici.



Flusso di lavoro: Strumenti e Tecnologie



Flusso di lavoro: Strumenti e Tecnologie





UNIVERSITÀ
DI TORINO

Il TBRaymarcherPlugin

- E' un plugin open source per Unreal Engine.
- Il plugin è caratterizzato dalle seguenti funzionalità:
 - E' basato sui voxel.
 - Offre un Ray Marcher per il rendering di texture volumetriche.
 - Integra i concetti di illuminazione volumetrica.
 - Offre un parser per diverse tipologie di file.
 - Supporta il windowing.
 - Supporta le funzioni di trasferimento.

Il Voxel: l'unità fondamentale di un volume tridimensionale.

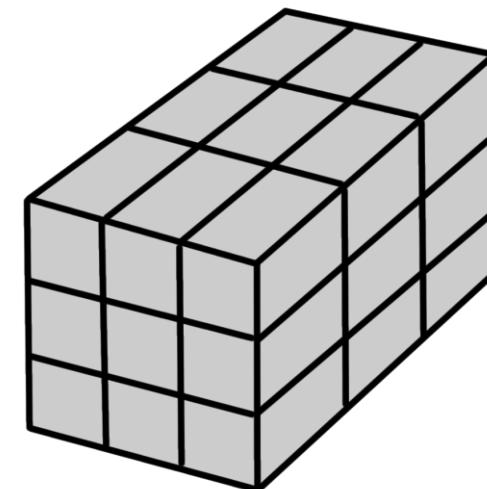
- **Definizione:** rappresentazione di un volume in un calcolatore.
 - Un dato che rappresenta un volume è un insieme di campioni (**x, y, z, [p1, p2, ...]**).
 - Il volume deve essere **ricostruito** per essere visualizzato.
 - Il volume di proprietà costante, centrato in un campione, è chiamato voxel.
- La **forma** di un voxel varia in funzione di:
 - Tipologia di campionamento.
 - Funzione di ricostruzione.

↓

Un cubo

Un parallelepipedo rettangolo

Altre forme.





Perché i Voxel?

- **Minore necessità di pre-elaborazione:** i dati ottenuti da una tomografia computerizzata o una risonanza magnetica sono dati volumetrici.
- **Conservazione dei dettagli:** poiché il voxel è la più piccola unità di dato in uno spazio tridimensionale, il rendering di voxel permette di catturare meglio i dettagli delle strutture anatomiche.

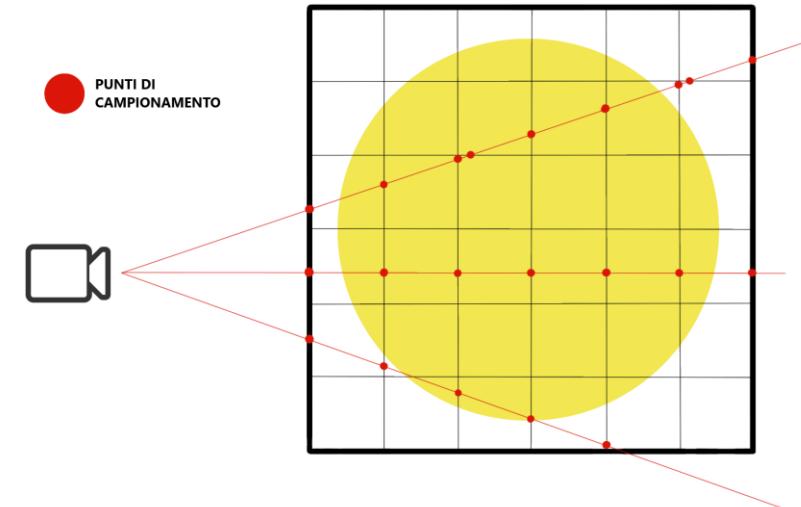


Il TBRaymarcherPlugin

- E' un plugin open source per Unreal Engine.
- Il plugin è caratterizzato dalle seguenti funzionalità:
 - E' basato sui voxel.
 - Offre un **Ray Marcher** per il rendering di texture volumetriche.
 - Integra i concetti di illuminazione volumetrica.
 - Offre un parser per diverse tipologie di file.
 - Supporta il windowing.
 - Supporta le funzioni di trasferimento.

Volumetric Ray Marching

- Il metodo:
 - Marcia all'interno del volume, **campionandolo.**
 - Plane Intersections.
 - Sphere Assisted.
 - Cube Assisted.
 - Passa i punti di campionamento al **modello ottico.**
 - **Accumula** i valori di opacità e colore calcolati dal modello ottico. Per farlo, sfrutta l'**over-operator**.



$$\begin{aligned}C^{out} &= C^{in} + (1 - \alpha^{in}) \alpha^{voxel} C^{voxel} \\ \alpha^{out} &= \alpha^{in} + (1 - \alpha^{in}) \alpha^{voxel}\end{aligned}$$



Il TBRaymarcherPlugin

- E' un plugin open source per Unreal Engine.
- Il plugin è caratterizzato dalle seguenti funzionalità:
 - E' basato sui voxel.
 - Offre un Ray Marcher per il rendering di texture volumetriche.
 - Integra i concetti di illuminazione volumetrica.
 - Offre un parser per diverse tipologie di file.
 - Supporta le funzioni di trasferimento.
 - Supporta il windowing.



