Corso di *Tecniche Avanzate per la Grafica*

Image-based rendering

Docente:

Massimiliano Corsini

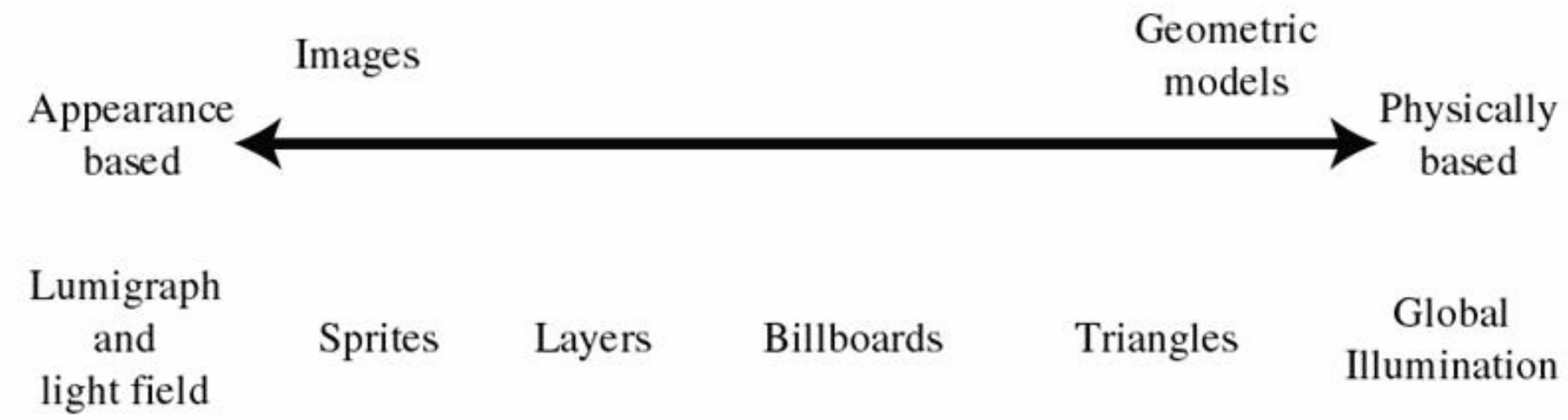
Laurea Specialistica in Informatica

Facoltà di Scienze MM. FF. NN.
Università di Ferrara

Image-based Rendering

- L’idea base è utilizzare (in vari modi) immagini al posto dei dati degli oggetti (geometria + materiali).
- Un oggetto complesso può essere visualizzato più velocemente rimpiazzandolo (in modo coerente) con una sua immagine.
- Si ottiene maggior fotorealismo con minore sforzo computazionale.

Image-based Rendering



- Si va dalle tecniche di rendering completamente basate su immagini (nessun modello 3D della scena) → Lumigraph, Light Fields, QuickTime VR
- Ad ibridi scena 3D-immagini (billboards)
- Fino ad arrivare a rendering dove nessun'altra immagine viene utilizzata per generare l'immagine finale (algoritmi di illuminazione globale)

Image-based Rendering

- Metodi
  - Sprites
  - Billboards
  - Impostors
- Applicazioni
  - Lens Flare
  - Particles System
  - Full Screen Billboarding
  - Skybox

Billboarding

- Molti effetti visivi, come il lens flares ad esempio, sono basati sulla tecnica del ***billboarding***.
- Il ***billboarding*** consiste nel renderizzare un'immagine su un quadrilatero orientato secondo il punto di vista.
- Insieme all'*alpha texturing* questa tecnica può essere utilizzata per visualizzare fenomeni come la nebbia, il fumo, il fuoco, le nuvole, oppure per rimpiazzare con immagini oggetti complessi, come ad esempio un albero.

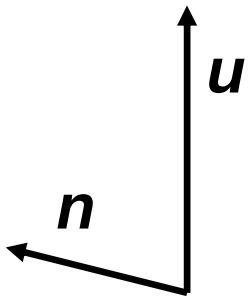
Alpha Texturing

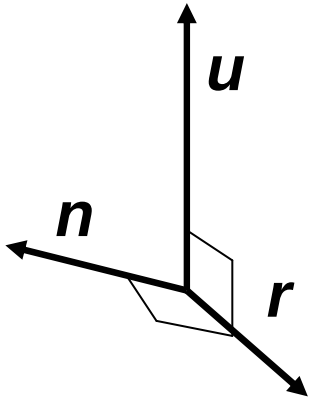
Billboarding

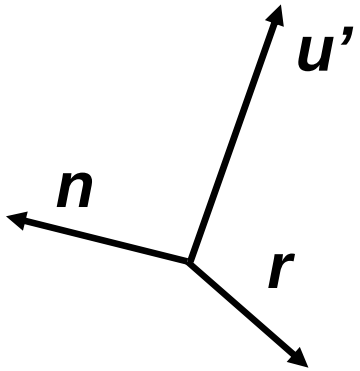
- Tipicamente, l'orientazione del quadrilatero desiderata è caratterizzata dalla sua normale (***n***) e dalla direzione di "alto" (up direction ***u***).
- Grazie a questi due vettori è possibile definire una base ortonormale alla superficie e quindi una matrice di rotazione per orientare il quadrilatero nel modo desiderato.
- In tutte le tecniche di billboarding uno di questi due vettori è fisso e deve mantenere una determinata direzione. L'altro deve essere reso perpendicolare a questo.

