**MOBILE APPLICATION DEVELOPMENT**



**PLATFORM GAME DEVELOPED IN UNITY: SPECIFICATION REQUIREMENTS**

Carlos Martínez de la Riva Valdaliso

C115875

Introduction

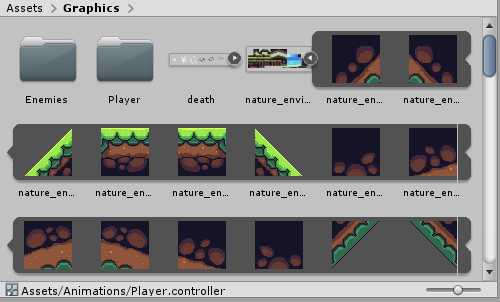
We all like to relax from time to time. Disconnect from our work and take a moment of rest. A good way to do this is through a good mobile game. We always carry the mobile with us, so it is very comfortable to open the application.

In our case, the application has been developed in unity, an environment that perfectly combines C Sharp programming (using scripts) and an intuitive development interface full of possibilities.

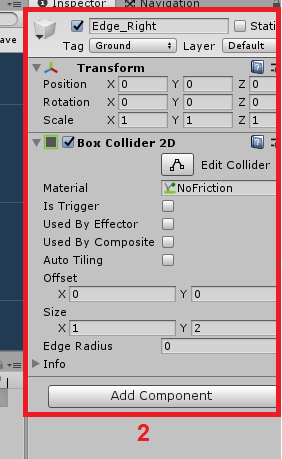
The following document will explain in detail some functionalities developed in the scripts as well as the use of the unity interface.

Game design

For the design of the components that make up the game, we will use the sprites. These are the different cells (tiles) that will form both the player and the enemies and the formations used in the level.



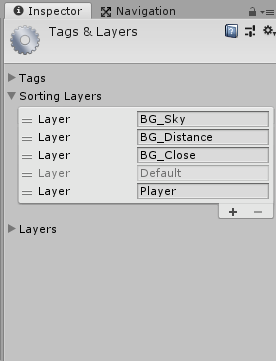
Our goal is to create objects (prefabs) and save the position of the sprites that make it up (1), as well as the properties that we want to give them (2) (colliders, scripts, rigidbody ...). In this way, we can use them as many times as we want.





Different layers have been created with priority to create the level, so some sprites will be above others following a hierarchy:

* BG\_Layers for environment.
* Default layers for enemies and level blocks.
* Player layer reserved to the player.

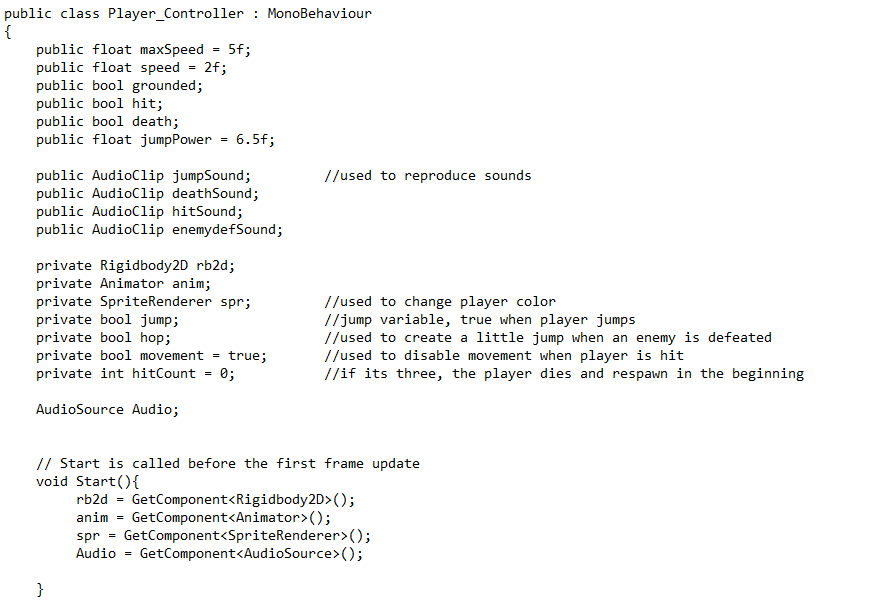


Player

It is the most important part of the game. In the script, Player\_Controller, it is contained the code used to determine the behavior with the different objects of the level. An additional script (CheckGround) has been used together with a lower 2D collider to detect when the player is touching the ground, useful to program the jump movement.