UNIVERSITÀ
DI TORINO

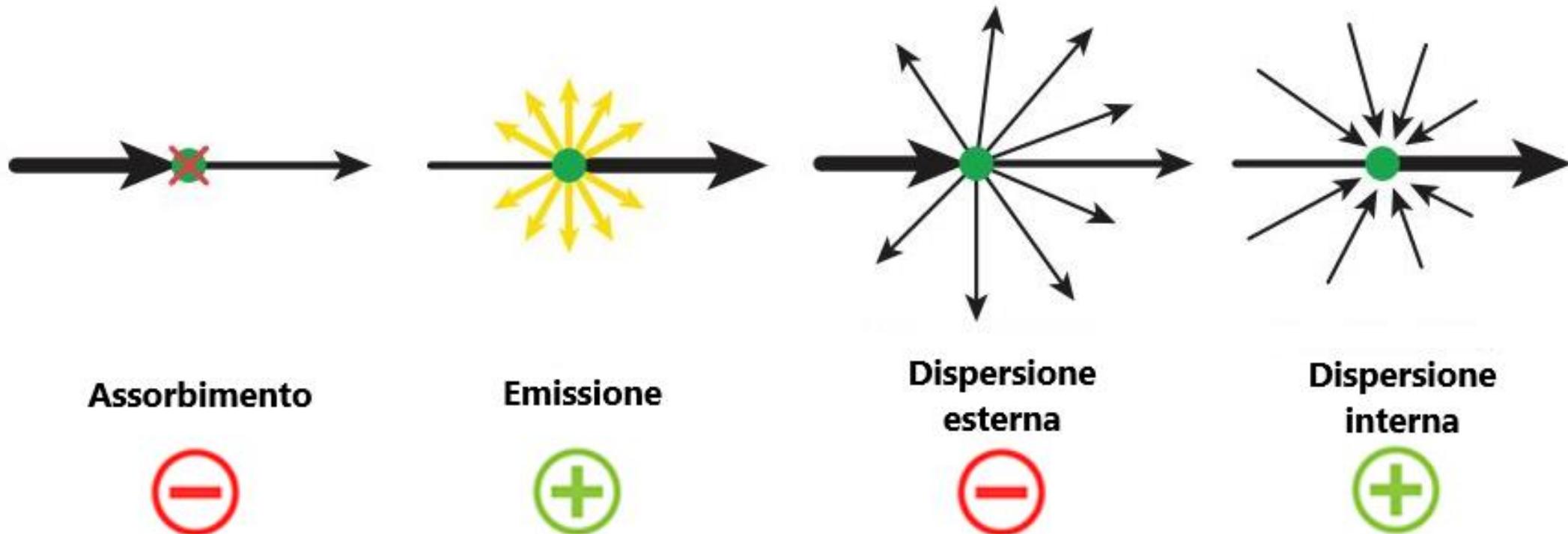
Debolezze del TBRaymarcherPlugin

- Il modello ottico realizzato è **troppo semplice**.
 - È considerato solo il fenomeno di **emissione**.
- L'emissione non è l'unico fenomeno caratteristico dell'interazione tra la luce ed un volume.

Un volume può **emettere, assorbire e disperdere la luce al suo interno e/o esterno**.

Tutto questo deve essere considerato dal modello ottico.

Emissione, Assorbimento e Dispersione interna/esterna





UNIVERSITÀ
DI TORINO

Assorbimento e Dispersione esterna

- Legge di **Beer-Lambert**.
 - σ_a = coefficiente di assorbimento.
 - σ_s = coefficiente di dispersione.

$$T = e^{-thickness (\sigma_a + \sigma_s)}$$



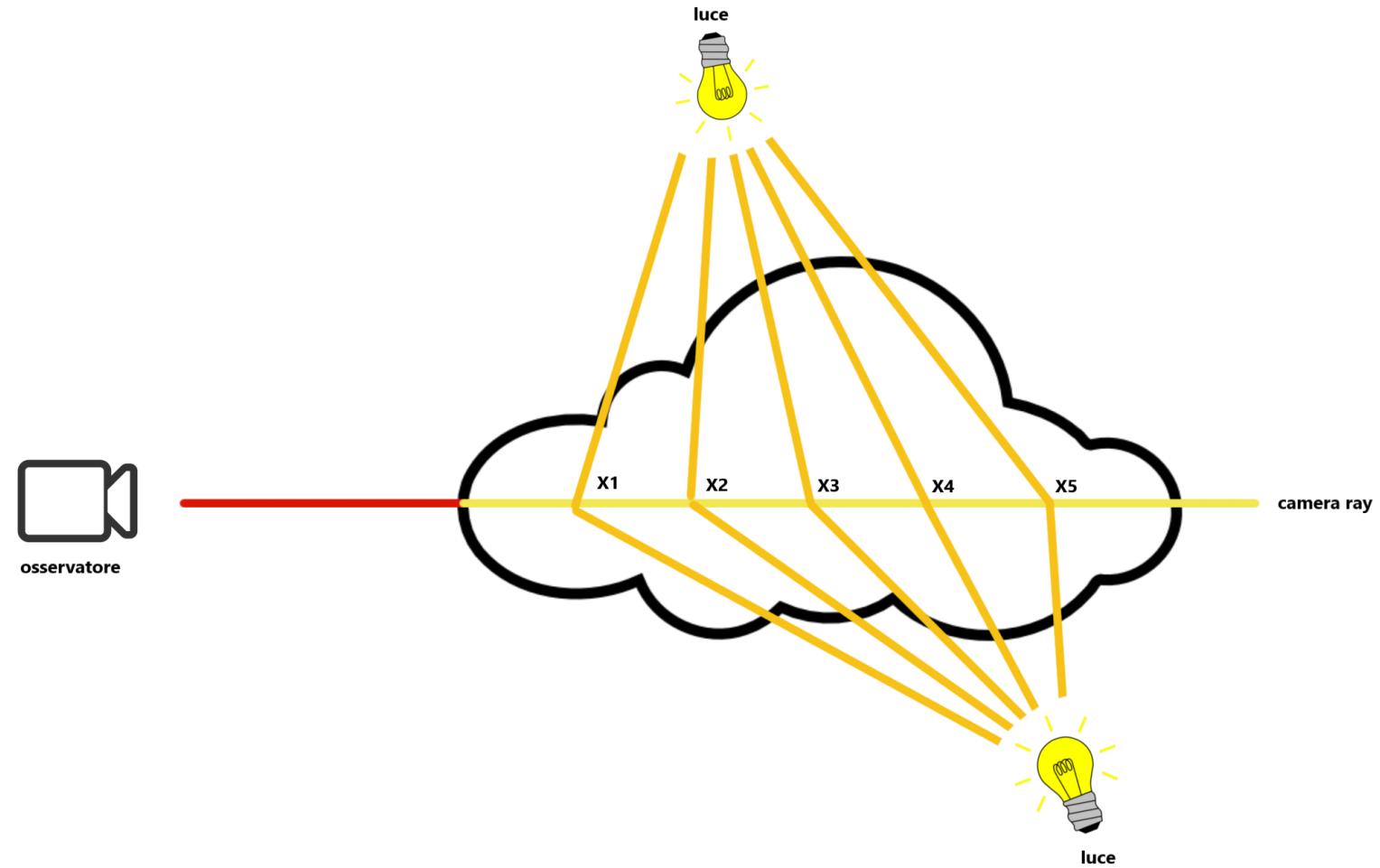
Formulazione per volumi omogenei
APPROXIMAZIONE

