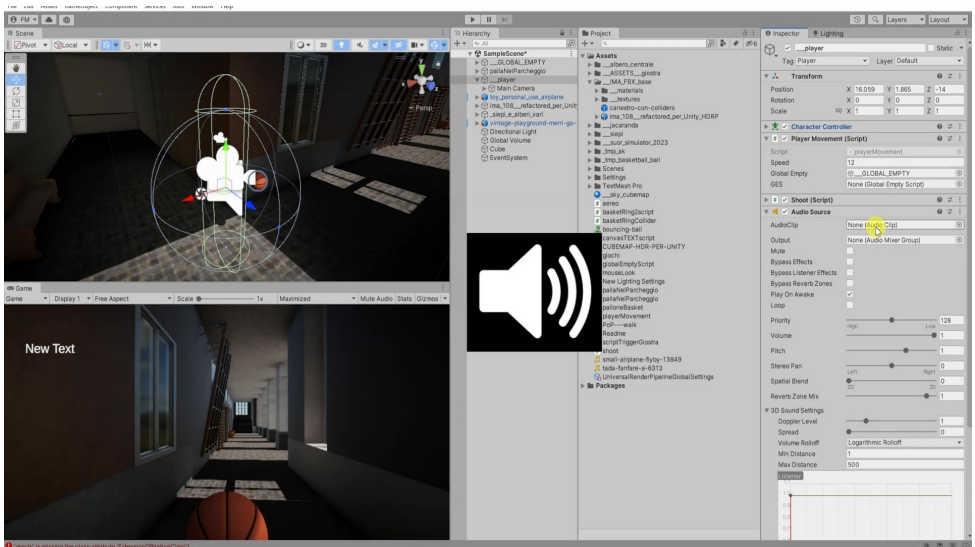


This PDF is available for free; feel free to share it!

[Subscribe to my YT channel](#) & turn on notifications to get updates on new videos!

NOTE - In the tutorial, you'll notice comments on other parts of the code in Italian. This is because I am Italian, and the scenario I'm showing you in this example is from a purely personal project. Please ignore those comments, as they pertain to parts of the script not relevant to this tutorial.



Hello everyone!

In this tutorial, I will show you how to play an audio file in **Unity** and, more importantly, how I resolved an issue related to **overlapping audio playback** from an **Audio Source**. Before we proceed, please note that the solution presented in the video is just the one I adopted for this problem, but there are likely other (and potentially better) solutions out there; so, don't take the information provided here as the definitive approach!

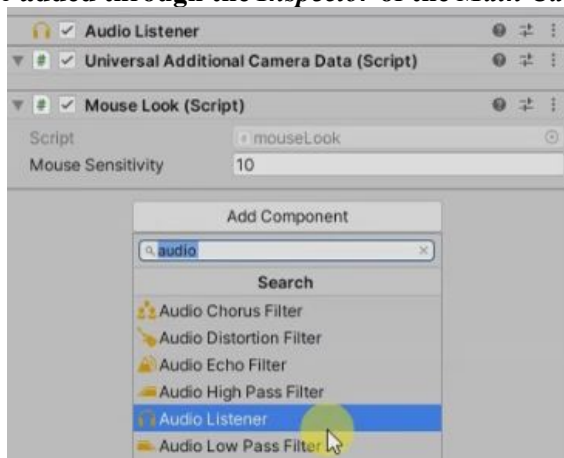
The tutorial was created using **Unity 2022**.

Specifically, I want to play a footsteps audio file (sourced from the first *Prince of Persia* game, by **Jordan Mechner**, which was my first PC gaming experience) when the character moves within the scene. This may seem like a straightforward task, but as we will discover, it conceals a minor challenge.

To play the footsteps audio, the first step is to **attach an *Audio Source* object to the character**. Within this object, we will define the specific **audio file** to be played and configure other playback properties.

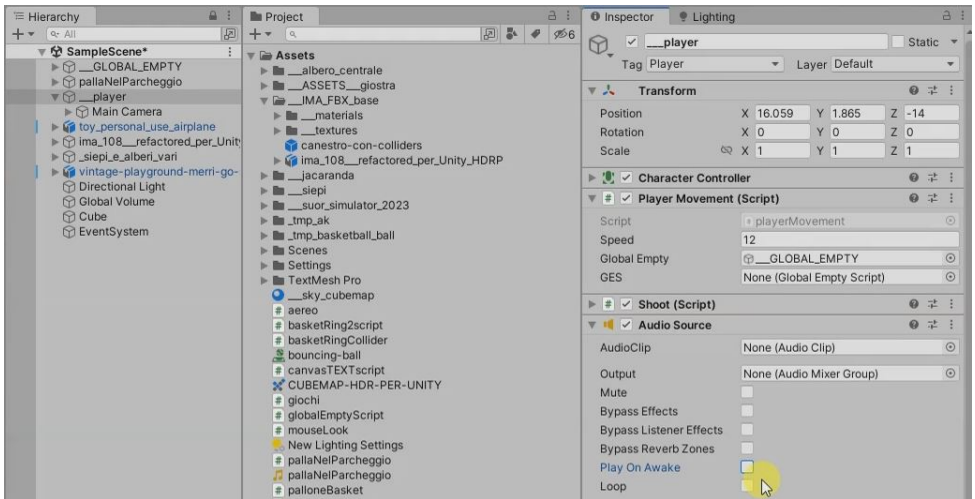
An ***Audio Source*** is a component that needs to be added to an object within the scene. This could be the character, a door, a ball, or even an invisible object—useful for playing an audio file when the character enters a specific room or gets near a particular item.