* Player\_Controller script

-Variables and Start method

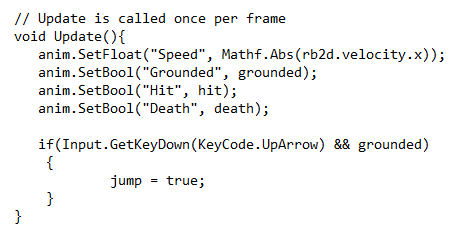


Public variables used like speed or jumpPower. That way, you can change the values from inspector interface in Unity.

Audio clips for every move of the player: jump, when the player its hit, death and when the player defeats an enemy.

Private variables used in the script are explained in the comments.

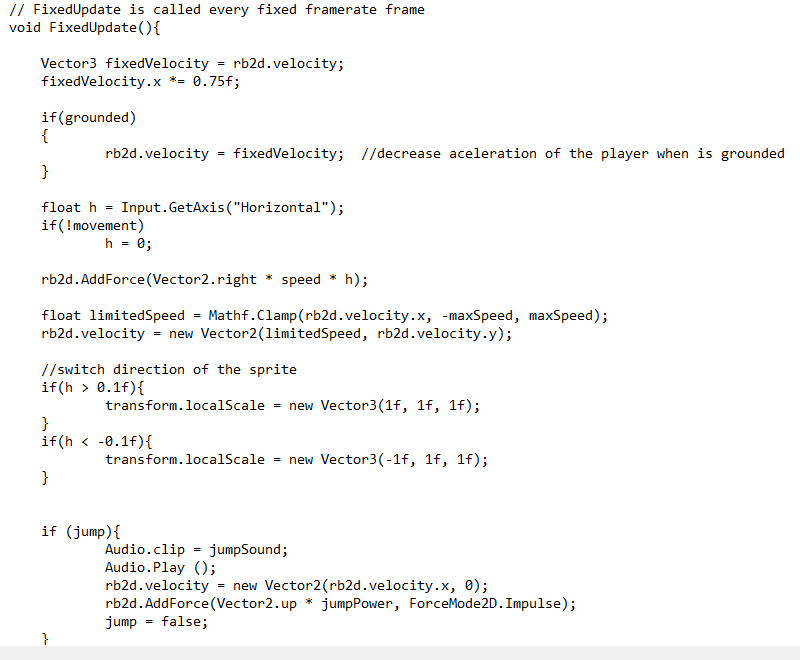
-Update method



Variables used in player animations.

The player jumps if up arrow is pressed and the player is in the ground.

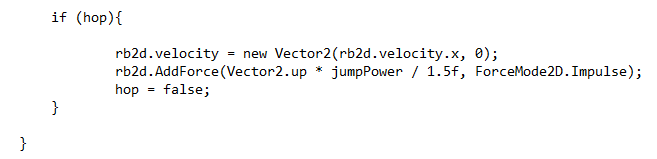
-Fixed Update method



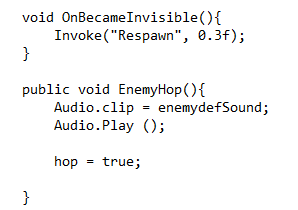
Fixed Velocity variable used to decrease player acceleration

With limited Speed variable we limit the player speed (in x coordinate).

When jump is true, jump sound sounds and an impulse is added in y coordinate, depending in jumpPower value.

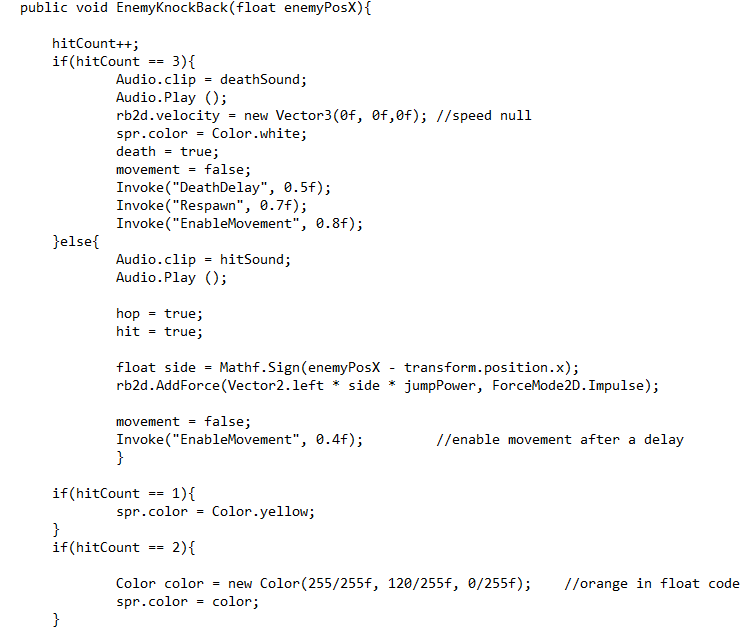


Hop is used when player defeats an enemy, like a smaller jump.