Dispersione interna

- Si analizza la quantità di luce deviata verso l'osservatore lungo il raggio della telecamera.

- La luce è composta da molteplici raggi.

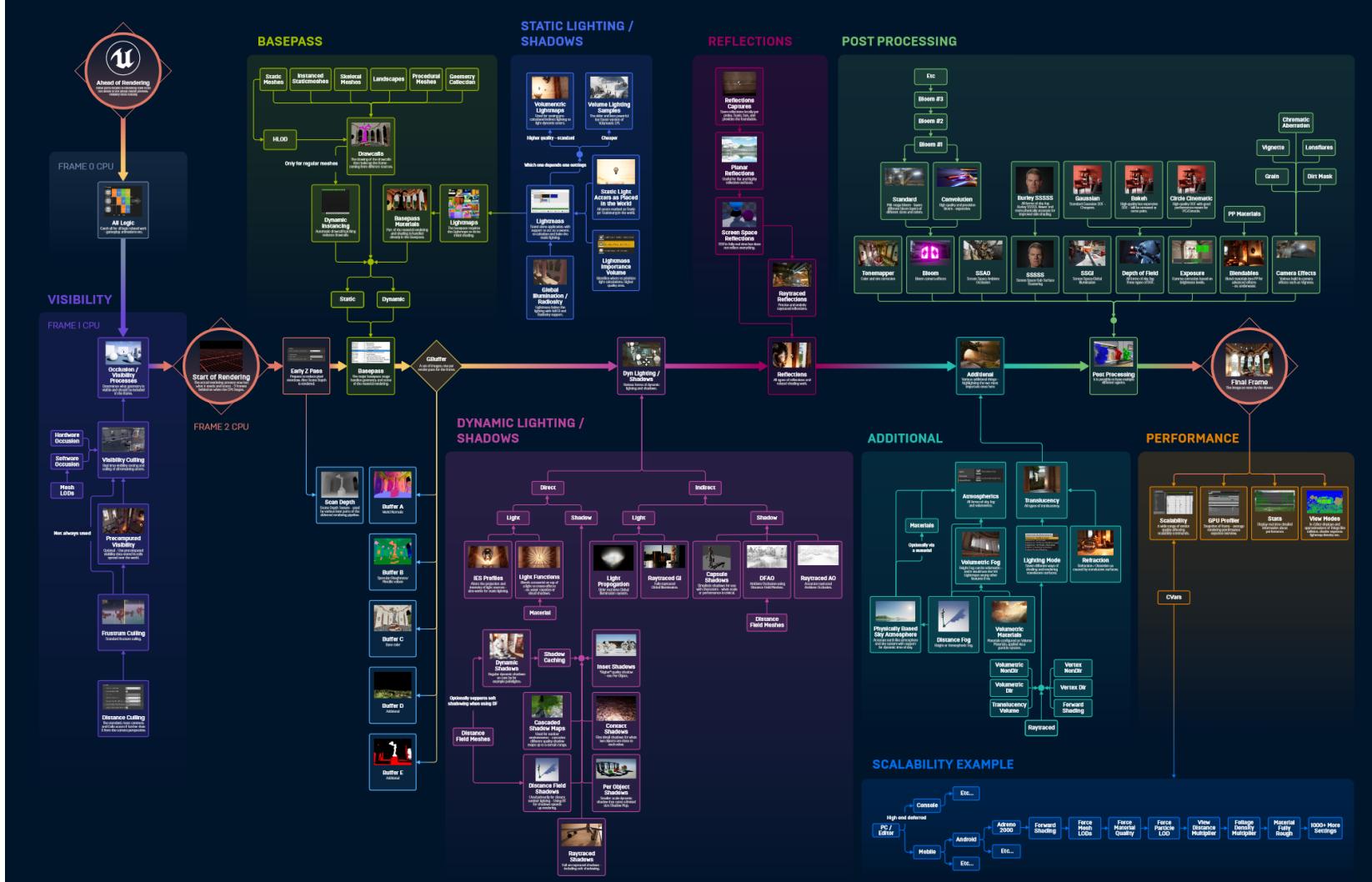
- La scena è composta da molteplici sorgenti d'illuminazione.