For the audio to be heard by the Main Camera, **the Camera needs to have an active *Audio Listener* component**. In new Unity projects, the Main Camera typically comes with **an active *Audio Listener* component by default**, as it is assumed that users will want to hear audio files. If the default *Audio Listener* is missing, it can be **added through the *Inspector* of the *Main Camera* object**.



In my scenario, the audio file needs to be linked to the character.

Within my project, there is a ***Player*** object that serves as the parent of the *Main Camera*. The **movement script** is associated with the ***Player*** object, while the camera is equipped with a script to enable looking around. Consequently, I will **attach an *AudioSource* component to the *Player* object**.

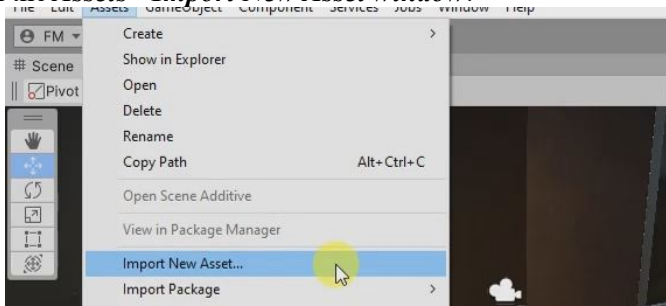


Upon accessing the tab of the newly created component, you will immediately see the **AudioClip** field, which is initially set to 'None.'

Additionally, there is a checkbox labeled "**Play on Awake**" that is selected by default. This option plays the audio file when the game starts, which can be beneficial in certain scenarios. However, it isn't necessary for our current use case, so **I will deselect it.**



Importing an audio clip into the project is quite straightforward: simply drag the audio file from a **File Browser** window into the **Project** tab or choose it from your disk through the **Assets - Import New Asset** window.



Unity is compatible with various audio formats, such as mp3, and it will display a musical note icon alongside the corresponding element within the **Project**.

As you can see, my project already contains several audio files intended for playback at different moments. Among these files, the one we're specifically interested in is "**PoP---walk**".

To proceed, I first select the **Player** in the **Hierarchy**.

Next, I click and hold the button down on the **PoP---walk** asset in the **Project**.

Finally, I drag this element into the **AudioClip** box of the **AudioSource** component associated with the **Player**.

As previously mentioned, the sound should be played when the character moves.

To achieve this, I will open the **script** responsible for handling the character's **movement** and insert the required code.

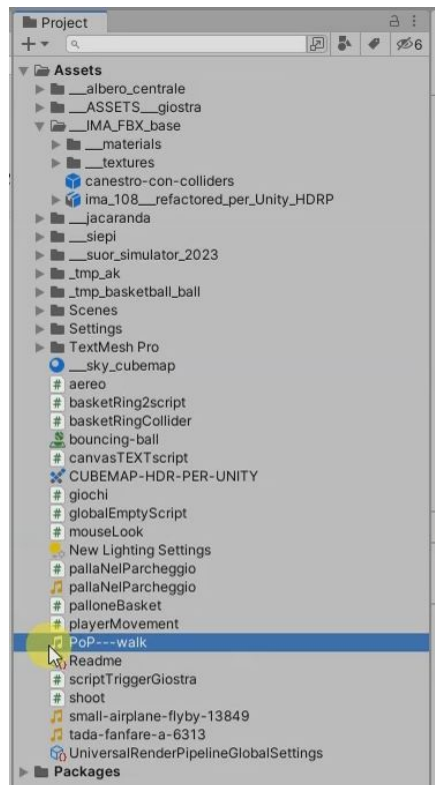
First, I create an **AudioSource** variable and initialize it within the **Start** function by associating it with the **Player's AudioSource component**.

To do this, I write the following code:

```
private AudioSource  
playerAudioSource;
```

and, inside the **Start** function:

```
playerAudioSource =  
GetComponent<AudioSource>();
```



I also need to create an **AudioClip** variable to associate with the audio file that needs to be played. I'll call this variable **myAudioClip** and use the **SerializeField** attribute because I want it to be visible in the Inspector.