Billboarding

- Per ottenere la base ortonormale si procede nel seguente modo:
  - Si calcola il vettore “right” ***r = n x u*** e lo si normalizza
  - Se ***n*** è il vettore fisso si ottiene la nuova up direction (***u’***) calcolando il prodotto vettoriale ***u’ = n x r***
  - Se ***u*** è il vettore fisso si ottiene la nuova normale (***n’***) calcolando il prodotto vettoriale ***n’ = u x r***







Billboarding

- Questi tre vettori ci permettono di ottenere la matrice di rotazione per orientare il quadrilatero.
- Ad esempio, nel caso di normale fissa la matrice M risulta → ***M = (r , u' , n)***
- È importante sottolineare che la matrice si applica assumendo che il quadrilatero di partenza giacia sul piano xy con +y come direzione di alto ed il centro del quadrilatero come *anchor position*.

Tipi di billboarding

- Esistono diverse tipologie di billboarding a seconda del modo di orientazione:
  - Screen-Aligned Billdboarding
  - World-Oriented Billboarding
  - Axial Billboarding

Screen-Aligned Billboarding

- È la forma più semplice di billboarding.
- L'obiettivo di questo tipo di billboarding è ottenere un effetto ***"sprite"***, ossia l'immagine sempre parallela allo schermo e con il vettore up costante.
- Per questo tipo di billboarding la normale desiderata è la normale del view plane cambiata di segno ed ***u*** è l'up vector della camera ($\boldsymbol{u}_{\boldsymbol{camera}}$).
- Quindi sia ***n*** che ***u*** sono fissi (quindi anche ***r*** è fisso) → si deve solo ruotare le billboard.
- Questa tecnica è utilizzata per visualizzare annotazioni o per il lens flares.

Sprite...?

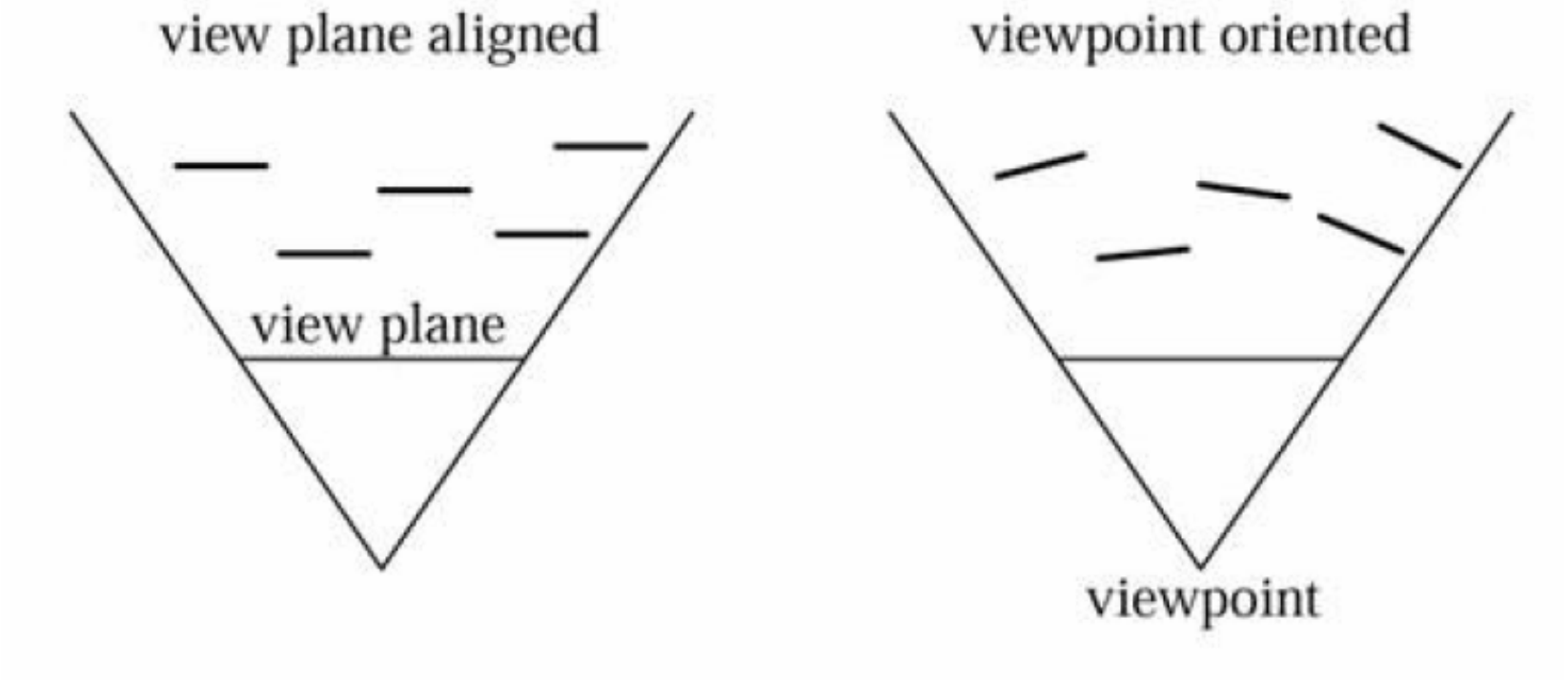
- Sprite
  - Usati nei vecchi videogiochi 2D
  - Esempio: cursore del mouse
  - Definiti in un rettangolo
  - Mapping 1-to-1 con I pixels dello schermo

