[Nível 1 1](#_Toc1666578364)

[Nível 2 1](#_Toc1438726857)

[Nível 3 2](#_Toc1607908545)

[Nível 4 3](#_Toc1520103362)

[Nível 5 4](#_Toc2014676879)

# Nível 1

We created a PlayerStats script in the assets > Scripts > Player folder. We removed monoBehaviour and replaced it with ScriptableObject, also adding the attribute that allows us to create the object from the Unity menu (Fig.1).

Inside the Assets folder, we created a new folder called SO and a ScriptableObject named Player Stats, where we defined the configuration variables, level, current health, and maximum health of the player.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Os conteúdos gerados por IA podem estar incorretos.

Fig1

Next, we created the Extra folder and added the IDamageable script (Fig.2), which we turned into an interface containing the method responsible for applying damage. We also created the PlayerHealth script, which implements this interface, allowing us to use the TakeDamage() method.

Uma imagem com captura de ecrã, software, texto, Software de multimédia

Os conteúdos gerados por IA podem estar incorretos. Fig.2

# Nível 2

In the PlayerHealth script, we added a variable of type PlayerStats and made it visible in the inspector. We implemented the TakeDamage() method, which subtracts the player’s health according to the damage received.

We also created a check that calls the PlayerDead() function when health reaches zero or below, displaying the message "Player Defeated" in the console (Fig.3).

Finally, we added a small test in the Update method that allows simulating damage by pressing the P key, confirming that the system works correctly (Fig.4).

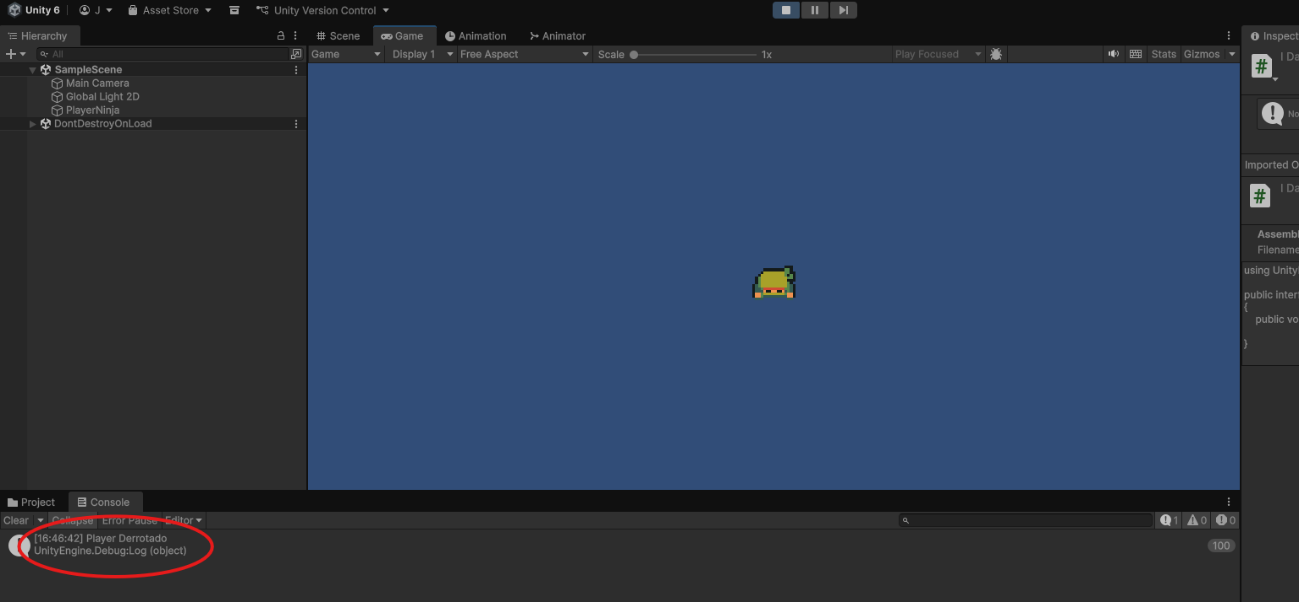


Fig.3

Uma imagem com texto, captura de ecrã, Software de multimédia, software

Os conteúdos gerados por IA podem estar incorretos.

Fig.4

# Nível 3

We created an animation called Player\_Dead, composed of a single keyframe with the sprite corresponding to the defeated player (Fig.5).

In the Animator, we configured the Dead trigger and a transition from Any State to the death animation, with no exit time or duration.

Then, we created the PlayerAnimations script, responsible for managing the movement and death animations (Fig.6).

Uma imagem com Software de multimédia, software, Software gráfico, Edição

Os conteúdos gerados por IA podem estar incorretos.

Fig.5

Uma imagem com captura de ecrã, texto, Software de multimédia, software

Os conteúdos gerados por IA podem estar incorretos.

Fig.6

# Nível 4

In the PlayerMovement script, we added a PlayerAnimations variable, obtained through GetComponent(). We replaced the old direct references to the Animator with the new functions created in PlayerAnimations (Fig.7).

In PlayerHealth, we also added a reference to PlayerAnimations, allowing us to trigger the death animation as soon as the player’s health reaches zero.

Uma imagem com captura de ecrã, texto, software, Software de multimédia

Os conteúdos gerados por IA podem estar incorretos.

Fig.7

# Nível 5

We created the Player class, which contains a private PlayerStats variable, visible in the inspector, and a public property that allows access to this data.

In the PlayerMovement script, we added a variable of type Player and checked, within the movement function, whether the health is less than or equal to zero. If so, movement is blocked.

We assigned the Player Stats ScriptableObject to the Player component and tested the final result: the character stops moving when out of health, and the death animation plays correctly (Fig.8).

Uma imagem com captura de ecrã, Software de multimédia, software, Software gráfico

Os conteúdos gerados por IA podem estar incorretos.

Fig.8