Implementazione: Rendering in Unreal Engine



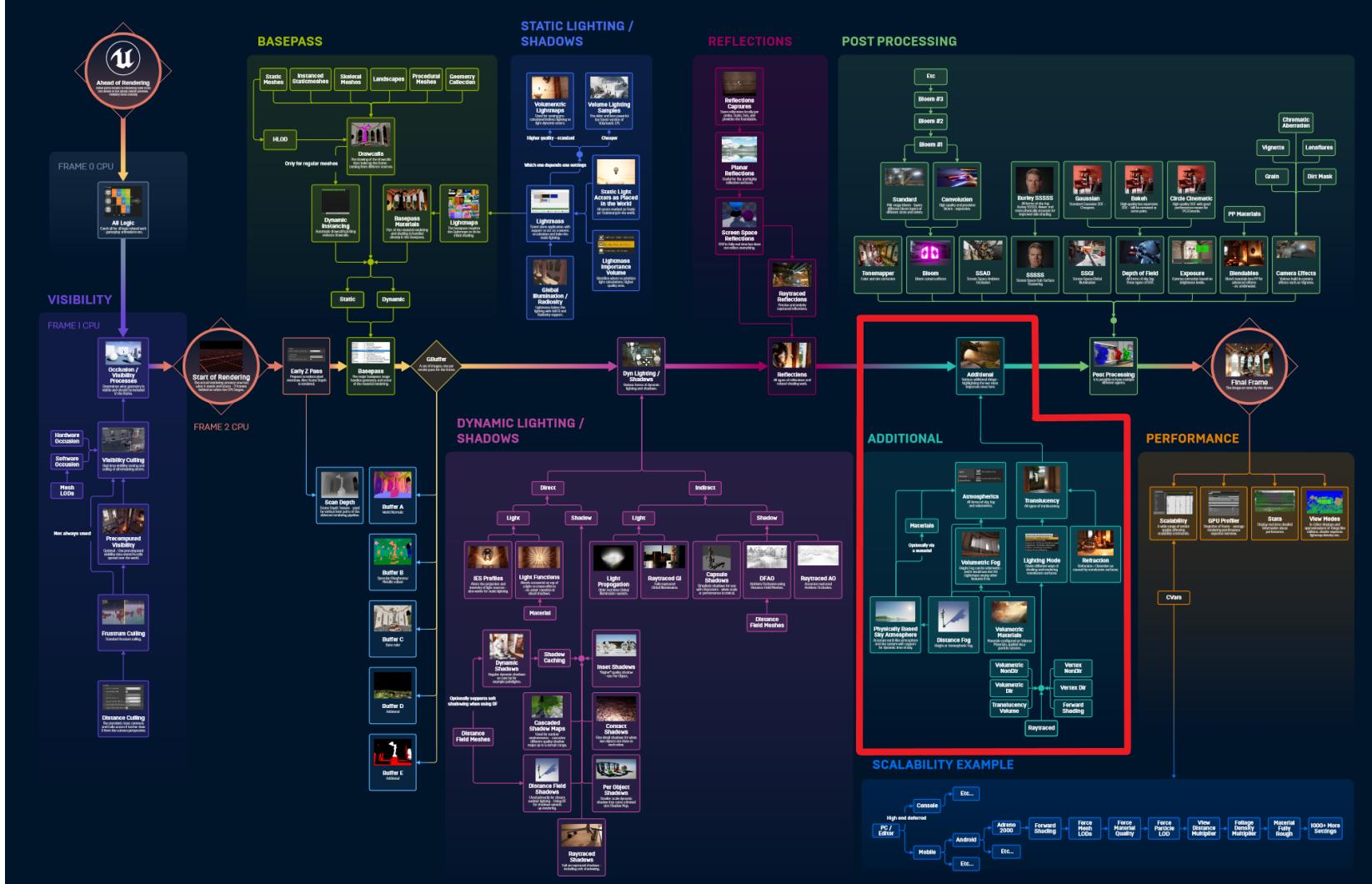
UNIVERSITÀ
DI TORINO



Implementazione: Rendering in Unreal Engine



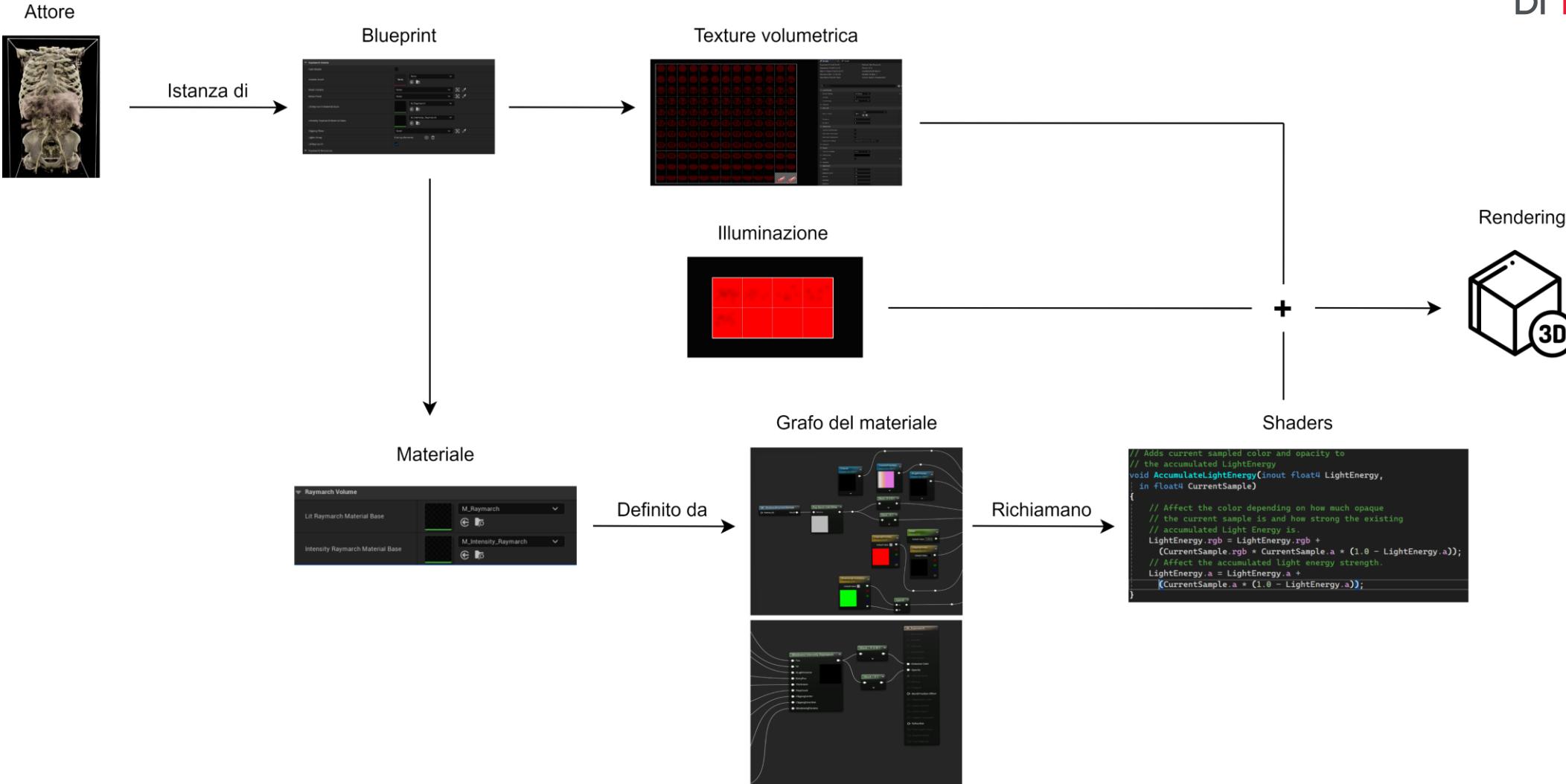
UNIVERSITÀ
DI TORINO



Implementazione: Rendering in Unreal Engine



UNIVERSITÀ
DI TORINO



Implementazione: Assorbimento e Dispersione esterna



UNIVERSITÀ
DI TORINO

- Legge di **Beer-Lambert** → $T = e^{-thickness (\sigma_a + \sigma_s)}$