background **sprite image** **result**

World-Oriented Billboarding

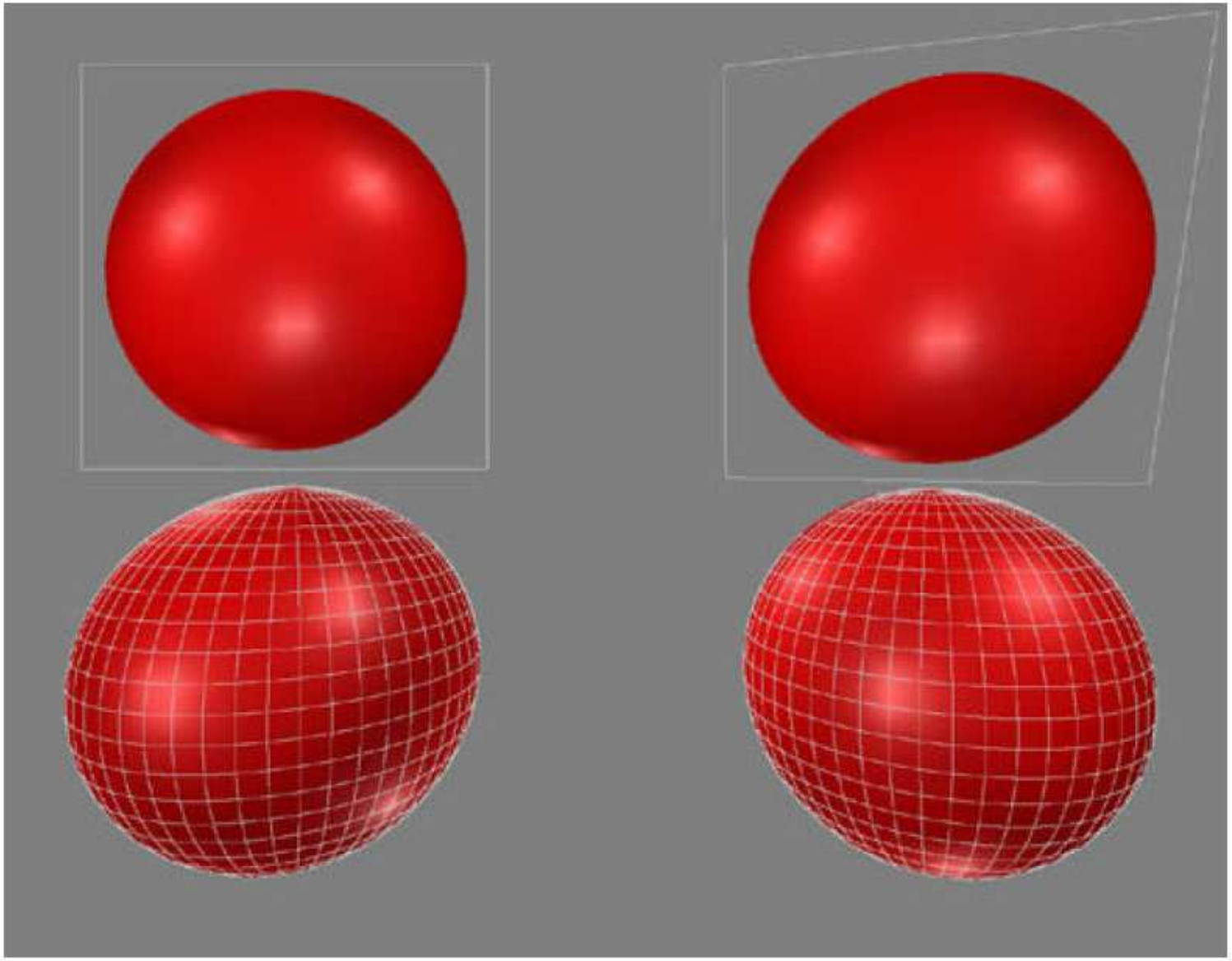
- Screen-aligned billboarding funziona bene per oggetti a simmetria circolare, per i quali è possibile utilizzare $\boldsymbol{u}=\boldsymbol{u}_{camera}$ senza grossi inconvenienti
- Ma in generale, se la camera ruota intorno alla direzione di vista questo può causare problemi. Si utilizza quindi al posto di $\boldsymbol{u}_{camera}$ l'up vector della scena ($\boldsymbol{u}_{world}$). In questo caso il vettore $\boldsymbol{n}$ rimane lo stesso del caso precedente.
- Un altro modo di allineare gli oggetti è quello di utilizzare come vettore $\boldsymbol{u}=\boldsymbol{u}_{world}$ e come vettore $\boldsymbol{n}$ la direzione che connette il centro della billboard al punto di vista → ***VIEWPOINT ORIENTED BILLBOARDING***.

