

Lezione 8 27/03/2024

Soft and Fluid Bodies

Fin ora abbiamo parlato di corpi rigidi, ma ci sono anche degli elementi chiamati **soft o fluid bodies** che per loro natura non hanno una struttura rigida, ma sono oggetti deformabili (vestiti, acqua, fumo, fuoco).

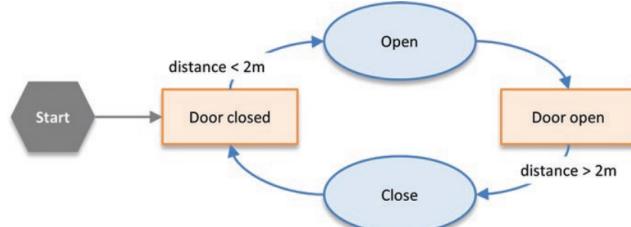
Per questi elementi che sono tendenzialmente animati, non vanno bene quelle keyframe, ma servono animazioni più ricercate. Su questi elementi si usano i **particle systems**, dove alcuni pezzi sono settati ma poi è il game engine ad animarli.

Un altro modo è quello di specificare una **finite state machine**. Con i **behaviour** si possono modellare dei comportamenti che sono reattivi, a seguito di eventi. Nella finite state machine vengono definiti degli stati in cui si può trovare l'oggetto, e vengono definite le transizioni tra gli stati.

Possono essere fatti con uno **switch node** nel **graph scene**.

Possiamo per esempio dire che componenti ha una macchina nello stato normale, e poi uno stato diverso per la macchina incidentata a cui si passa dopo una collisione.

La finite state machine può essere vista come un grafo, gli archi sono le transizioni, i rettangoli sono gli stati e i cerchi sono gli eventi.



Un **trigger** è un evento che causa un cambio di stato in una finite state machine:

- **Timer event**, accade dopo che passa un certo periodo di tempo
- **Touch event**, accade dopo la selezione di un'oggetto dall'utente
- **Proximity event**, quando l'utente si avvicina sotto ad una certa distanza dall'oggetto
- **Visibility event**, quando l'oggetto entra nell'FOV dell'utente

Le finite state machine sono il passo successivo dei keyframe ma possono non essere abbastanza, per esempio per gli avatar umani hanno comportamenti estremamente complessi (si possono fare tramite finite state machine gerarchiche, decision trees, e behavior tree).

Una volta che ho strutturato la finite state machine, come viene integrata nel rendering?

Innanzitutto gli elementi della scena vengono modificati dall'esterno (es da un physics engine, che manda le informazioni al finite state machine che deve capire se c'è un cambiamento di stato). Alcune finite state machine hanno un supporto nativo per le animazioni con keyframe, con dei nodi speciali.

Light, sound, background

Senza una fonte di **luce**, il virtual world sarebbe scuro. Vediamo la luce perché è riflessa dagli oggetti. Abbiamo vari tipi di luci.



Directional light models a very far away or even infinitely distant light source, whose rays arrive in the virtual world in parallel directions (ex: sun)



Point light (often called **omni light** too) source emits light spherically in all directions (like a light bulb)



Spot light sources produce a light cone



Area lights emit light from a 2D rectangular area, in one direction only



Headlights moves along with the viewer (a directional light source whose direction is aligned with the viewing direction)

Per le **point e spot light**, la luce decresce con la distanza, ci sono quindi dei parametri che possono essere scelti.

Per la **directional light** invece l'attenuazione con la distanza non ha senso, è il caso del sole.

Si potrebbe fare un'attenuazione in base a particelle presenti nell'aria (atmospheric attenuation).

Per le **spotlights** c'è attenuazione anche ai bordi del cono di luce.

Le **area lights** sono più realistiche ma sono molto più pesanti computazionalmente.

Suoni

Si può giocare con i **suoni di background**, che vengono uditi in qualunque punto dell'ambiente.

Ci sono i **spatial audio sources** che sono emettitori di suoni, ovvero punti o oggetti nel mondo che emettono il suono, che quindi diventa più delimitato nello spazio, più alto quando ci si avvicina e più basso quando ci si allontana. Si può anche fare un suono 3D per sentire la direzione del suono.

C'è quindi anche l'**attenuazione acustica**, ovvero quanto si attenua il suono quando ci si allontana dalla fonte.

Nel background sound non viene quindi definita l'attenuation.

Backgrounds

Il **background** è lo sfondo come le montagne in lontananza, il cielo, un qualcosa che è intorno alla scena ma **non è un elemento tangibile**. Viene quindi usata un'immagine statica a 360°, oppure una semisfera o un cubo intorno alla scena.

è un volume, e il centro di questo volume è sempre la viewpoint dell'utente, **non cambia la distanza**.

Si può ruotare la sfera del cielo per simulare il movimento di nuvole.

Special 3D objects

Ci sono altri oggetti particolari come i virtual humans, i particle systems, il terreno e la vegetazione.

Questi sono spesso creati fuori da altri tool e importati, anche per le animazioni.

Virtual Humans

Con il termine **avatar** si parla di un virtual human che è un proxy dell'utente, il corpo virtuale dell'utente.

Gli altri virtual human sono detti **bot**, o **NPC** (non-player characters), che sono fuori dal controllo dell'utente, controllati dal game engine.

