SPACE SHOOTER INTERFACCE E RIGIDBODY

CORSO DI GAME PROGRAMMING
1° ANNO

Docente **Davide Caio**

ACCADEMIA ITALIANA VIDEOGIOCHI

EREDITARIETÀ vs INTERFACCE

Quando usare una classe e quando usare una interfacce?

Il mio approccio è questo:

- quando delle entità condividono delle caratteristiche intrinseche, cioè quando quelle entità SONO tutte una stessa categoria, ma ognuna è anche diversa dalle altre, allora si definisce una classe base per quella categoria e ogni entità è una classe che deriva da quella categoria (animale → gatto, cane, uomo)
- quando delle entità condividono un COMPORTAMENTO, cioè sono entità del tutto distinte e di categorie completamente diverse ma condividono un comportamento, allora quel comportamento deve essere definito attraverso una interfaccia (nemico, proiettile, player, una porta possono tutte subire danno, ma non hanno niente in comune. Si crea una interfaccia IDamagable).

SPACE SHOOTER

Con in mente la distinzione fatta nella slide precedente, e avendo in mente come funziona il nostro engine per ora, esistono delle funzionalità che comuni a diversi tipi di oggetti (a livello di engine, non di gameplay) che potrebbero essere definiti attraverso una interfaccia?

IUpdatble e IDrawable

Esistono degli oggetti che devono eseguire delle azioni ogni frame e devono essere disegnati ogni frame.

Potremmo introdurre queste due funzionalità come due interfacce che le varie classi possono implementare.

Quale/Quali metodi bisogna definire all'interno di queste due interfacce?

Game Engine - Motore grafico e Motore fisico.

Motore grafico → Motore che gestisce quello che si vede.

Motore fisico → Motore che gestisce la fisica del gioco.

La cosa principale che dobbiamo capire è che non tutto quello che vede il motore grafico è visto anche dal motore fisico e viceversa.

Esempio?

- La GUI di un gioco molto spesso esiste solo nel motore grafico.
- Il background esiste solo nel motore grafico.
- Barriere “invisibili”, cosa sono? Sono oggetti che esistono solo nel mondo fisico (da evitare perché fanno molto anni 90. A livello di game design è sempre meglio contestualizzare il perché il giocatore non può andare in quell’area/direzione).

Game Engine - Motore grafico e Motore fisico.

Quando un oggetto appartiene al motore grafico?
La risposta è abbastanza semplice, quando ha una sprite (o più in generale quando possiede un componente che renderizza).

Quando un oggetto appartiene al mondo fisico?
Per rispondere a questa domanda dobbiamo prima chiederci il perché di queste distinzioni.

Il problema è prestazionale. Nel mondo reale, non esiste questa distinzione. Se noi abbiamo un oggetto dalla forma molto complessa e lo lanciamo verso un muro, rimbalzerà in un certo modo contro il muro e finirà a terra. Anche a terra rimbalzerà/rotolerà in un certo modo che è dato esattamente dalla sua forma.

Game Engine - Motore grafico e Motore fisico.

In una simulazione questo tipo di comportamento è molto time consuming più la forma dell'oggetto è complicata.

Per questo si cerca sempre di approssimare al meglio la forma dell'oggetto con una forma meno complessa e su cui è più facile rilevare collisioni. Gli engine moderni hanno molte forme diverse, (fino a quella che segue passo per passo quello che si vede), ma non per questo bisogna sempre scegliere quella più precisa. La soluzione ideale è ottenere un giusto trade off tra precisione e computazione.

I COMPONENTI DEL MOTORE FISICO

Per il nostro gioco creeremo due componenti del motore fisico:

- Rigidbody: che introduce l'oggetto su cui è collegato alla simulazione fisica.
- Collider: che indica la forma dell'oggetto per il motore fisico.

In particolare creeremo due tipi di collider:

- CircleCollider: un collider circolare.
- BoxCollider: un collider rettangolare.

Problematica:

- Bisogna capire come rilevare le collisioni tra le diverse forme (circle vs circle, rect vs rect, circle vs rect).

INHERITANCE vs COMPOSITION

Il problema:

Voglio creare le seguenti entità nel mio gioco:

- Entità solo nel motore grafico
- Entità solo nel motore fisico
- Entità sia nel motore grafico che nel motore fisico.

Utilizzando l'ereditarietà come posso fare?

E se si aggiungono altri due possibili comportamenti, le combinazioni quante diventano?

INHERITANCE vs COMPOSITION

Detta semplicemente (in futuro un engine full component costruito sopra Aiv.Fast2D)

Invece di creare classi che ereditano e specificano un comportamento aggiuntivo (con la necessità di creare classi che hanno lo stesso comportamento meno uno, e tutte le possibili combinazioni), sarebbe comodo avere un oggetto al quale si può “attaccare” una funzionalità. Questa funzionalità è descritta in un componente che viene aggiunto all’oggetto “master”. In questo modo un oggetto “master” con diversi componenti attaccati avrà un determinato comportamento, e attaccando diversi componenti ad un altro avremo un altro diverso comportamento.

Noi vedremo una soluzione mixed in questo progetto.