World-Oriented Billboarding

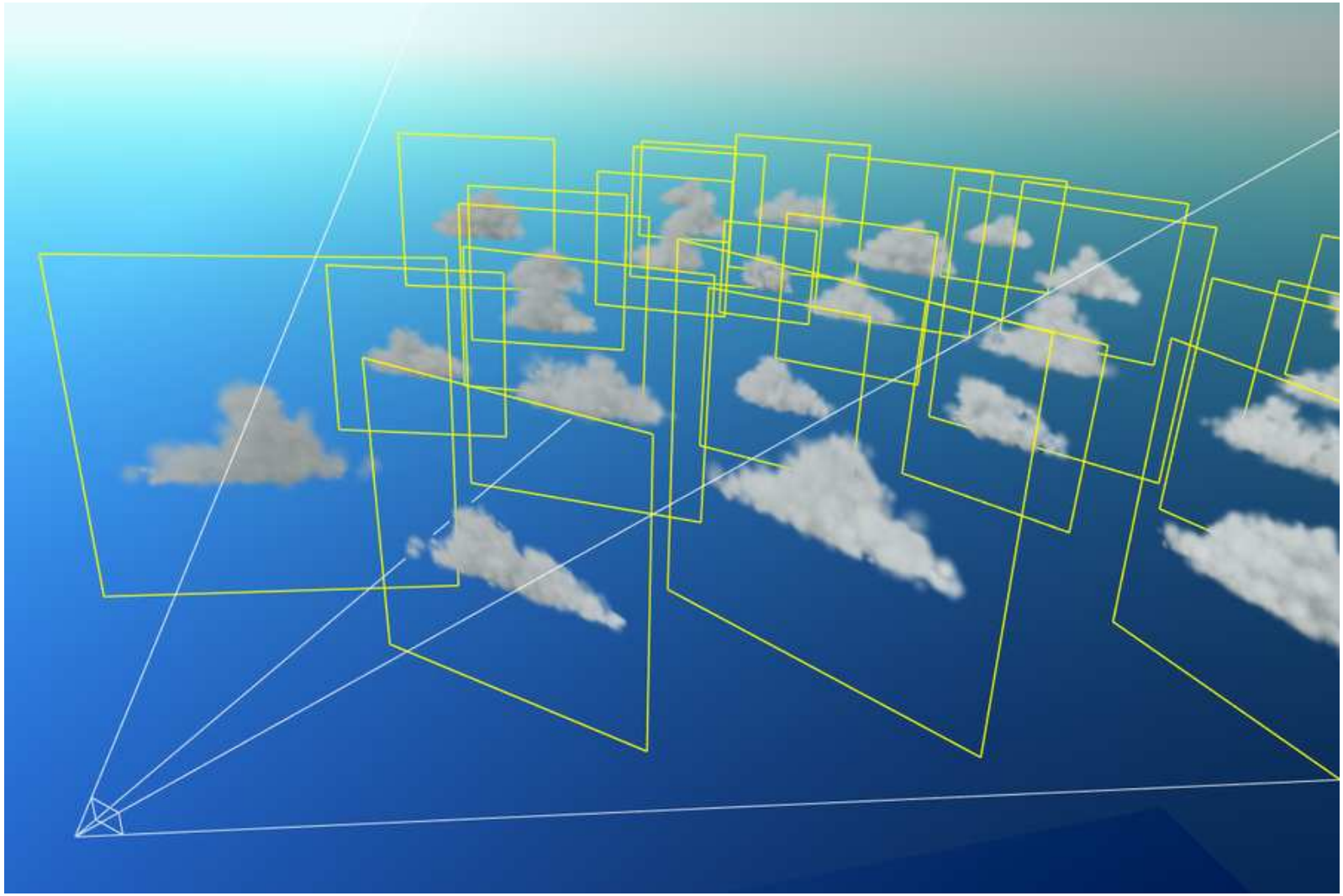


- Viewpoint orientation introduce distorsione prospettica nella billboard → questo conferisce un maggiore realismo all'oggetto
- Risulta una buona tecnica per visulizzare ***impostor***