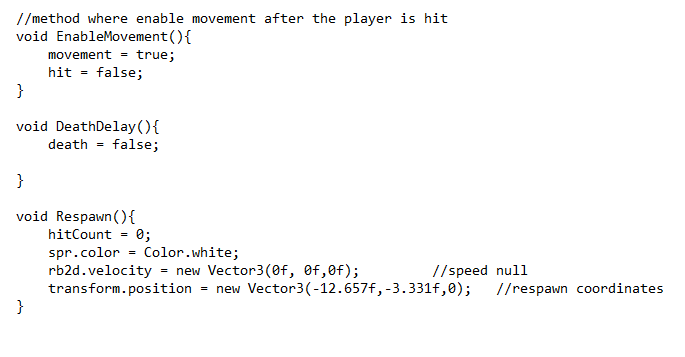
OnBecameInvisible is the method called when player falls in a pit. The player reappears in the beginning after a short delay.

-EnemyKnockBack method



It is the method used to determine player's behaviour when they hurt him. Each time he is hit, the hit counter adds one and player suffer a knockback, which consists of a diagonal impulse, preventing the player from moving at that time. Depending on the number of hits received, the player will change color, first yellow then orange and on the third hit he dies and reappears at the beginning.

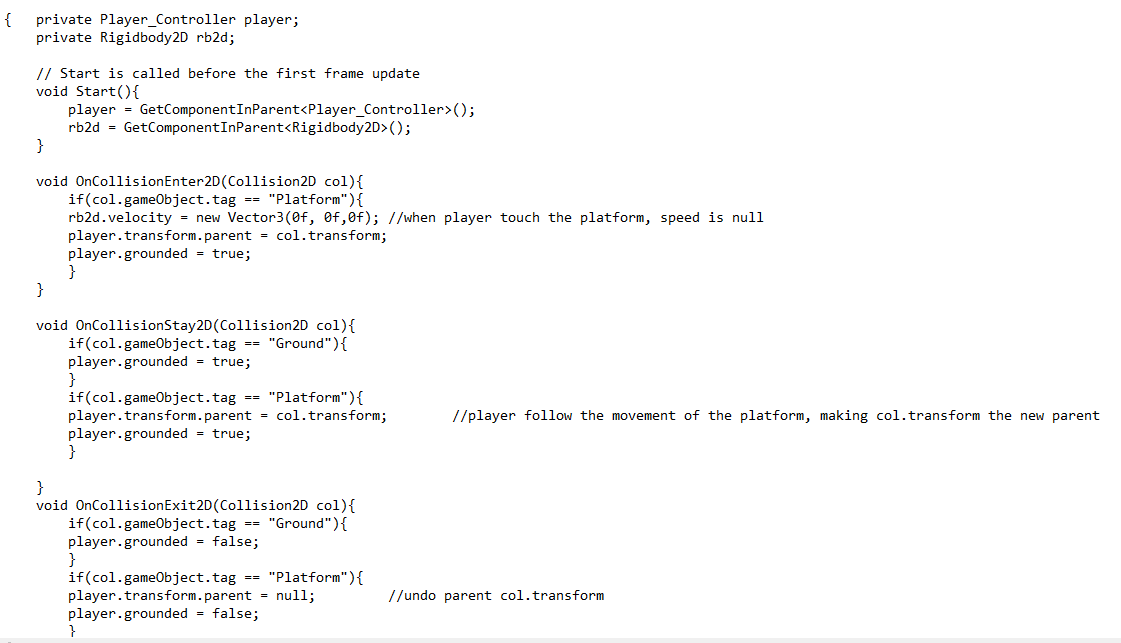
-Other methods



Respawn method used to reappear at the beginning when he falls into the pit or when he dies from being hit three times.

DeathDelay method used to give time to death animation.

* CheckGround script



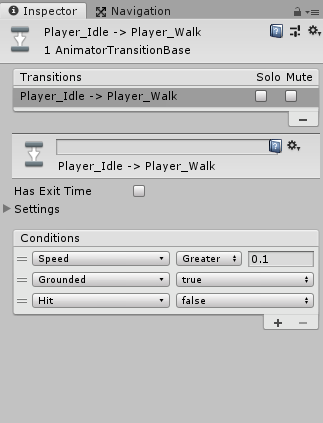
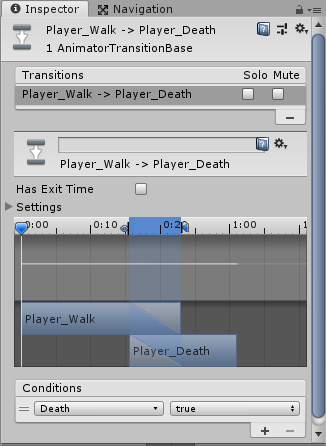
In this script the behavior of the character with the different surfaces is developed.