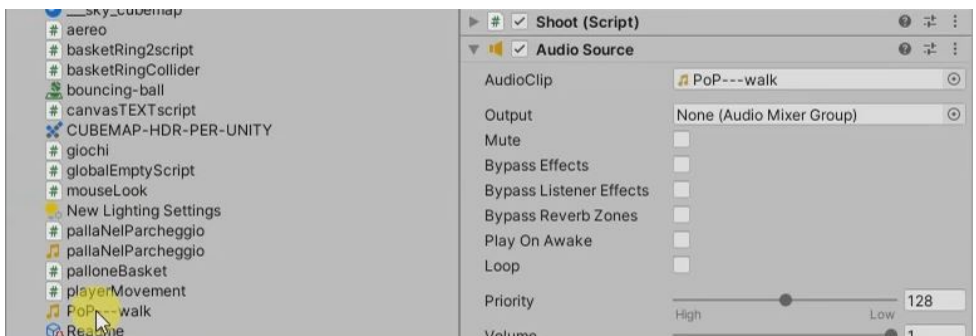
After saving the script file, the next step will be to **drag the "PoP---Walk" asset into the *myAudioClip* field of the script** within the Inspector.

Immediately following the definition of *playerAudioSource*, I write the following line of code:

```
[SerializeField] private AudioClip myAudioClip;
```

```
9      private float gravityValue = -9.8f;
10     private float jumpHeight = 30000f;
11     Vector3 gravityVelocity;
12
13
14     private AudioSource playerAudioSource;
15     [SerializeField] private AudioClip myAudioClip; // Inspector
16
17
18
19     public GameObject globalEmpty; // Inizializzata nell'Inspector
20     public globalEmptyScript gES;
21
22     // Start is called before the first frame update
23     Unity Message | 0 references
24     void Start()
25     {
26         playerAudioSource = GetComponent<AudioSource>();
27
28         controller = GetComponent<CharacterController>();
29         Cursor.lockState = CursorLockMode.Locked;
30         gravityVelocity = new Vector3(0f, 0f, 0f);
31         gES = globalEmpty.GetComponent<globalEmptyScript>();
```

After saving the script, I return to the Unity Editor and proceed to **drag the "PoP---Walk" asset into the *myAudioClip* field of the script**, as previously described.



At the moment, the audio is not being played.

Within the **Update** function of the script, there are some instructions to detect the user's keypresses, which control the character's movement.

Specifically, when the user moves the character, the value of **Input.GetAxis** for **Horizontal** and **Vertical** (which I have renamed *x* and *y* for convenience) can be either -1 or 1, depending on the direction.

In this case, I could write the following code:

```
if ((Mathf.Abs(x) == 1) || (Mathf.Abs(z) == 1))  
    player.AudioSource.PlayOneShot(myAudioClip);
```

A screenshot of a C# script in Unity, showing the Update function. The code is as follows:

```
32 // Update is called once per frame  
33 void Update()  
34 {  
35     if (giochi.giocoInCorso == giochi.Gioco.BASKET)  
36     {  
37         float x = Input.GetAxis("Horizontal");  
38         float z = Input.GetAxis("Vertical");  
39         Vector3 move = (transform.right * -z) + (transform.forward * x);  
40  
41         // L'istruzione seguente riproduce l'audio dei passi se c'è movimento (X e Z sono gli input,  
42         if ((Mathf.Abs(x) == 1) || (Mathf.Abs(z) == 1)) playerAudioSource.PlayOneShot(myAudioClip);  
43     }  
}
```

and try to run the game; however, doing so would be a significant mistake because Unity would play the audio file during EVERY FRAME OF THE GAME!

The result doesn't improve if we merely move the **PlayOneShot** function inside a **Coroutine** and add a one-second delay command (*yield return new WaitForSeconds(1)*). The issue remains that **Update** will call the **Coroutine** at every frame, leading us back to the same error.

The **Coroutine**, therefore, is not necessary; instead, we need to implement a simple **check on the AudioSource**: play the audio only if the **AudioSource** is **NOT** already playing it.

This can be done within the **Update** function by writing the following code inside the **if** block that checks the values of *x* and *y*:

```
if (!(playerAudioSource.isPlaying))  
    playerAudioSource.PlayOneShot(myAudioClip);
```

```
void Update()
{
    if (giochi.giocoInCorso == giochi.Gioco.BASKET)
    {
        float x = Input.GetAxis("Horizontal");
        float z = Input.GetAxis("Vertical");
        Vector3 move = (transform.right * -z) + (transform.forward * x);

        // L'istruzione seguente riproduce l'audio dei passi se c'è movimento (X e Z sono gli
        if ((Mathf.Abs(x) == 1) || (Mathf.Abs(z) == 1))
        {
            if (!playerAudioSource.isPlaying) playerAudioSource.PlayOneShot(myAudioClip);
        }
    }
}
```

Now we can save the script, start running the game, and observe (or, better yet: listen to) the result.

Before concluding this tutorial, I'd like to share a thought: the *AudioClip* field of the *AudioSource* component can, of course, be modified via script.

This means you can use this AudioSource to play a standard footsteps sound file, but also replace the audio clip during the game with another one if, for example, the character is walking on gravel, on the shoreline, or other surfaces, which can be identified using **Colliders**.

Additionally, if your character can fly, you can disable the *AudioSource* so that no sound is played.

Well, that's all for this tutorial; I hope it was helpful!
See you soon!