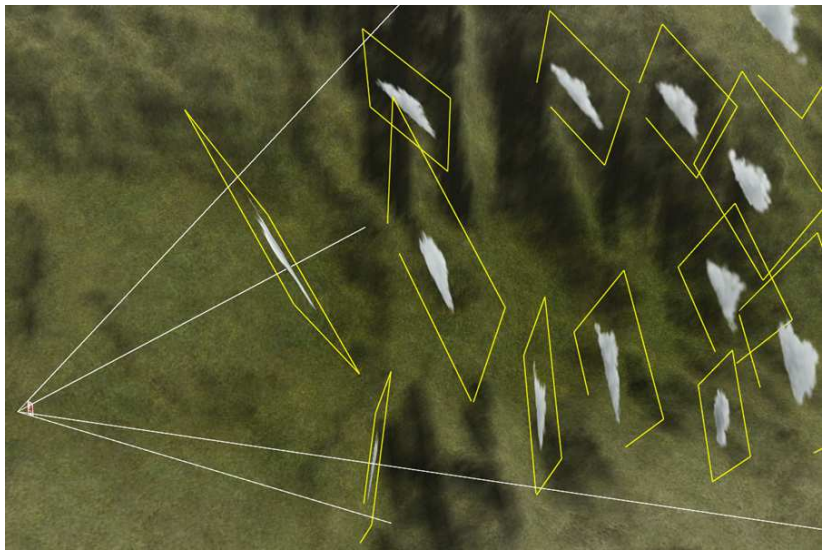
Viewplane vs Viewpoint

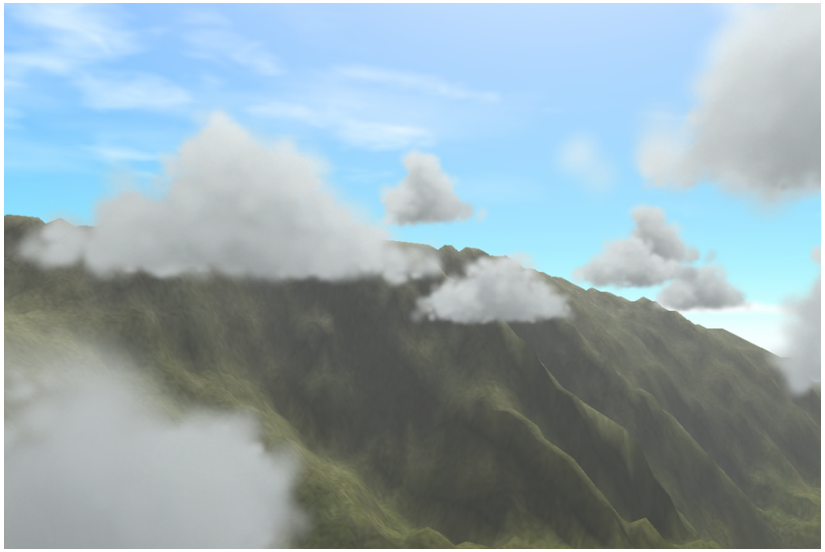


Viewpoint-oriented Billboarding (esempio)



Viewpoint-oriented Billboarding (esempio)

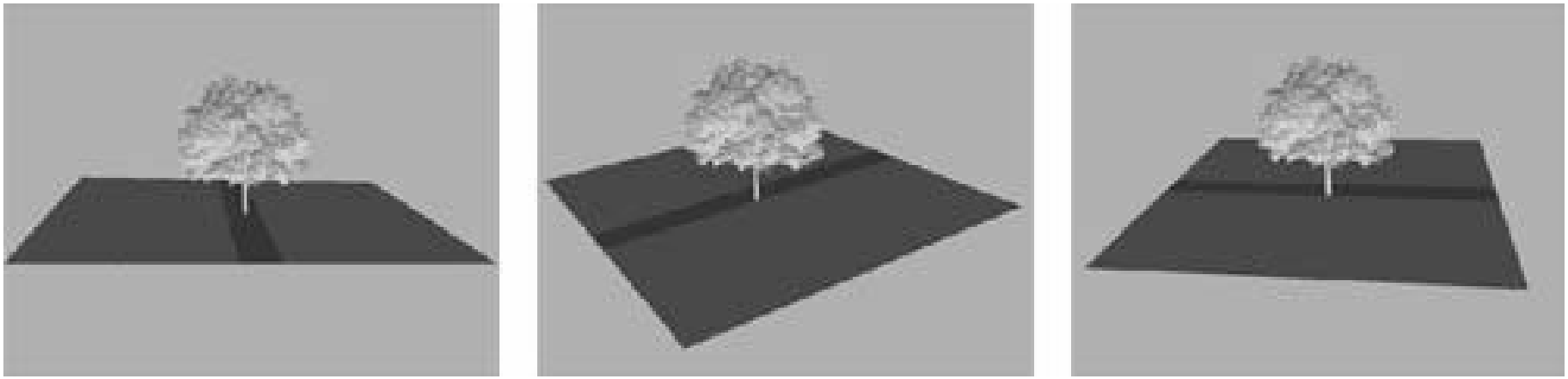




Axial Billboarding

- Questa tecnica è molto utile per rappresentare oggetti a simmetria cilindrica.
- Per esempio è molto usato per visualizzare alberi o raggi laser.
- In questo schema il vettore ***u*** è fisso e coincide con $\boldsymbol{u}_{\boldsymbol{world}}$, e la direzione di vista è utilizzata come vettore da aggiustare.
- In questo caso l'oggetto deve essere ruotato intorno ad un asse fisso fino ad allineare la billboard ***il più possibile*** con l'osservatore.
- Una volta che la billboard è orientata secondo l'osservatore può essere traslata nella sua posizione.

