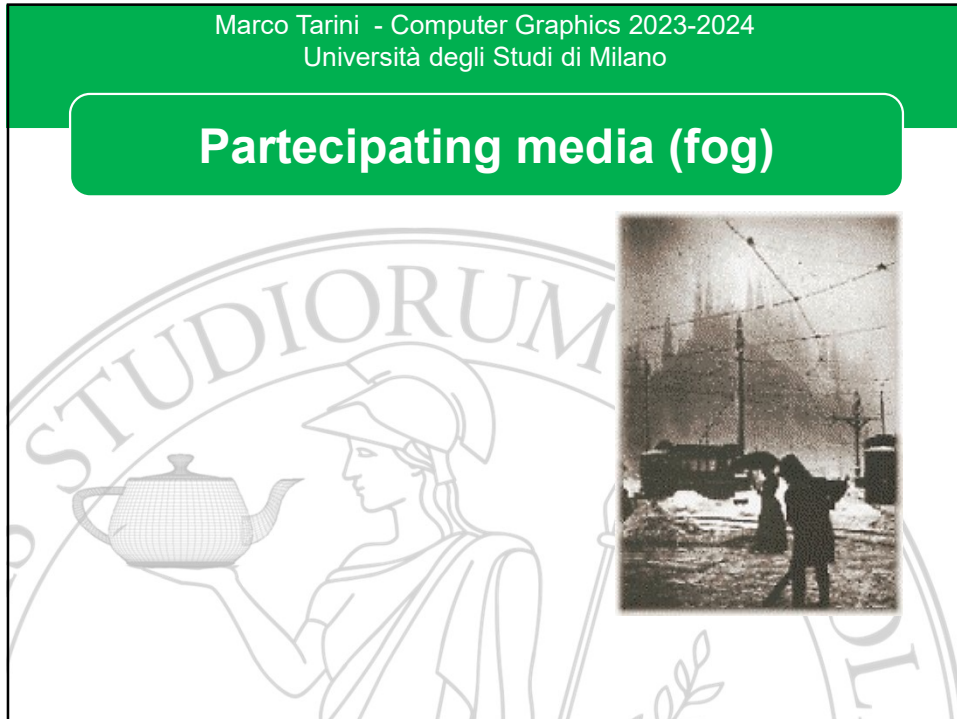


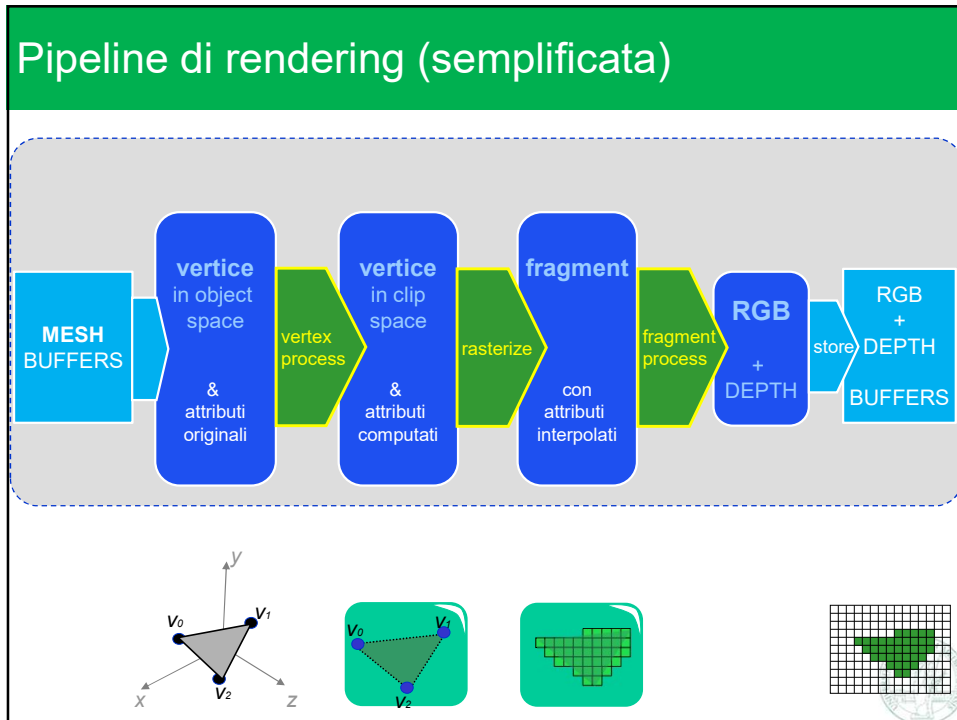
Marco Tarini - Computer Graphics 2023-2024
Università degli Studi di Milano

Participating media (fog)



The slide features a green header with the author's name and university. Below it, a white rounded rectangle contains the title "Participating media (fog)". The background is a light gray watermark of the University of Milan logo, which includes a woman holding a teapot. On the right side, there is a small, square, black and white photograph showing a person standing in a foggy, outdoor environment with some structures in the background.

