COMPOSIZIONE 2

CORSO DI GAME PROGRAMMING
1° ANNO

Docente **Davide Caio**

IL PROBLEMA

La nostra struttura prevede che un GameObject è formato da più componenti.

Problema: Come possiamo fare per, dato un gameobject, avere la reference di un componente (se esiste) di tipo specifico attaccato ad esso?

Esempio: come faccio dato un GameObject A ad ottenere il suo SpriteRenderer (ammesso che ne ha uno attaccato?)

REFLECTION

La **reflection** fornisce oggetti (di tipo Type) che descrivono assembly, moduli e tipi. È possibile usare la reflection per creare in modo dinamico un'istanza di un tipo, associare il tipo a un oggetto esistente oppure ottenere il tipo da un oggetto esistente e richiamare i metodi o accedere ai relativi campi e proprietà.

Type: Classe che descrive assembly moduli e tipi.
Activator.CreateInstance → ti permette di creare un'istanza di quel tipo a run time.

GetType () → metodo che ti permette, dato un qualsiasi object, ti ritornare l'istanza di tipo Type che rappresenta il suo tipo.

typeof (nome classe) → ti ritorna un oggetto di tipo Type della classe inserita come parametro.

ADD COMPONENT E GET COMPONENT

Creiamo quindi un override di AddComponent che sfrutta la Reflection, e un metodo GetComponent che ritorna la reference ad un componente di un certo tipo (oppure null) attaccato al gameobject.

FIND GAMEOBJECT

Un’altra funzionalità che mi piacerebbe avere nel mio engine è la possibilità di ottenere una reference a un gameobject nella mia scena filtrandolo per nome.

Aggiungiamo quindi un metodo FindGameObject all’interno della classe scene.

KEEP INSIDE SCREEN

Correggiamo il Keep Inside Screen

MOTORE FISICO

Come abbiamo detto ogni cosa sarà un componente, quindi anche i Collider e Rigidbody saranno Componenti (dell'Engine e quindi erediteranno direttamente da Component).

Collider: sarà esattamente come in SpaceShooter, ma al posto di avere una reference al Rigidbody per saperne la posizione in realtà dovrà solamente utilizzare la reference alla transform che gli arriva direttamente da Component. *I BoxCollider e CircleCollider invece rimarranno invariati*.

Rigidbody: era già stato costruito come "component" in space shooter ma non aveva la classe padre. Basterà quindi farlo ereditare da Component e da *IFixedUpdatable*. La sua Update sarà sostituita con la FixedUpdate.

Il **PhysicsManger** poi avrà una lista di tutti i Rigidbody che avranno il metodo FixedUpdate che sposterà la transform in base alla velocità.

PROBLEMA

Il Rigidbody creava il suo collider, mentre adesso saranno due componenti completamente staccati.

Il Rigidbody ha bisogno di una reference del suo collider (perché ne definisce la forma nel mondo fisico). Come possiamo quindi ottenere una reference al Collider dal Rigidbody se non sappiamo quando è stato creato un Rigidbody se c'è il Collider e viceversa?

RIADATTIAMO PLAYER CONTROLLER

Come lo riadattiamo il player controller adesso che c’è il motore fisico?

BULLET

Nella versione non a componenti il Bullet era una classe astratta che derivava da GameObject, aveva una Sprite, un Rigidbody e un Collider.

Ora invece sarà un Component da attaccare a GameObject che hanno un component SpriteRenderer, un Rigidbody e un Collider.

Avremo solo due divisioni:

PlayerBullet
EnemyBullet

Vedremo come implementare tutte le versione in queste sole due sottoclassi.

BULLET MANAGER

Nella versione non a componenti era un array di Queue. Ora vediamo una versione di con un array multidimensionale, in cui l'IsActive di gameobject è il parametro secondo cui filtriamo i proiettili disponibili (quindi quelli non attivi) rispetto quelli disponibili. → un po' più alla unity like :)

La Init **ritornerà** una lista di gameobject che saranno tutti i proiettili creati.
Ci sarà un metodo per creare ogni tipo di bullet:

- BlueBullet
- GreenGlobe
- FireGlobe
- IceGlobe

La lista ritornata da Init sarà aggiunta alla lista dei GameObject nella scena.

SHOOT MODULE

Ora creiamo un component per sparare!