Axial Billboarding



- Vantaggio: effettuo il rendering di un'immagine anzichè di una geometria complessa
- Svantaggio: se osservassi l'albero dall'alto avvicinandomi vedrei quest'ultimo assottigliarsi (pensate ad una carta da gioco)

Billboarding (all-together)

Beam Runner Hyper Game
(http://www.cs.unc.edu/~andrewz/twa/screenshots.html)

Impostor

- Un ***impostor*** è una billboard creata *on-the-fly*
- Si usa per velocizzare il rendering di oggetti complessi multipli, se ad esempio ho più istanze di uno stesso oggetto creo l'impostore e rimpiazzo la geometria al volo
- Utilissimo per oggetti statici e distanti

Impostor

- Per creare un impostor
  - Scelgo un punto di vista appropriato
  - Renderizzo l'impostor su una texture (tenendo conto di settare lo sfondo trasparente)
  - Uso la texture come immagine per la billboard
- Per la visualizzazione dell'impostor solitamente si utilizza la tecnica di billboarding viewpoint-aligned

Blending

- È una tecnica di imaging che ci permette di mescolare i colori di più immagini
- Per la precisioni i colori del framebuffer vengono mescolati con i colori della nuova immagine che si sta generando
- Utilizzo immediato → gestione della trasparenza

Semitrasparenza

vecchia:
(già sul framebuffer)

nuova:
che sovrascrivo

risultato

(1-α) x  + α x  =  α = 0.25

(1-α) x  + α x  =  α = 0.5

(1-α) x  + α x  =  α = 0.75

Alpha Blending

- I colori hanno 4 componenti:
  - R,G,B, α
- Quando arriva un frammento (sopravvissuto al depth test) invece di sovrascriverlo lo miscelo con la formula:

$$(r,g,b)_{finale} = (r,g,b)_{vecchio} \cdot (1-\alpha) + (r,g,b)_{nuovo} \cdot (\alpha)$$

NOTA: Alpha blending e Z-buffer bisticciano... l'ordine di rendering è determinante per il risultato finale.

Alpha Blending & OpenGL

- Per attivare l'alpha blending si utilizza il comando **`glEnable(GL_BLEND);`**
- Per settare la funzione di blending si utilizza il comando **`glBlendFunc(source_factor, dest_factor);`**
  - Esempio:

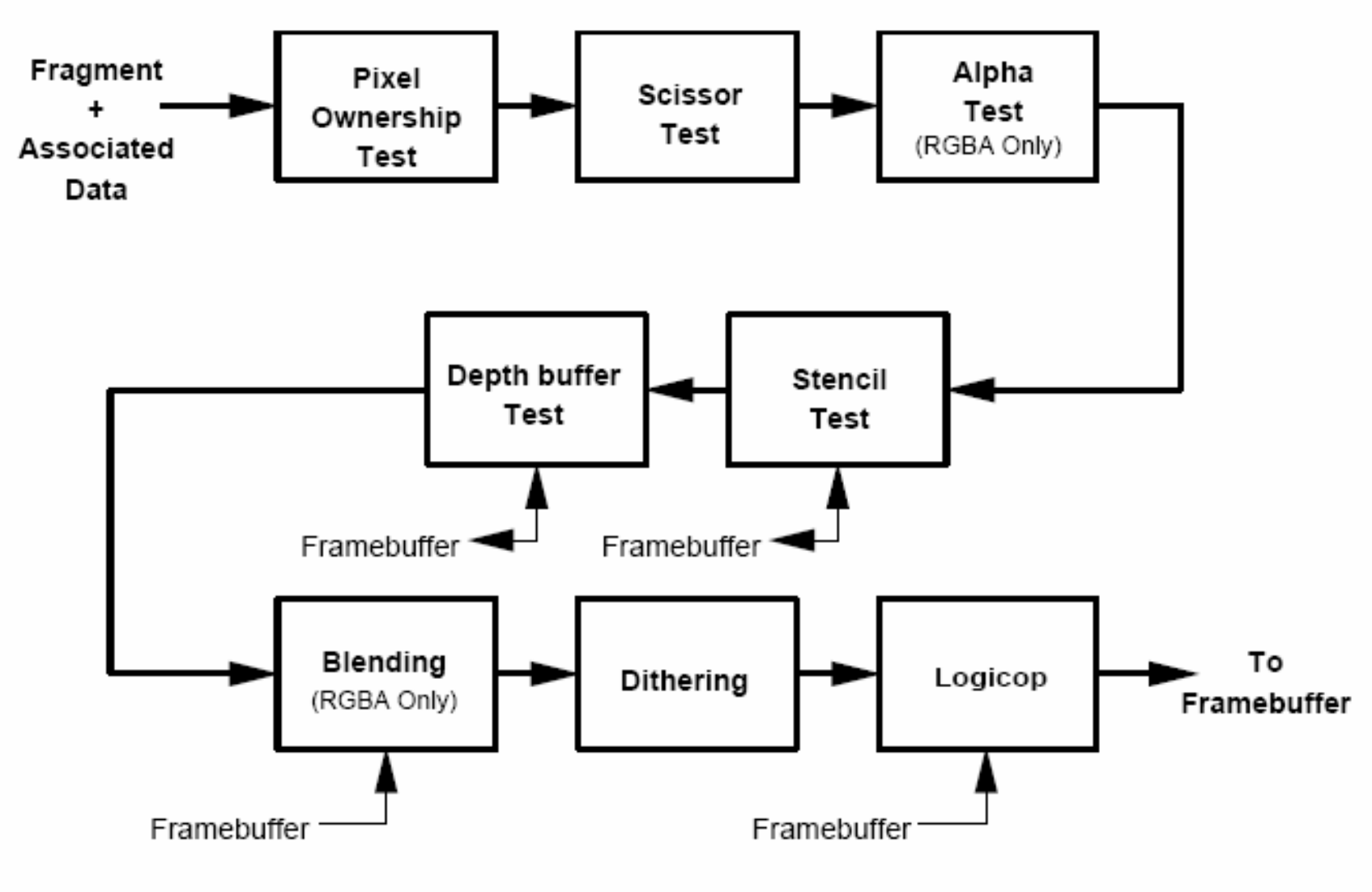
```
glBlendFunc(GL_SRC_ALPHA,
  GL_ONE_MINUS_SRC_ALPHA);
```