1



2

Pipeline di rendering: un sommario

- ✓ Gli algoritmi di rendering su GPU (basati su rasterizzazione) sono pensati per essere implementati sulla pipeline di rendering
 - ⇒ Quale attributo è necessario per vertice?
 - ⇒ Quale quantità è calcolata in quale fase? (per vertice, o per frammento)
 - ⇒ Le variabili prodotte dal vertex program (in qualsiasi numero e di qualsiasi tipo siano) sono interpolate dal rasterizzatore e i valori sono forniti dal fragment program
- ✓ Vediamo un esempio: fog effect



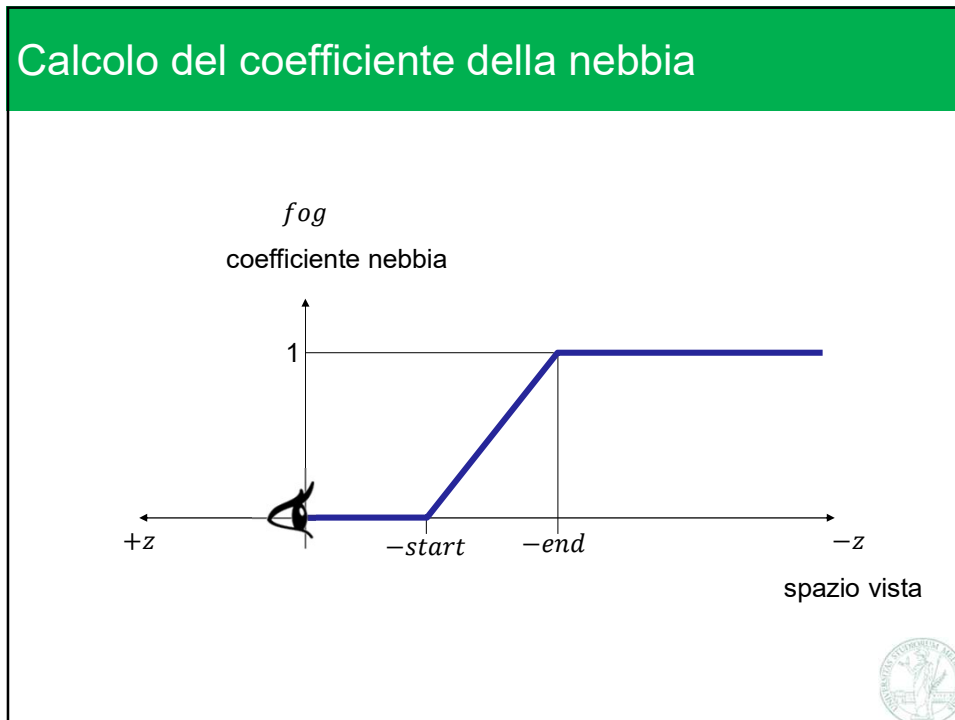
3

Fog (anche detto participating media, o depth cueing)

- ✓ L'effetto reale che vogliamo simulare: "participating medium":
 - ⇒ il mezzo (il «medium») attraverso il quale viaggia la luce influenza la luce stessa, assorbendone una parte (su alcune frequenze)
 - ⇒ Maggiorente lungo è il percorso fino alla camera, più accentuato è l'effetto
- ✓ E' facile simulare questo effetto nel rendering basato su rasterizzazione:
- ✓ Idea Base:
 - ⇒ tanto più un frammento è lontano dall'osservatore quanto più il suo colore è impattato da nebbia, foschia, etc
- ✓ Quindi:
 - la Z del vertice in *spazio vista* determina il fattore di nebbia
 - ⇒ potremmo calcolare l'effetto anche in spazio clip o spazio schermo



4



5

Calcolo del coefficiente della nebbia

- ✓ A parole: «il coefficiente nebbia è una funzione lineare clampata (fra 0 e 1) della z in spazio vista»
- ✓ In formula, dato un punto di coordinata z in spazio vista :

$$fog = clamp\left(\frac{-end - z}{-end + start}, 0, 1\right)$$

coefficiente nebbia
0=niente nebbia
1=si vede solo nebbia

funzione
che porta a 0 se <0
e a 1 se >1

Verificare che questa formula restituisce...
0 quando $z = -start$
1 quando $z = -end$

- ✓ $start$ e end sono parametri dell'effetto:
 - ⇒ $start$: la distanza (positiva) dall'osservatore prima della quale non si percepisce alcuna nebbia
 - ⇒ end : la distanza (positiva) dall'osservatore, dopo la quale si ha 100% nebbia (l'oggetto sparisce del tutto)
- ✓ Anche altre formule (più realistiche di questa) sono possibili

