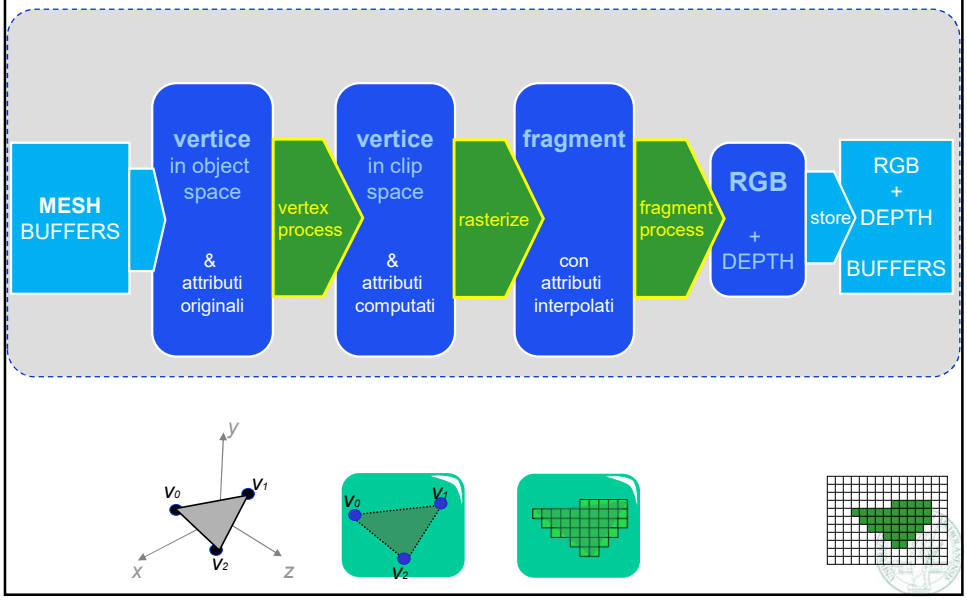
Marco Tarini - Computer Graphics 2022-2023
Università degli Studi di Milano

Partecipating media (fog)

1

Pipeline di rendering (semplificata)



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, participating media

- ✓ L'effetto reale che vogliamo simulare: "participating medium":
  - ⇒ il mezzo attraverso il quale viaggia la luce influenza la luce stessa, assorbendone una parte (su alcune frequenze)
  - ⇒ Maggiore è il percorso fino alla camera, più forte è l'effetto
- ✓ E' facile simulare questo effetto nel rendering basato su rasterizzazione
- ✓ Idea Base:
  - ⇒ tanto più un punto è lontano dall'occhio quanto più il suo colore è impattato da nebbia, foschia, etc
- ✓ La Z del vertice in spazio vista determina il fattore di nebbia

4

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 coordianta z in spazio vista :

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

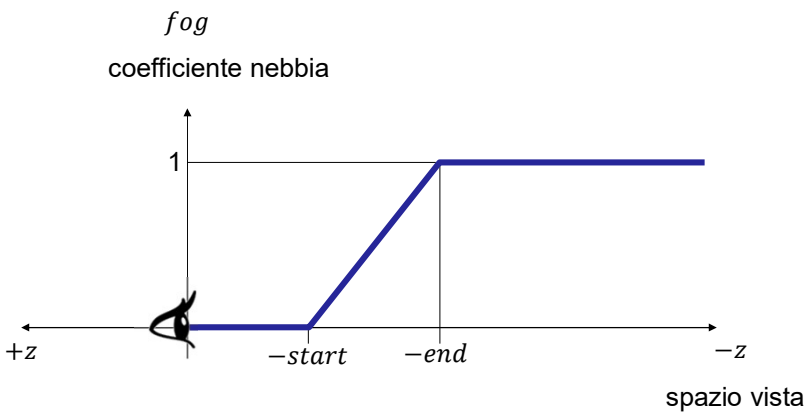
opacità finale della nebbia
0=niente nebbia
1=si vede solo nebbia

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

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

5

Calcolo del coefficiente della nebbia



6

Fog: applicare il colore

✓ Dopo aver calcolato l'opacità, la applico al colore dei frammenti

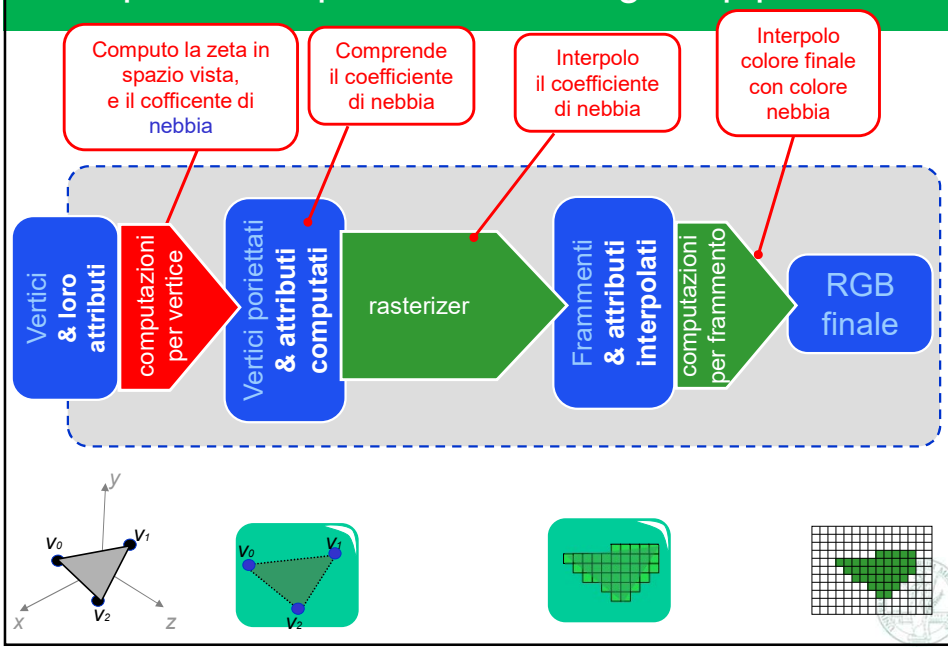
$$colore_{finale} = fog \cdot colore_{nebbia} + (1 - fog) \cdot colore_{originale\_frammento}$$