- Per *source* si intende l'alpha del frammento, per *dest* l'alpha di destinazione, ossia quello del framebuffer
- Ci sono molte combinazioni possibili per le funzioni di blending:
  - **GL_ZERO, GL_ONE,**
    GL_SRC_ALPHA, GL_ONE_MINUS_SRC_COLOR,
    GL_DST_ALPHA, GL_ONE_MINUS_DST_ALPHA,
    GL_DST_COLOR, GL_ONE_MINUS_DST_COLOR

Per-fragment Operations (intermezzo)

- Una volta uscito in output, il frammento subisce delle operazioni dette ***per-fragment operations*** prima di arrivare al framebuffer
- Alcune di queste operazioni sono
  - Scissor Test
  - Alpha Test
  - Stencil Test
  - Depth Test
  - Blending
  - Dithering

Per-fragment Operations

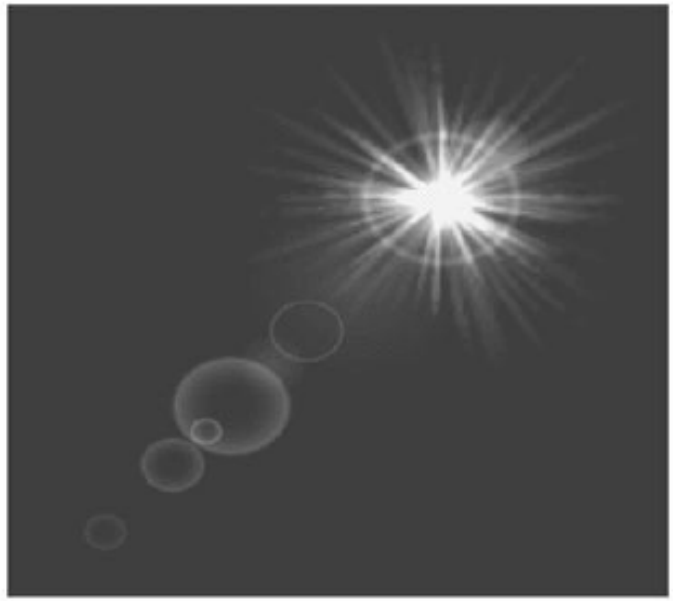


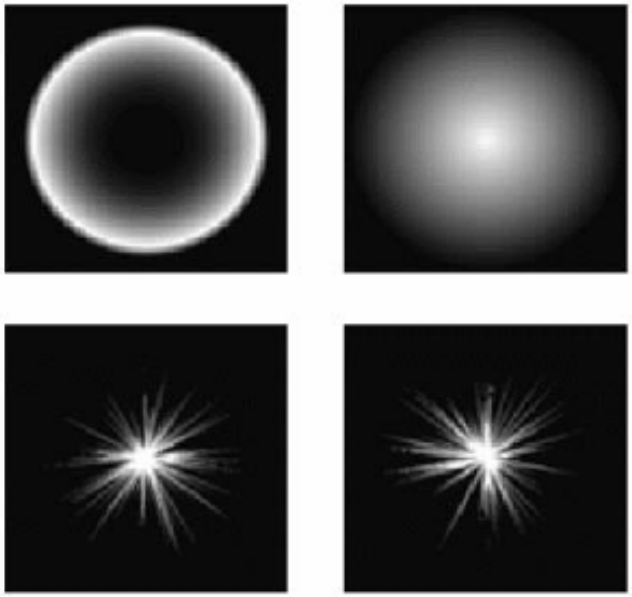
Alpha Test

- Se un frammento é troppo trasparente, scartiamolo:
  - `if ( frammento.alpha < k)`
    `then scarta frammento`
- Al solito, può essere abilitato o disabilitato con i soliti comandi OpenGL glEnable(...), glDisable(...)
  - `glEnable(ALPHA_TEST);`
- Anche la funzione di test può essere settata:
  - `glAlphaFunc(GL_GREATER, 0.01);`

Lens Flares

- Effetto risultante da un insieme di effetti visivi causati dalla riflessioni della luce diretta sulla lente della camera (glare effects)
- Halo – rifrazioni sulla lente
- Ciliary corona – variazioni di densità della lente
- Bloom – dovuto allo scattering, evidente intorno alle zone di forte intensità





Lens Flares

- Tutti quesi effetti (glare effects) possono essere combinati insieme per dare l'illusione di maggiore realismo
- Per il rendering dei singoli effetti si utilizzano delle texture base ed il billboarding (screen-aligned)
- Per l'implementazione potete fare riferimento al tutorial: http://www.gamedev.net/reference/articles/article813.asp

Particle System

- I sistemi particellari vengono utilizzati per renderizzare una grande quantità di effetti quali il fuoco, il fumo, una fontana, un'esplosione, ecc.
- In realtà si tratta di una tecnica di animazione e non di visualizzazione.
- Si parte da un insieme di piccoli oggetti (*punti*, *linee* o *texture di piccole dimensioni*) che poi vengono visualizzati (view-plane aligned billboarding) ed animati a centinaia per ottenere il fenomeno desiderato.