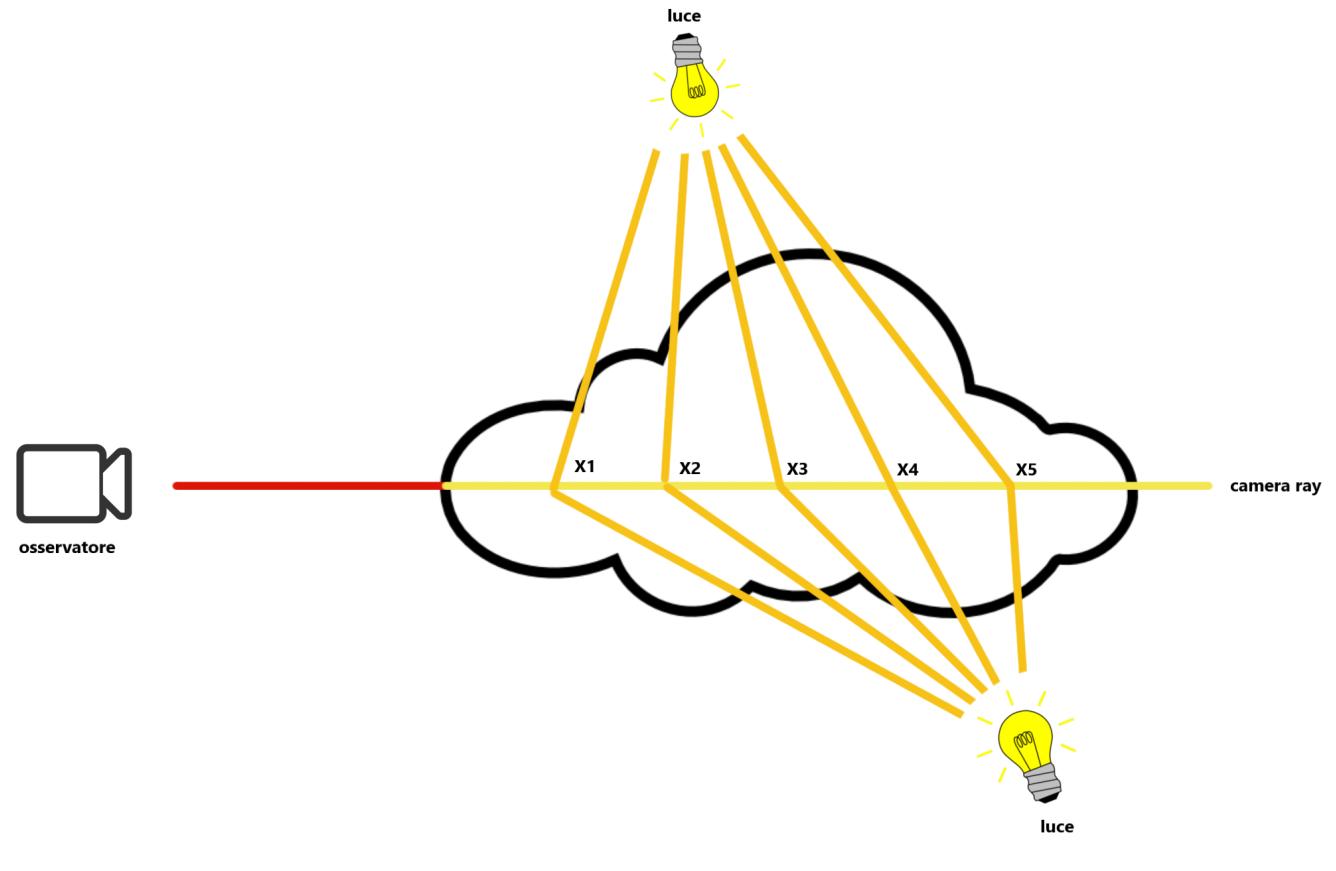
```
1 ...
2 float3 LightEnergy = (0, 0, 0); // rgb
3 float Transmission = 1; // = 1 - a
4 float sigma_a = 0.04; // Coefficiente di assorbimento.
5 float sigma_s = 0.15; // Coefficiente di scattering.
6 float sigma_t = sigma_a + sigma_s; // Coefficiente di estinzione.
7 ...
8 // attenuazione dovuta ad assorbimento e out-scattering.
9 Transmission *= exp(-StepSize*saturate(ColorSample.a)*sigma_t);
```

