SPACE SHOOTER COLLISIONI E BITMASK

CORSO DI GAME PROGRAMMING
1° ANNO

Docente **Davide Caio**

ACCADEMIA ITALIANA VIDEOGIOCHI

COLLISIONI CERCHIO vs CERCHIO

Dati due cerchi:

- Cerchio 1 di centro C1 (x1, y1) e raggio R1
- Cerchio 2 di centro C2 (x2,y2) e raggio R2

I due cerchi collideranno tra loro quando la distanza tra C1 e C2 sarà minore di R1 + R2

Più performante: la distanza tra C1 e C2 al quadrato è minore della somma dei raggi al quadrato.

RICORDA: l'operazione di radice quadrata è molto dispendiosa in termini computazionali. Ogni volta che è possibile è sempre meglio evitarla.

COLLISIONI RETTANGOLO vs RETTANGOLO

Dati due rettangoli NON ruotati:

- Rect 1: x1,y1,W1,H1
- Rect 2: x2,y2,W2,H2

si intersecano se la distanza tra le due coordinate X è minore della somma della metà delle due width e se la distanza tra le due coordinate Y è minore della somma della metà delle due height.

DeltaX = Math.Abs(x1 - x2)
DeltaY = Math.Abs(y1 - y2)

DeltaX <= W1/2 + W2/2 && DeltaY <= H1/2 + H2/2

COLLISIONI CERCHIO vs RETTANGOLO

Dati:

- Cerchio 1 di centro C1 (x1, y1) e raggio R1
- Rect 2 di centro C2 (x2,y2), width W2 e height H2

Quando queste due figure si intersecano?

Tips:

- Per sapere se un punto interseca una circonferenza basta controllare la distanza tra il centro del cerchio e il punto stesso sia minore della circonferenza.
- Per il rettangolo come possiamo fare?

COLLISIONI CERCHIO vs RETTANGOLO

- L’intersezione tra cerchio e rettangolo si ha quando il punto del rettangolo più vicino al cerchio dista dal centro del cerchio meno del raggio del cerchio stesso.

Quindi:

NearestX = Max (x2 - W2/2, Min (x1, x2 + W2/2))
NearestY = Max (y2 - H2/2, Min (y1, y2 + H2/2))

Ma come si trova il punto del rettangolo più vicino al cerchio?

COLLISIONI CERCHIO vs RETTANGOLO

Il punto del rettangolo più vicino al cerchio si trova attraverso questa formula:

- NearestX = Max (x2 - W2/2, Min (x1, x2 + W2/2)) → *massimo tra il punto più a sinistra del rettangolo e il minimo tra il centro del cerchio e il punto più a destra del rettangolo*
- NearestY = Max (y2 - H2/2, Min (y1, y2 + H2/2)) → *massimo tra il punto più in alto e il minimo tra il centro del cerchio e il punto più in basso del rettangolo (ATTENZIONE: i segni sono invertiti perché la Y cresce verso il basso nel nostro engine).*

La formula totale è quindi:
DeltaX = x1 - NearestX
DeltaY = y1 - NearestY

E la formula per l'intersezione si ottiene con il solito pitagora (togliendo la radice quadrata):
(DeltaX * DeltaX + DeltaY * DeltaY) < (R1*R1)

PHYSICS MANAGER

L’ultimo problema che ci rimane da affrontare è come poter dire se due oggetti possono collidere tra loro. è inutile rilevare la collisione nel engine e poi ignorarla se non ci interessa.

Vogliamo cercare di poter dire all’engine quali oggetti possono collidere con quali altri oggetti. E se l’engine deve controllare la collisione tra quei due oggetti semplicemente passa oltre. Risparmiamo il tempo di calcolo della collisione.

Quale metodo utilizzereste?

BITMASK

Una bimask è un dato utilizzato esclusivamente per effettuare operazioni bitwase su di esso.

Operatori principali:

- & operatore di end bit per bit
- | operatore di or bit per bit
- |=

Perché è molto comodo utilizzare le bitmask? Perché sono molto veloci e ci possono aiutare di associare a ogni bit della maschera un valore logico importante ai fini del nostro software.

L'esempio più comune nel videogioco è il sistema fisico. è possibile creare più layer fisici (nel nostro esempio RigidBodyType) di interazione dei nostri oggetti e prima di effettuare il check della collisione basterà controllare che quei due particolari layer possono tra loro interagire fisicamente.

BITMASK - ESEMPIO

```
enum PhysicLayer {layer_A = 1, layer_B = 2, layer_C = 4, layer_D = 8}


class RigidBody
{
	protected uint collisionMask;
	public PhysicLayer Layer;

	public AddInteractableLayer (PhysicLayer layer) {
		collisionMask |= (uint) layer;
	}

	public bool CanInteract (PhysicLayer layer) {
		return ((uint) layer & collisionsMask) != 0;
	}
}
```