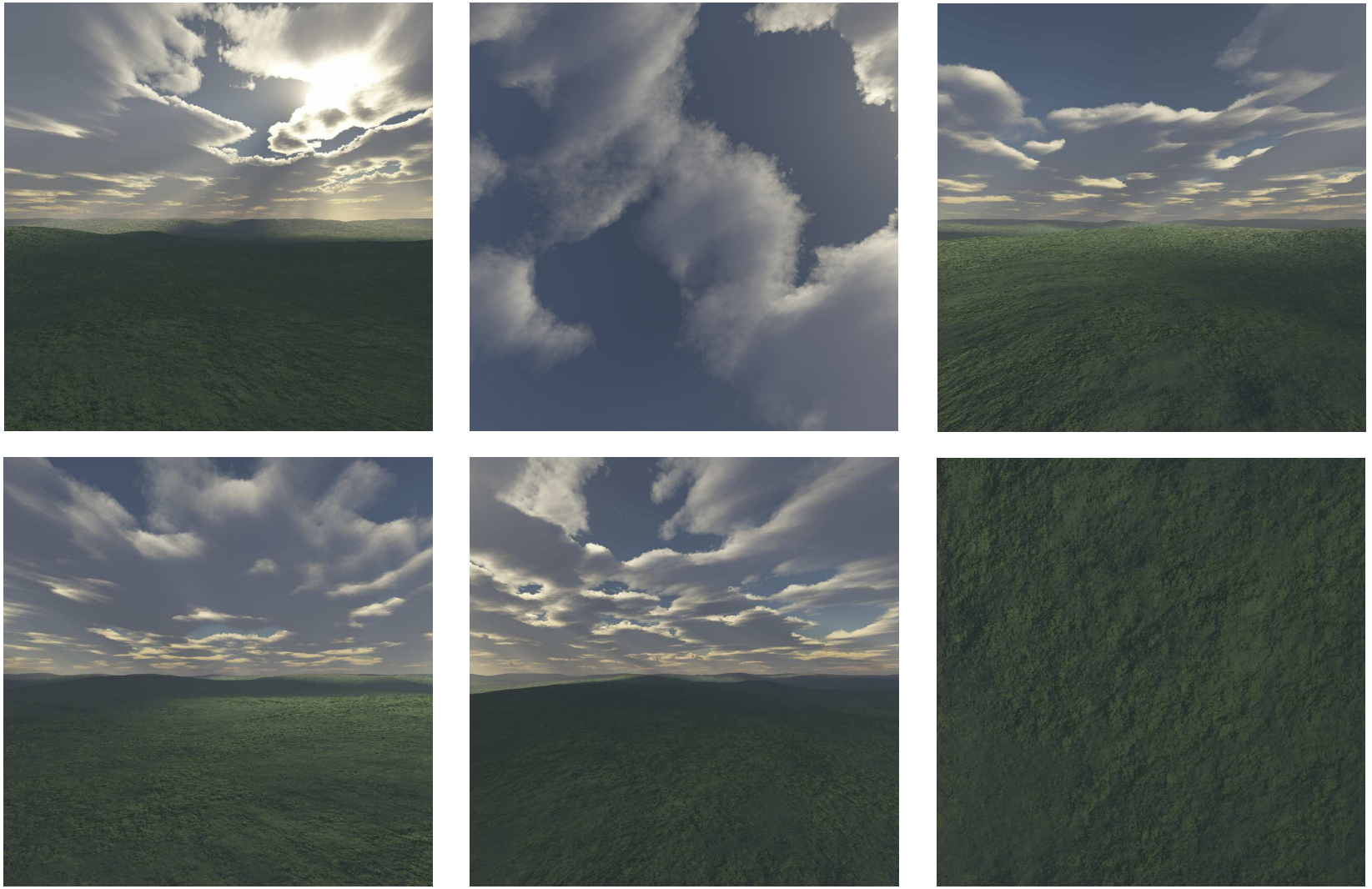
Particle System

- Ogni oggetto ha la sue proprietà di velocità, accelerazione, decadimento, colore che possono essere funzioni delle particelle vicine, funzioni del tempo, ecc. Ad esempio:
  - ***pos(t+1) = f(pos(t), ...)***
  - ***vel(t+1) = f(vel(t), ...)***
  - ***colore(t+1) = f(colore(t), p(t), vel(t), ...)***
- Spesso le particelle si combinano tramite blending
- Vediamo alcuni esempi...

Skybox

- Uno ***skybox*** è un metodo per creare lo sfondo di una scena tridimensionale in modo da dare l'illusione dei dintorni (distanti) della scena.
- Si utilizza il CUBE MAPPING con delle texture opportune che rappresentano gli oggetti distanti nella scena (solitamente cielo, nuvole, montagne, ecc).

Skybox



Domande?