

Costruzione di Interfacce
Lezione 30
MMeditor e collision detection

cignoni@iei.pi.cnr.it
<http://vcg.isti.cnr.it/~cignoni>

Riordiniamo

- ❖ Vogliamo tenere condiviso un po' di codice
- ❖ Directory principale mm
- ❖ Con le subdir vcg, CI e MM con il codice condiviso
- ❖ Due dir diverse per le due app
 - ❖ Qtmm per l'editor
 - ❖ Sdlmm per il player
- ❖ Rifare a mano tutti I pro,
- ❖ cambiamo qualche nome per coerenza

Gestire fisica e dinamica

- ❖ Il nostro scene graph deve essere in grado di evolversi nel tempo autonomamente
 - ❖ Dinamica != grafica
 - ❖ Non e' detto che siano la stessa cosa da fare nello stesso tempo
 - ❖ Scollegare draw dal tempo
 - ❖ Tenere lo stato della scena (anche quello legato al tempo) dentro lo scene graph e aggiornarlo indipendentemente dal disegno

Collision Detection

- ❖ La gestione delle collisioni rientra nel più generale problema di simulare la *fisica* del nostro environment ad un adeguato livello di accuratezza:
 - ❖ livello 1:
 - ❖ Assicurarsi che il personaggio guidato dal giocatore non attraversi liberamente tutta la scena
 - ❖ livello 2:
 - ❖ Tutti gli oggetti mobili della scena si interagiscono correttamente con l'ambiente
 - ❖ livello 3:
 - ❖ Tutti gli oggetti mobili interagiscono correttamente anche tra di loro.

Collision Detection: Livello 1

- ❖ Distinzione a livello di scene graph:
 - ❖ tra ambiente e giocatore
 - ❖ Distinzione tra ambiente da controllare e ambiente visualizzato
 - ❖ Il numero di controlli che si fa per frame è lineare con la grandezza della scena
 - ❖ Ottimizzando diventa logaritmico

Approssimare!

- ❖ Il giocatore con una sfera
 - ❖ Nella maggior parte dei casi la differenza non si nota.
- ❖ L'ambiente da testare con pochi poligoni (o con poche sfere)
- ❖ Il test sfera poligono è abbastanza semplice

Vincolare!

- ❖ Ridurre se possibile il problema a 2 dimensioni
 - ❖ Ad esempio in molti giochi first person perspective o racing
- ❖ Ridurre la libertà di movimento
 - ❖ Lo spazio dove si muove il player non è quello cartesiano ma è un insieme discreto di posizioni (e.g. in tutti i giochi di labirinto tipo pacman)

Costruzione di Interfacce - Paolo Cignoni

7

Formule di base

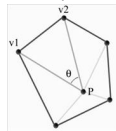
- ❖ Piano (o meglio semispazio)
 - ❖ \mathbf{n} Normale al piano
 - ❖ d distanza dall'origine
- ❖ Per i punti x sul piano vale:
 - ❖ $\mathbf{n} \cdot \mathbf{x} = d$
- ❖ Distanza (con segno) punto p piano pl
 - ❖ $pl \cdot \mathbf{n} \cdot p - pl \cdot d$
- ❖ Proiezione p' di un punto p sul piano pl
 - ❖ $k = pl \cdot \mathbf{n} \cdot p - pl \cdot d$
 - ❖ $p' = p - pl \cdot \mathbf{n} \cdot k$

Costruzione di Interfacce - Paolo Cignoni

8

Formule

- ❖ Con le formule precedenti è facile sapere il punto più vicino (di contatto?) tra una sfera ed un piano.
- ❖ Per passare a poligoni (triangoli) basta fare il test punto in poligono
- ❖ Molti metodi diversi più o meno efficienti, il più facile da implementare:
 - ❖ la somma degli angoli dal punto ai vertici del poligono è 2π se e solo se sono dentro al poligono.



Costruzione di Interfacce - Paolo Cignoni

9

Intersezione raggio triangolo

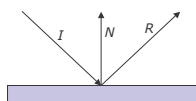
- ❖ Il metodo precedente non è il migliore
- ❖ In effetti quello che spesso serve è il test di intersezione raggio triangolo
- ❖ Questo perchè si vuole sapere se nel prossimo intervallo di tempo lungo la traiettoria corrente colpisce qualcosa.
- ❖ Il metodo che trovate implementato qui è uno dei più efficienti
 - ❖ Tomas Möller and Ben Trumbore:
"Fast, Minimum Storage Ray-Triangle Intersection"
 - ❖ <http://www.acm.org/jgt/papers/MollerTrumbore97/>

Costruzione di Interfacce - Paolo Cignoni

10

Calcolo della risposta

- ❖ Una volta che sappiamo che abbiamo colpito una superficie dobbiamo calcolare la risposta.
- ❖ Il minimo: $R = I - 2 \cdot (I \cdot N) \cdot N$



Costruzione di Interfacce - Paolo Cignoni

11

Ottimizzazioni

- ❖ Uso di gerarchie di bounding objects semplici (sfere o box) che approssimino in maniera sempre migliore gli oggetti da testare



Costruzione di Interfacce - Paolo Cignoni

12

Collision e Scene Graph

- ❖ In genere conviene integrare tutto assieme. Ogni nodo che contiene geometria (compresi i gruppi) dovrebbe saper rispondere, efficientemente, alle seguenti query
 - ❖ Bound object
 - ❖ Usato per costruire un bound object per ogni gruppo e limitare la discesa nelle gerarchie
 - ❖ Collision con un punto/sfera/raggio
 - ❖ Restituendo punto di collisione e normale
- ❖ In questo modo si sfrutta la gerarchia dello scene graph e si permette ai singoli nodi di sfruttare la propria conoscenza interna per rispondere efficientemente.