FOOTSTEPS SYSTEM GUIDE

*"Thank you for purchasing **Footsteps System*** for any more info or support email me directly*"*

Author: **Murad Elboudy**

INTRO:

Footsteps System is an extremely simple and flexible system. It supports both simple first person controllers made from cylinders, spheres or what have you, as well as animated humanoid models. You have the flexibility of adding and removing surfaces with their respective footstep audios of both legs. It also comes with high quality audios for 7 surfaces (asphalt, wet, mud, sand, wood, metal and grass). In this documentation I'll be going over how to set everything up.

GETTING STARTED FOR CONTROLLER:

1. Add *FootstepsSystem* script to your controller.
2. In the **Audio Lists** property inside the inspector set the size to **2**. This means you will have 2 different surfaces to detect.
3. Set the **Ground Tag Name** for each one. Name the first *Asphalt* and the second *Sand.*
4. Expand the **Audios** property, the size will be set to 4. And no it can't be changed. Drag and drop the audio sources you have for each footstep to each surface type audio source. You can of course use the ones that come with the package. They're at your disposal.
5. Now you have two surface types set with their respective *tag names* and *audios*. Now the only thing missing is the actual surfaces.
6. Create two simple 3D planes and set one **tag** to Asphalt and the other Sand (the two surface types we created in the inspector of the system).
7. Great! Now we have the actual surfaces inside our game and each one tagged with it's surface type so the system knows which audio to play.
8. Give both of your new surfaces a new distinctive layer like **Ground**
9. In the **Ground Layer** property inside *FootstepsSystem* component set it to the same layer as the planes.
10. Now inside your character movement script *GetComponent<FootstepsSystem>()* and inside the update method of your character movement simply call **PlayFootSteps()**. Make sure you make proper logic and validation and trigger the method only when your character is actually moving if the velocity is more than 0 or any of that sort. *The time gap between each audio will be taken care of by FootstepsSystem itself.*
11. When the character stops call **ResetFootSteps()**. This will reset everything and make the loop more seamless.
12. Now as your character moves between different plane tags the appropriate audios will play seamlessly.
13. You can change the interval gap where the footsteps audios play by changing the **Footsteps Interval** value in the inspector. To make it faster or slower syncing it with your movement speed.

GETTING STARTED FOR HUMANOID:

1. Add *FootstepsSystem* script to your character. **You have to add it where the animator component is.**
2. **Follow controller steps 2 -> 7**
3. This is where things get different than the controller. You can surely follow the same procedure as that of the controller and everything will work splendidly but since animations are a big part of humanoid movement then accuracy and removing error is a must so the best way is using *animation events*. This package comes with an animation event ready to be triggered only and it does the rest.
4. Go to the actual animation prefab of your character's movement (run, walk) scroll down and add an animation event. You will need to add two. One on each leg just before hitting the ground.
5. On the frame of impact of each leg in the animation add the event **FootSound().**
6. Now as your character walks or runs the animation events will play the specific audio of the surface currently moving on.

P.S: you can add PlayFootSteps() of the normal controller movement as the animation event instead of FootSound(). But the former function takes into account the footstepsInterval property, so it won't play the other footstep unless the interval time gap has passed. One cool trick is to simply set the footstepsInterval to 0, now you can use PlayFootSteps() as an animation event. But again, as stated using FootSound as the animation event is much more reliable and easier to say the least.

PROPERTIES AND METHODS:

- **audioLists ->** A class array that contains two properties: (**string**) *GroundTagName* and (**AudioSource[4])** *Audios.* This is responsible for containing all the surface types with their respective sounds for each foot. *If you have only two audio sources for each surface type you can add them again in the 3rd and 4th audio source.*

- **defaultTag ->** When a character walks above a surface with a tag that isn't added to the system it'll fall back to this tag and play the audio from this tag surface.

- **currentGround ->** Returns an **all lower-case string** of the current surface type tag

- **footstepsInterval ->** The time gap in seconds between each footstep

- **groundLayer ->** Layer Mask that the detection raycast will hit

- **downwardRayLength ->** The length of the raycast going downwards that detects the surface types

- **PlayFootSteps() ->** Plays the footsteps with a gap of certain set interval between each footstep. This should be called inside the update method of the movment script.

- **ResetFootSteps() ->** Should be called only when the PlayFootSteps method has been triggered

and this method should be called when the character movement has stopped.

- **FootSound() ->** An animation event that should be called exactly as an animation event inside the animation prefab on each foot before it hits the surface. The respective audio of the current surface will play accordingly when the animation is played and this method is triggered.