6

Usare il fattore nebbia applicare il colore

- ✓ Dopo aver calcolato il fattore nebbia, lo applico al colore finale del pixel associato al frammenti
⇒ ricordando che colore = tripletta di valori r,g,b

$$colore_{finale} = fog \cdot colore_{nebbia} + (1 - fog) \cdot colore_{originale_frammento}$$

il colore
nebbia
(è una
costante,
e un
parametro)

a parole:
"il colore finale è un'interpolazione lineare
fra il colore-nebbia e il colore-originale"
(l'ennesimo l'uso di interpolazione lineare
che incontriamo, questa volta sui colori)

Verificare che questa formula restituisce... $colore_{nebbia}$ quando $fog = 1$
 $colore_{originale}$ quando $fog = 0$



7

Usi dell'effetto fog in CG 1/2

1. Simulare (in modo approssimato) il participating media
⇒ Nebbia, foschia, acqua sporca (rendering subacquei) etc;
⇒ Se color fog (e sfondo) = nero: effetto oscurità (artificiale)
2. Mascherare gli artefatti di *popping* dovuto al *clipping* contro il *far-clipping-plane*
⇒ Senza fog: gli oggetti spariscono o riappaiono di colpo quando attraversano il *far-clipping plane*
⇒ Con fog: gli oggetti subiscono un *fade-out*, cioè si dissolvono progressivamente nella nebbia, con continuità, diventando color-sfondo, man mano che si avvicinano al *far clipping plane*.
⇒ Per ottenere questo effetto dobbiamo scegliere:
(a) il colore della nebbia identico al colore sfondo
(b) la distanza *far* coincidente con la distanza *far* usata nella matrice di proiezione
⇒ Usato nei videogames (che spesso hanno bisogno di *far clipping plane* più vicini di quanto si desidera, per motivi di efficienza)



8

Usi dell'effetto fog in CG 2/2

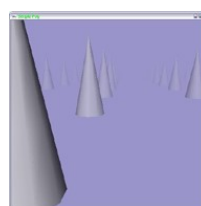
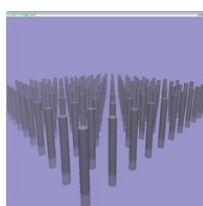
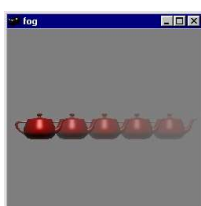
3. Sugerire visivamente la profondità degli oggetti:

- ⇒ Nel contesto della visualizzazione scientifica (e altri)
- ⇒ Il fog consente di leggere la profondità relativa degli oggetti nei rendering 3D, e produrre immagini intuitive e facili da interpretare
- ⇒ Particolarmente utile in assenza di altri indizi, come la prospettiva (cioè quando è usata proiezione ortografica)
- ⇒ Particolarmente utile in presenza di forme 3D complesse e non intuitive
- ⇒ Il fog usato per questo scopo è detto «**depth cueing**» («dar spunti di profondità»)

