6. Detecció alternativa del pas a estat kill

6.0 Recapitulació sobre com gestionem fins ara la mort del Player...

El Player mor quan col·lisiona amb l'objecte buit **DeathTrigger**, que està situat sota la plataforma. Aquest objecte té un component **collider 2d** que és un **Trigger** i això fa que, en col·lisionar, s'executi **OnTriggerEnter2D**, codificat a l'script **DeathTriggerScript**, que està associat a l'objecte DeathTrigger:

```
void OnTriggerEnter2D( Collider2D collidedObject )
	{
		Debug.Log ("hitDeathTrigger");
		collidedObject.SendMessage("hitDeathTrigger");
	}
```

El tractament és simplement enviar un **missatge al Player** (que és el collidedObject, l'objecte col·lisionat) amb el text **"hitDeathTrigger"**.

Perquè **el Player** pugui reaccionar a aquest missatge, **ha de tenir un mètode anomenat "hitDeathTrigger"**, on es farà el que toqui. Hem codificat aquest mètode a l'script PlayerStateListener d'aquesta manera:

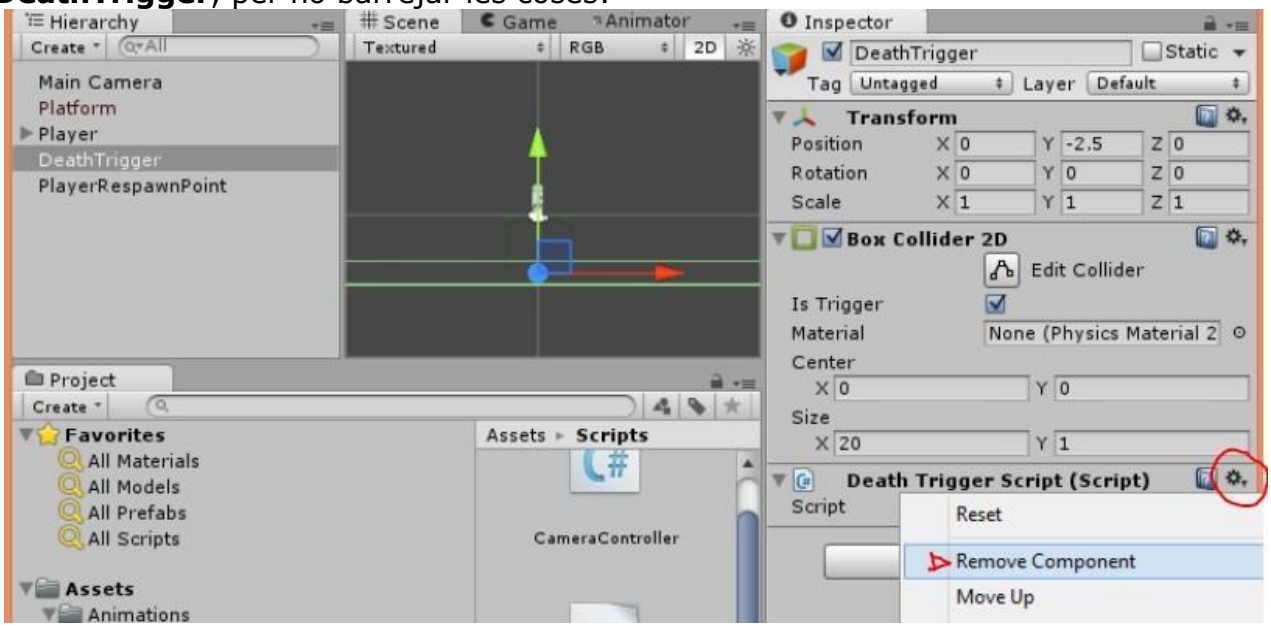
```
public void hitDeathTrigger()
	{ onStateChange(PlayerStateController.playerStates.kill);
	}
```

És a dir, simplement passem a estat **kill** i deixem que el mecanisme d'estats que estem implementant s'encarregui de fer el que s'hagi de fer.

Aquest sistema funciona bé i és un **exemple d'una manera de gestionar** les col·lisions. **Ara anem a veure'n una altra d'alternativa**.

6.1 Gestionar la mort del Player amb l'objecte SceneryToggler

Primer de tot, **desconnectem l'script DeathTriggerScript de l'objecte DeathTrigger**, per no barrejar les coses:



Ara programem el tractament de la col·lisió del Player amb el DeathTrigger a l'script **PlayerColliderListener** on, fins ara, només detectavem el contacte del Player amb la Plataforma.

Simplement afegim

```
case "DeathTrigger":
      // El Player ha caigut sobre el DeathTrigger. El matem.
   targetStateListener.onStateChange(PlayerStateController. playerStates.kill);
break;
```

Prèviament, hem assignat el valor "DeathTrigger" al tag de l'objecte DeathTrigger.