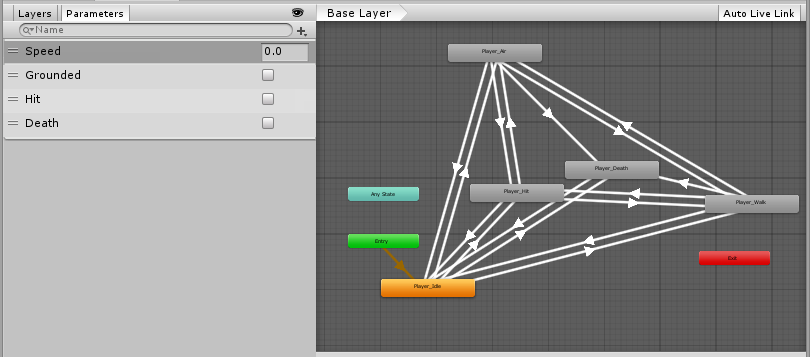
For this you need to access the variables of the Player\_Controller script (GetComponentInParent <Player\_Controller>).

When the player hits the ground, the grounded variable is true. When the player comes into contact with a mobile platform, he becomes a child of the platform. In this way, the player goes at the same speed as the platform.

* Animator

Here the different animations of the character are related to each other through transitions. Transitions are controlled by different variables, like hitting or speed.

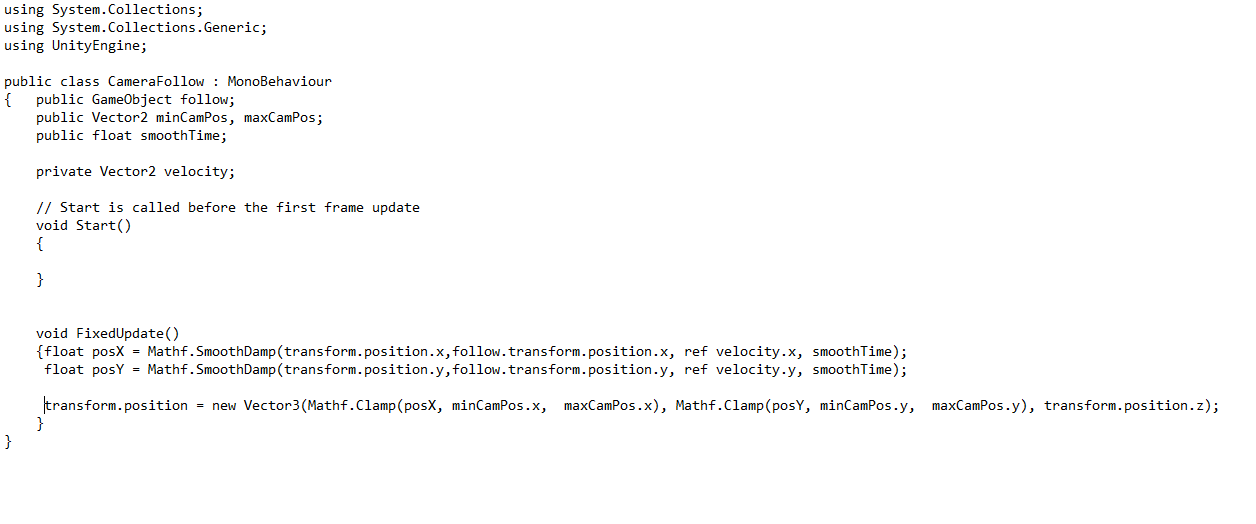




Camera

The game is mostly developed to advance from left to right and horizontally. However, a maximum camera position on the Y axis is contemplated due to some jumps on platforms.

Smooth Damp is a function used to create a smooth effect in the camera. It can be controlled with smooth Time public variable.

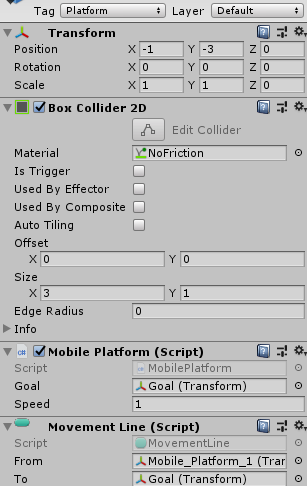


Mobile Platform

In order for the player to go at the same speed as the platform, the player becomes a child of the platform object as soon as he touches it, and ceases to be so when he leaves the platform.

The movement line script has been implemented to make the trajectory of the platform easier to see.

The platform will go to the goal and when it arrives, the goal will be in the initial position of the platform.



Falling Platform

The falling platform is designed to fall after a short time after being touched. The platform will reappear after three seconds. It can also be traversed from below (use one way function in 2D effector).