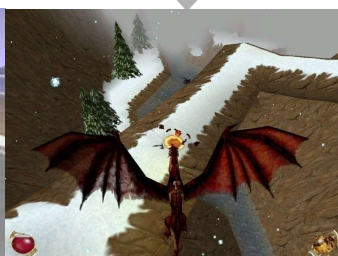


9

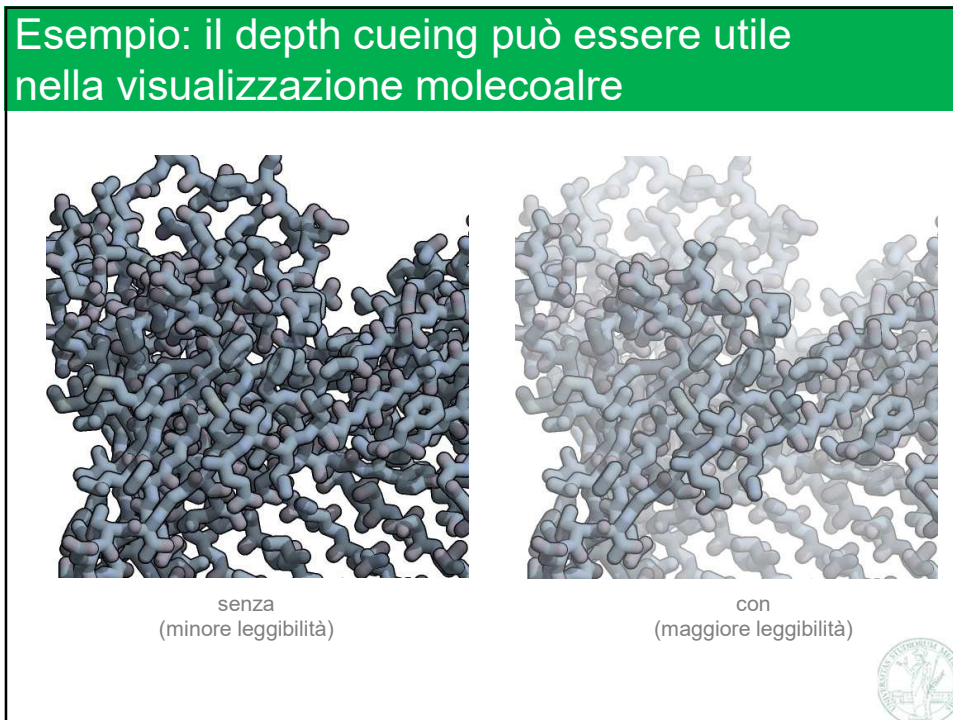
Esempi



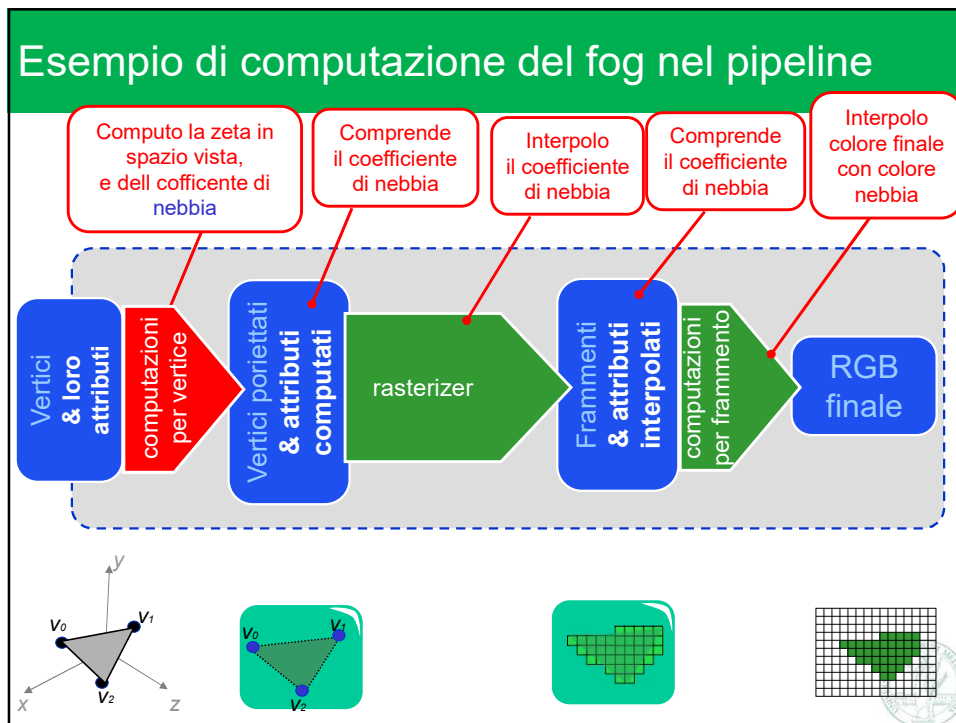
spesso usato nei videogames per mascherare il **popping** dovuto al **clipping** contro il **far-clipping-plane**



10



11



12

Implementare l'effetto fog 1/2

- ✓ Il computo del fog è implementato dal programmatore a livello dell'app grafica
 - ⇒ Nessun particolare supporto HW è necessario
 - ⇒ (a differenza di meccanismi come depth-test o back-face culling che sono implementati in HW)
- ✓ A basso livello (in API come OpenGL o WebGL):
 - ⇒ l'effetto fog è implementato programmandolo nel vertex shader (il programma utente eseguito su ogni vertice) e nel fragment shader (il programma utente eseguito su ogni frammento)
- ✓ Nelle librerie ad alto livello (come three.js o Unity)
 - ⇒ l'effetto fog è già implementato dalla libreria, e deve solo essere abilitato
- ✓ (le stesse considerazioni varranno per il lighting)



13

Implementare l'effetto fog 2/2

- ✓ Le librerie ad alto livello, come `three.js`, implementano per noi l'effetto fog. Dobbiamo solo settarne i parametri e abilitarlo (vedi `cgLab04.html` e `.js`)

```
var colorNebbia = 0x00FFFF; // un azzurro cielo
var near = 0.5; // misurato in spazio vista
var far = 4.0; // misurato in spazio vista
var nebbia = new THREE.Fog(colorNebbia, near, far);

miaScena = new THREE.Scene();
// ...
miaScena.fog = nebbia;
```



14