Nota: es proposa afegir la línia remarcada, per provar d'escriure missatges a la consola:

```
case "DeathTrigger":
                  // El Player ha caigut sobre el DeathTrigger. El matem.
Debug.Log("Col.lisio amb DeathTrigger. Tag: " + collidedObject.tag);
targetStateListener.onStateChange(PlayerStateController. playerStates.kill);
break;
```

La finestra de consola es pot obrir des del menú principal **Window > Console**

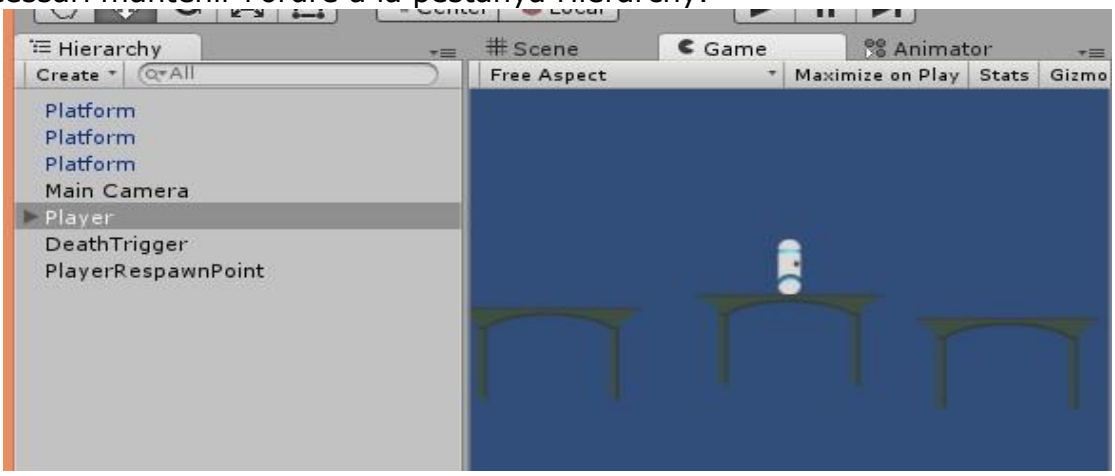
7. Ampliar el món del joc. Prefabs.

Anem a afegir més plataformes per poder-hi saltar. Com que totes les plataformes seran iguals, és adequat definir un **objecte estàndard** del qual proguem crear còpies. En Unity, aquest motlle d'objecte s'anomena **Prefab**.

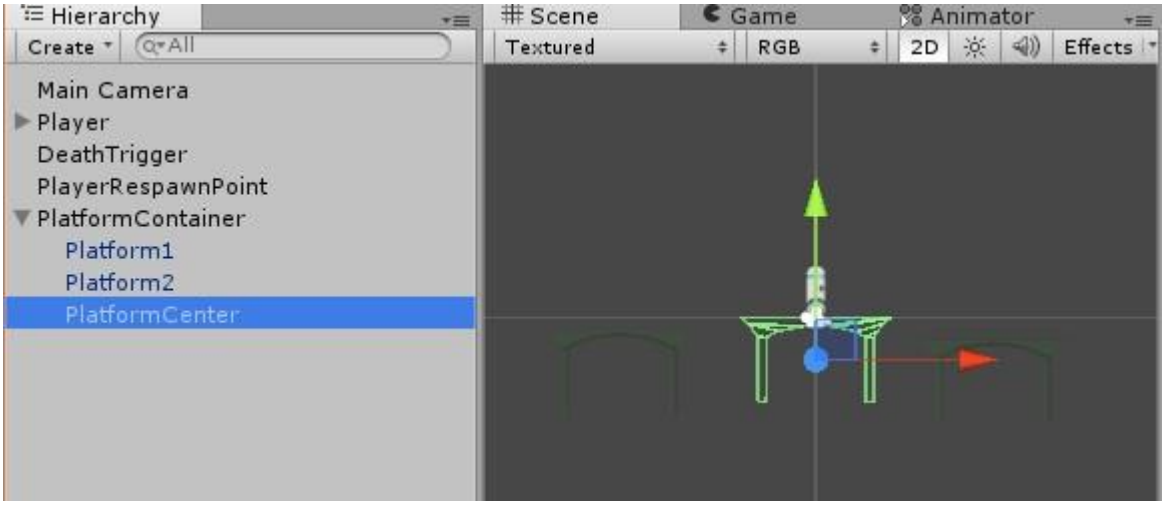
A la pestanya de projecte, dins de la carpeta Assets creem un nova carpeta **Prefabs**. Ara arrosseguem la plataforma des de la pestanya Hierarchy fins a la nova carpeta Prefabs. Amb això ja hem creat un nou Prefab de tipus Platform.

A partir d'ara per afegir plataformes al joc, simplement les arrosseguem des de la carpeta Prefabs. Un avantatge addicional de treballar amb Prefabs és que, si canviem alguna característica en el Prefab, automàticament el canvi es propaga a totes les còpies que s'estiguin utilitzant al joc.

Quan comencem a afegir plataformes veiem que, si el nombre d'objectes creix, és necessari mantenir l'ordre a la pestanya Hierarchy.



Per això creem un nou objecte buit, que anomenem PlatformContainer i hi arrosseguem totes les plataformes dins, per tenir-les ordenades. Les hi canviem el nom si volem.



Activitats

Fer un llistat amb els conceptes de introduits en aquest document, indicar la pàgina i explicar què són i per a què s'utilitzen.