Implementazione: Dispersione interna



UNIVERSITÀ
DI TORINO

```
1 ...
2 // Ciclo che esegue gli step di raymarching.
3 for (i = 0; i < MaxSteps; i++)
4 {
5     ...
6     // Ciclo per calcolare l'amplificazione luminosa dovuta
7     // all'in-scattering.
8     float DensityLight = 0;
9     float LightAmp = 0;
10    for (int l = 0; l < 3; l++)
11    {
12        ...
13        // accumulo l'amplificazione luminosa portata dai campioni
14        // lungo lo shadow ray.
15        for (int s = 0; s < MaxShadowStep; s++)
16        {
17            ...
18            DensityLight += LightVolume.SampleLevel(
19                Material.Wrap_WorldGroupSettings, saturate(LPos), 0).r;
20        }
21        LightAmp +=
22            exp(
23                -DensityLight * ShadowStepSize * sigma_t * ShadowOpacity
24            ) *
25            PhaseHG(
26                -MaterialParameters.CameraVector, PosToLight, G
27            );
28    }
29    LightEnergy += ColorSample.rgb * LightAmp * sigma_s *
30        StepSize * ColorSample.a;
31 }
```





Risultati sperimentali: Confronto quantitativo

- Le statistiche sono state ricavate da simulazioni di interazioni tipiche di 3 minuti ciascuna.
- Caratteristiche della macchina:
 - RAM 32gb ddr4.
 - Intel(R) Core(TM) i7-7820X CPU @ 3.60GHz 3.60 GHz.
 - NVIDIA GeForce RTX 2080ti.
- Parametri utilizzati per simulare:
 - 3 sorgenti d'illuminazione.
 - 128 passi di Ray Marching.
 - 32 passi per calcolare l'amplificazione luminosa dovuta all'in-scattering.

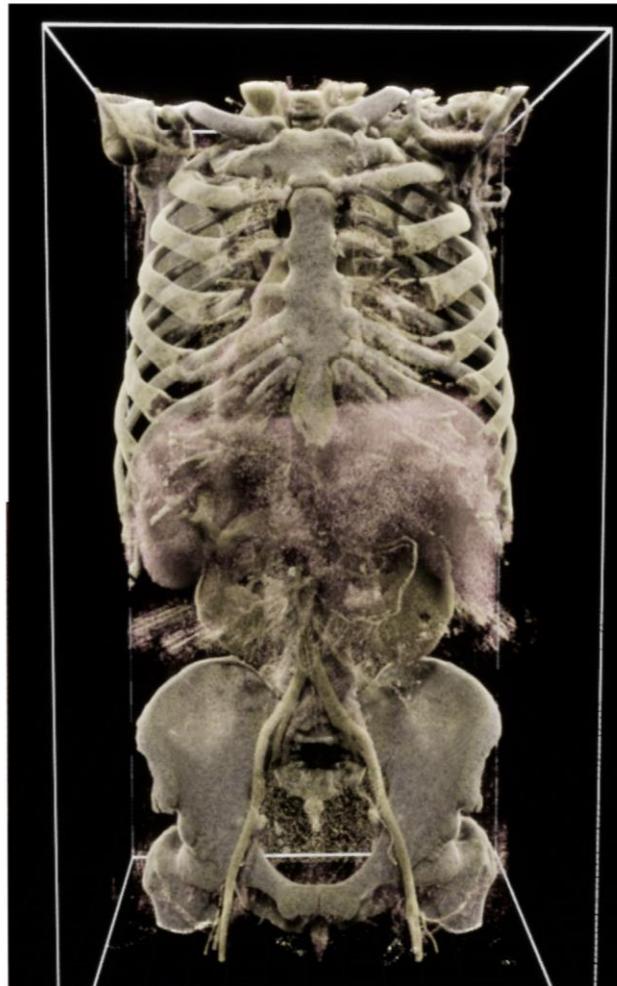
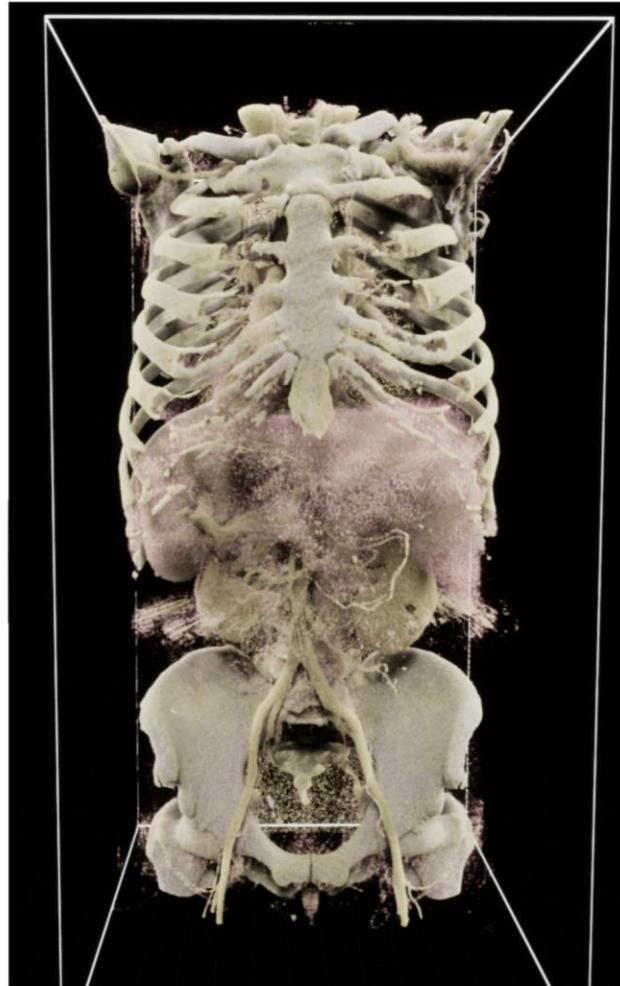