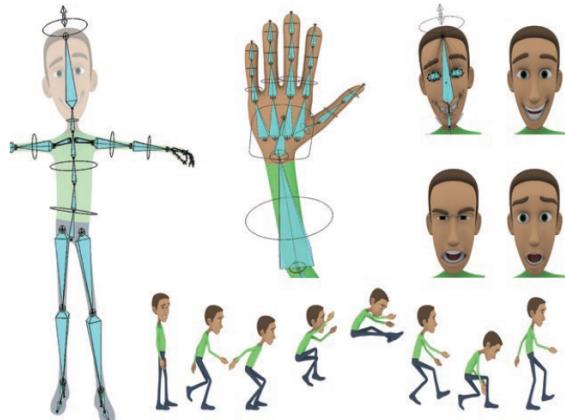
Il realismo è molto importante, come si muove, come parla, come reagisce, il suo aspetto.

Ci sono due approcci per l'aspetto, si usano **primitive** (sfere come testa, rettangolo come corpo, etc), che però rimane molto irrealistico.

Altrimenti con le **skeleton-based**

animation viene creato lo scheletro, e poi si aggiunge la pelle. Il rigging è il processo di scelta di una struttura scheletrica, i giunti, i constraints. Lo **skinning** è il processo di accoppiare i vertici della pelle con le ossa dello scheletro.

C'è una parte ad hoc legata al viso, che è ancora più complicata e più dettagliata. Bisogna anche rappresentare le emozioni con le espressioni facciali.



Le animazioni sono spesso create tramite **motion capture**.

Vengono usati metodi di inverse kinematics per computare le posizioni scheletriche per esempio degli arti, delle mani, di modo che siano nella posizione corretta.

Particle system

I particle systems sono quei sistemi dove abbiamo un insieme di particelle che si muovono, per esempio il fumo, sono fenomeni gestiti in maniera più flessibile.

Ogni particella è detta una **point mass**, ovvero un elemento che ha una massa ma non ha uno spazio. Per esempio c'è un punto da cui parte l'emissione di particelle (si può scegliere quante ne vengono emesse al secondo, se perdono massa man mano e quindi spariscono), il resto è gestito dal physical engine.



Sono quindi gestiti partendo dall'**emitter**. Gli emitter possono variare per direzione, parametri come il numero, il punto di partenza, forza di emissione, etc.

Queste particelle sono mostrate come poligoni con texture, usando quadrilateri che sono allineati al punto di vista dell'utente (chiamati **billboards**). Altri approcci sono l'utilizzo di segmenti lineari, geometrie più o meno complesse.

Questi oggetti sono molto computazionalmente pesanti per il rendering real time.

Il colore e la texture possono cambiare nel tempo.

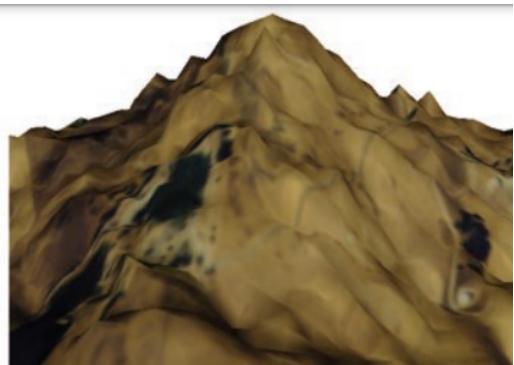
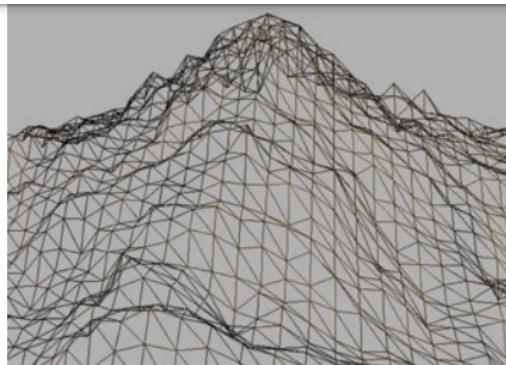
Terreni

Si può voler modellare anche il terreno, se serve avere l'elemento 3D (al posto di un'immagine background finta).

Si può usare una mesh di poligoni.

Si può creare una mappa 2D dove in ogni punto andiamo a specificare l'altezza (quelli più importanti, lasciando al tool la creazione degli altri). In questo modo non si può modellare perfettamente una montagna, è un metodo semplificato 3D.

Se bisogna utilizzare un terreno molto ampio su cui l'utente potrà andare, si utilizza il concetto di tiles, dividendo il terreno in pezzi, aumentando i poligoni nelle tiles intorno/di fronte all'utente.



Vegetazione

Gli alberi e la vegetazione aggiungono realismo se fatti bene, sono difficili da modellare e pesanti da gestire.

Qui si applica lo step procedurale con tool esterni, partendo dal tronco, si usano degli algoritmi per aggiungere dei rami più piccoli, e infine si aggiungono le foglie.

Le foglie di solito sono approssimate in piccoli rettangoli, con una texture di foglia, magari giocando sulla trasparenza per rendere l'effetto buono da lontano. Anche qui si può aumentare il livello di dettaglio in base alla distanza dell'utente.