il colore nebbia (è un altro parametro)

a parole:
"il colore finale e' una interpolazione lineare fra il colore-nebbia e il colore-originale"
(l'ennesima l'uso di interpolazione lineare che incontriamo)

7

Esempio di computazione del fog nel pipeline



8

Usi dell'effetto fog in CG 1/2

1. Simulare crudelmente la presenza di partecipating media
   - ⇒ Nebbia, foschia, acqua sporca (rendering subacquei) etc;
   - ⇒ Se color fog (e sfondo) = nero: effetto oscurità (finto)
2. Mascherare gli artefatti di *popping* dovuto al *clipping* contro il *far-clipping-plane*
   - ⇒ Senza effetto nebbia:
     le cose «spariscono» di colpo quando attraversano il *far-clipping plane*
   - ⇒ Con: gli oggetti subiscono un *fade-out*, cioè si dissolvono progressivamente nella nebbia con continuità, diventando color-cielo man mano che si avvicinano al *far clipping plane*.
   - ⇒ Per ottenere questo:
     (a) il colore della nebbia deve coincidere con il colore sfondo
     (b) la distanza far del fog deve coincidere 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)

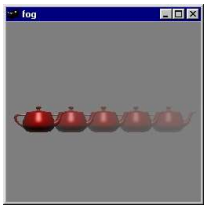
9

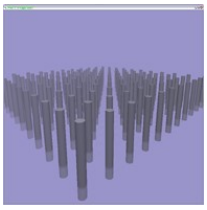
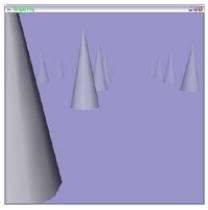
Usi dell'effetto fog in CG 2/2

3. Suggerire visivamente la profondità degli oggetti:
   - ⇒ Nel contesto della visualizzazione scientifica (e altri)
   - ⇒ il fog consente di leggere la profondità relativa 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»)

10

Esempi



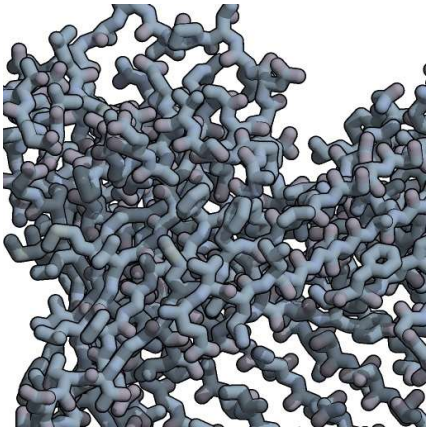



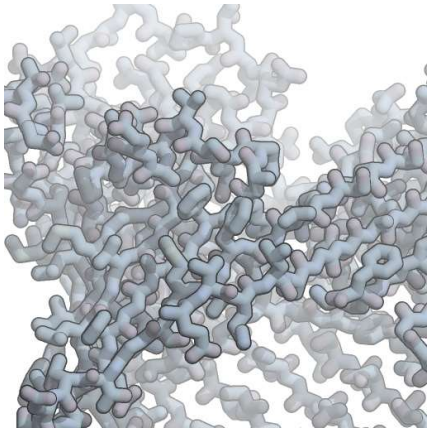
spesso usato nei giochi per mascherare il popping dovuto al clipping contro il far-clipping-plane



11

Esempio di depth cueing nella visualizzazione molecoalre


senza


con

12

Implementare l'effetto fog 1/2

- ✓ Il computo del fog è implementato dal programmatore a livello dell'app grafica
  - ⇒ non è una funzionalità prevista dalla GPU o dalle API
  - ⇒ Nessun particolare supporto HW è necessario
  - ⇒ A differenza di come come depth-test o back-face culling
- ✓ Nelle API a basso livello (come OpenGL o WelGL) l'effetto fog è quindi 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, l'effetto fog è implementato a livello di libreria, e va solo abilitato:

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 cgLab03.html e .js, righe markate con «(new!)»)

```
var colorNebbia = 0xFF00FF;
var near = 0.5; // in spazio vista
var far = 4.0; // in spazio vista
var miaNebbia = new THREE.Fog(colorNebbia, near, far);

miaScena = new THREE.Scene();
miaScena.fog = miaNebbia;
```

14