Risultati sperimentali: Confronto quantitativo

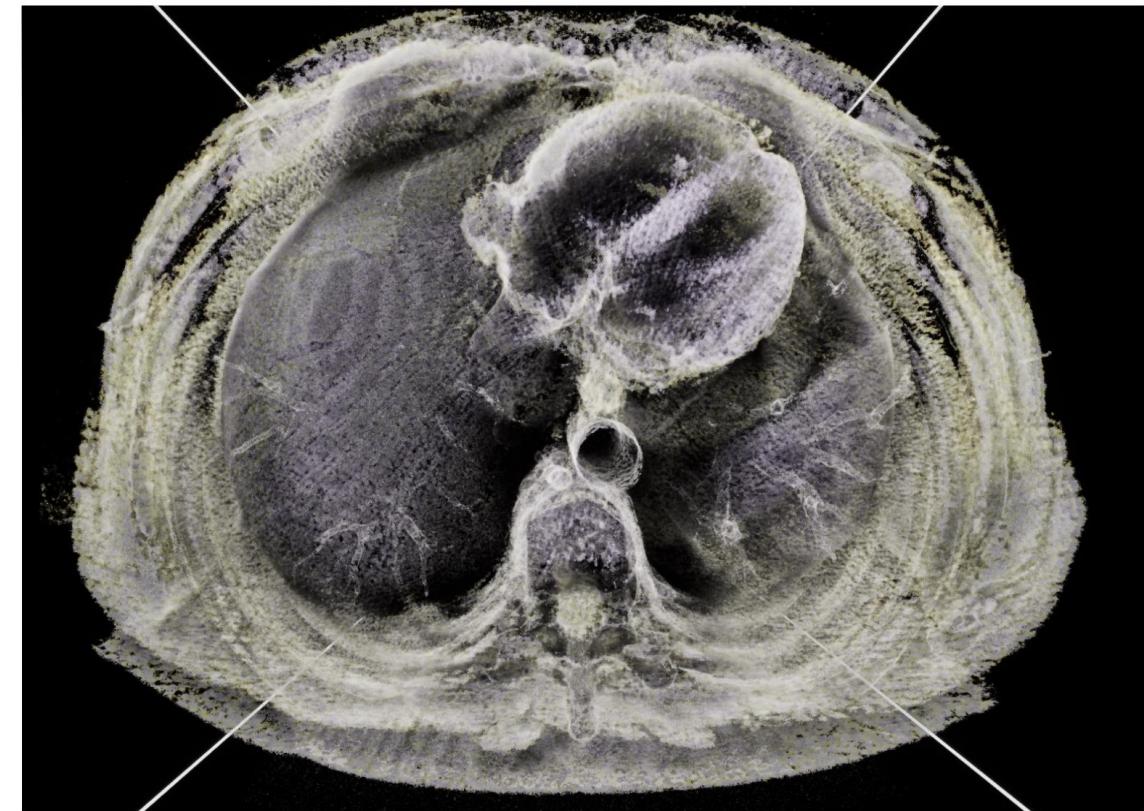
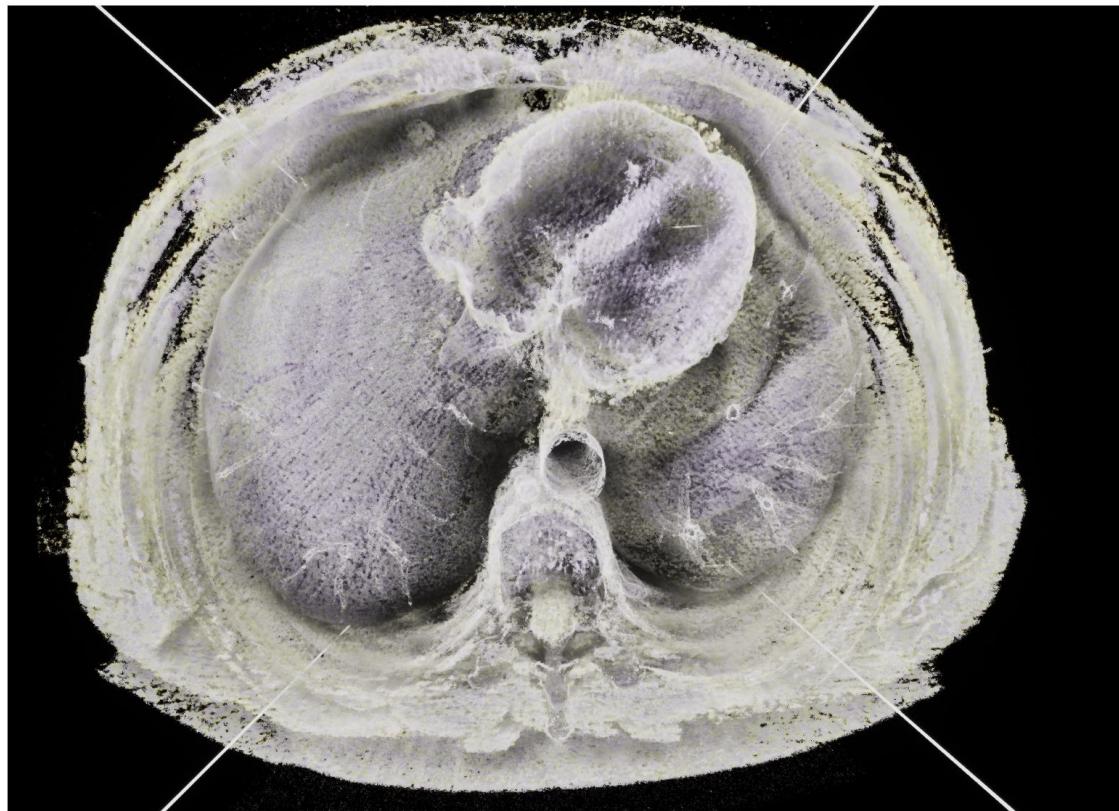
	TBRaymarcherPlugin	Ottimizzazione Proposta
Complessità Shader	236 istruzioni	325 istruzioni
FPS	36.92	34.94
Shader Compiling Thread	30.88 ms	33.60 ms
Game Thread	30.91 ms	33.59 ms
Rendering Thread	30.86 ms	33.54 ms
Pixel Shader Memory	728,96 KB	766.63 KB

- Complessità computazionale:
 - TBRaymarcherPlugin → $O(n_{pixel} * raymarching_steps)$
 - Soluzione proposta → $O(n_{pixel} * raymarching_steps * n_{light_sources} * shadow_steps)$

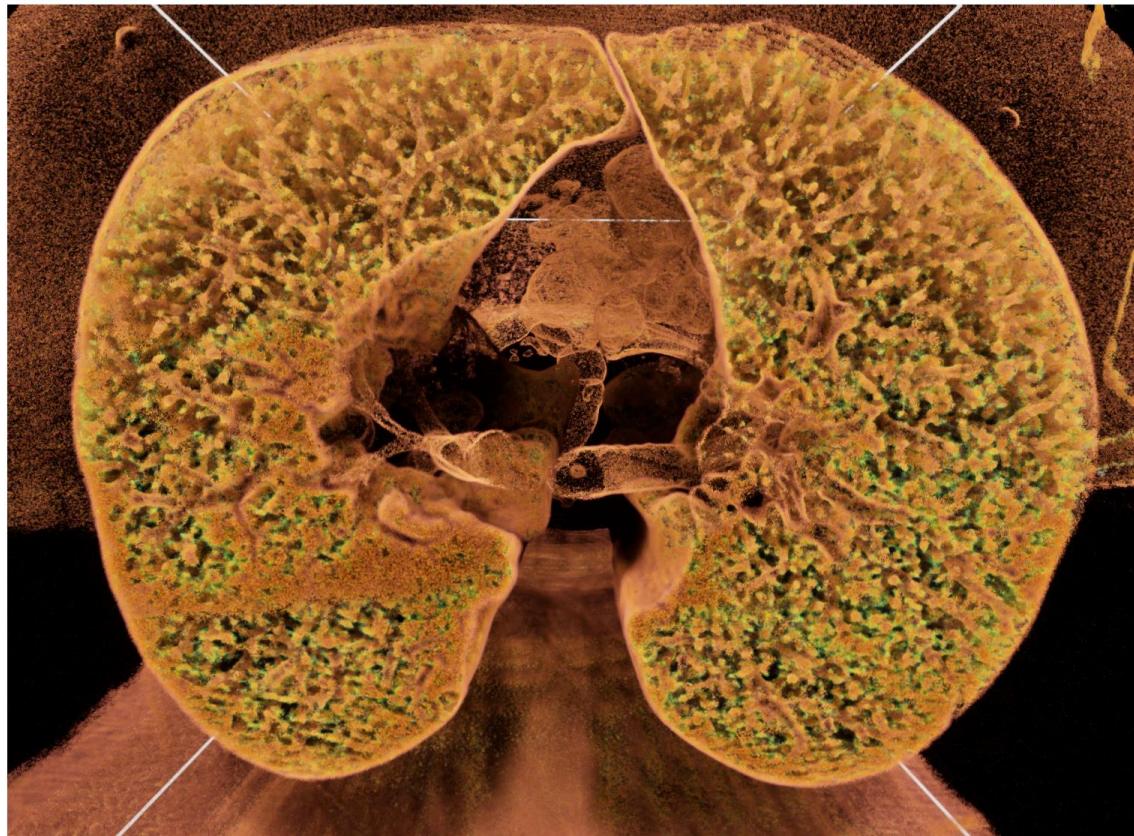
Confronto qualitativo: Esempio 1



Confronto qualitativo: Esempio 2



Confronto qualitativo: Esempio 3





Conclusioni e Sviluppi Futuri

- Confronto quantitativo:
 - Il nuovo modello è equivalente alla soluzione del TBRaymarcherPlugin dal punto di vista computazionale.
- Confronto qualitativo:
 - Miglioramento della percezione della profondità.
 - Le ombre prodotte dal nuovo modello possono rendere difficile la comprensione delle strutture.
- Sviluppi futuri:
 - Ottimizzazione delle prestazioni.
 - Miglioramento della qualità visiva.
 - Integrazione in applicazione di AR.



**UNIVERSITÀ
DI TORINO**

Grazie



UNIVERSITÀ
DI TORINO

Ottimizzazione delle prestazioni

- Tecniche di ottimizzazione:
 - Early Ray Termination (implementata).
 - Space-Leaping.
- Alternative al Ray Marching:
 - Volumetric Ray Tracing.
 - Volumetric Path Tracing.
 - Texture Based Volume Rendering.
 - Splatting.
 - Marching Cubes.



Assorbimento e Dispersione esterna: volume eterogeneo

- Legge di Beer-Lambert → introduzione di nuovi termini:
 - σ_a = coefficiente di assorbimento.
 - σ_s = coefficiente di dispersione.
 - $\sigma_t = \sigma_a + \sigma_s$ = coefficiente di estinzione.
 - $\tau = thickness * \sigma_t$ = profondità ottica per volumi omogenei.
- **Volumi eterogenei** → σ_t varia di voxel in voxel

$$\tau = \int_{s=0}^d \sigma_t(v_s) \ ds \quad \rightarrow \quad T(d) = e^{-\tau}$$