

# Starter Assets

Asset Store Documentation v1.1

Starter Assets include First and Third Person Character controllers built in a modular way as a solid foundation for any game genre.

- [Starter Assets - First Person Character Controller](#)
- [Starter Assets - Third Person Character Controller](#)

The Starter Assets packages are compatible with **Unity 2020 LTS and 2021 LTS**.

The Starter Assets require the Input System and Cinemachine packages to work.

The ThirdPerson PlayerArmature uses a **Humanoid character rig**.

The Character uses the built-in **CharacterController** component.

The First/Third Person Starter Assets packages require the **Input System** and **Cinemachine** packages to work.



# Table of contents:

[Table of contents:](#)

[Important note on package dependencies](#)

[Switching input systems](#)

[Playground Scene](#)

[Set up the Starter Assets in a new Scene](#)

[Mobile setup](#)

[Change Input style](#)

[Set up mobile UI in a new Scene](#)

[Building to a mobile device](#)

[URP/HDRP compatibility](#)

[Lighting setup \(HDRP\)](#)

[Layers](#)

[Setting up Ground and Character layers](#)

[Add a custom character \(Third Person only\)](#)

[Footsteps SFX \(Player Armature only\)](#)

[Add footstep SFX to your custom animation](#)

[Landing SFX \(Player Armature only\)](#)

[Add landing SFX to your custom animation](#)

[Adding a new Input Action](#)

[Technical documentation](#)

# Important note on package dependencies

The First/Third Person Starter Assets packages require the **Input System** and **Cinemachine** packages to work.

The packages will automatically install when you import a Starter Assets package into your project. This is handled by the PackageChecker.cs script.

Please accept the Input System pop-up and Editor restart to successfully install the New Input System. If you accidentally decline, you can install the New Input System through the Package Manager.

## Switching input systems

If you want to use the Input System and Input Manager (Old) together, or switch your project back to the old Input Manager, you will need to go to **Edit > Project Settings**, then select **Player**.

Under **Other settings**, you will see that **Input System Package (New)** is selected. Here you can switch to your preferred setup. Please bear in mind that the Starter Assets do not work with the Input Manager (Old) setting.

## Playground Scene

In **Assets/StarterAssets/ThirdPersonController/Scenes** or **Assets/StarterAssets/FirstPersonController/Scenes**, you will find the **Playground** scene. Here you can use the Starter Assets controller in a simple playground environment.

## Set up the Starter Assets in a new Scene

If you want to add the character controller to a new Scene, you need to complete a simple setup.

There are several ways to do this:

*Drag and drop (Nested Prefab)*

- Go to **Assets/StarterAssets/ThirdPersonController/Prefabs** or **Assets/StarterAssets/FirstPersonController/Prefabs**. Locate the Nested Prefab you want, for example **NestedParentArmature\_Unpack**. This is a Nested Prefab that contains everything you need to set up a PlayerArmature in a new Scene.
- Drag the NestedParent Prefab into your Hierarchy, right-click, and select Unpack.
- Drag the contents of the nested parent out into the Scene, and you are ready to go.

### *Automated setup*

- Go to **Tools > Starter Assets** and select the type of controller you want to set up, for example **Reset Third Person Controller Armature**. This will set up a new PlayerArmature in your Scene with everything hooked up.
- You can also use this tool to reset elements of your current Player back to default.

### *Manual setup*

- Drag the **PlayerCapsule** or **PlayerArmature** into the new Scene.
- Drag the **PlayerFollowCamera** into the Scene.
- Under **Follow** in the CinemachineVirtualCamera in the Inspector, select the **PlayerCameraRoot** object.
- Replace your Main Camera with the MainCamera prefab in the Prefabs folder, or assign a **CinemachineBrain** to the existing Main Camera.
- Press Play and you're good to go!

## Mobile setup

The Starter Assets packages come with a UI overlay and Input setup for mobile devices.

In the Playground Scene, the Hierarchy already contains

**UI\_Canvas\_StarterAssetsInputs\_Joysticks** with all the right connections. Enable it and you're good to go!

## Change Input style

If you prefer touch zone inputs over joysticks, go to

**Assets/StarterAssets/Mobile/Prefabs/CanvasInputs** and add

**UI\_Canvas\_StarterAssetsInputs\_TouchZones** to the Hierarchy instead. Delete the other UI Canvas.

In the **UI Canvas Controller** Component, make sure to assign the Player Prefab to **Starter Assets Inputs** under **Output** in the Inspector.

## Set up mobile UI in a new Scene

To set up mobile Inputs in a new Scene, first set up your preferred Player Prefab as instructed in the “Set up the Starter Assets in a new Scene” section above.

Now, go to **Assets/StarterAssets/Mobile/Prefabs/EventSystem** and find the **UI\_EventSystem** Prefab. Drag it into the Hierarchy.

Select your Player Prefab in the Hierarchy. Under the **Player Input** Component, go to **UI Input Module**. Here, select the **UI\_EventSystem** Prefab.

Next, you need to add the specific UI you want to use.

Go to **Assets/StarterAssets/Mobile/Prefabs/CanvasInputs**. Add **UI\_Canvas\_StarterAssetsInputs\_TouchZones** or **UI\_Canvas\_StarterAssetsInputs\_Joysticks** to the Hierarchy depending on your preferred control method.

Select the UI Canvas. In the Inspector in the **UI Canvas Controller** Component, make sure to assign the Player Prefab to **Starter Assets Inputs** under **Output**.

Now you're all set up to build on a mobile device with your preferred input method.

## Building to a mobile device

If you are unfamiliar with building on a mobile device, please take a look at the documentation below.

[Building for Android](#)

[Building for iOS](#)

Make sure to add Android/iOS Build Support to your Editor version in order to be able to change your build target.

For more information on how to add modules to the Editor and adjust your Build Settings to build on Android or iOS, please see the following documentation:

- [Adding modules to the Unity Editor](#)
- [Build Settings](#) (change Build target)

# URP/HDRP compatibility

By default, the Starter Assets materials are using the Built-in Render Pipeline. If you want to use the package in URP or HDRP, you will need to upgrade the materials.

Please follow the instructions in the official documentation below:

- [Upgrade shaders to URP](#)
- [Upgrade shaders to HDRP](#)

In `Assets/StarterAssets/Environment/Art/Materials/URP_HDRP_ShaderGraph`, you can find a customizable Triplanar Material for URP/HDRP projects.

## Lighting setup (HDRP)

If you upgrade the project to HDRP, you will need to enable the lighting in the Playground Scene.

In the Hierarchy, select the **Directional Light** under the **Lighting** parent and the lighting should show correctly. Make sure the Intensity is set to 100000lux if you are aiming for realistic lighting.

Finally, go to **Window > Rendering > Lighting**, and press the **Generate Lighting** button.

## Layers

When you import the Starter Assets into your project, you may want to do some layer setup to make sure that everything works as expected, since the layers are specific to each individual project and won't be transferred automatically with the Asset Store package.

The character controller determines if it's standing on solid ground via the layer system and is useful to control how objects interact with each other.

A good layer setup makes sure that the ray casting and physics always react as expected, because it knows each Prefab's place in the system.

## Setting up Ground and Character layers

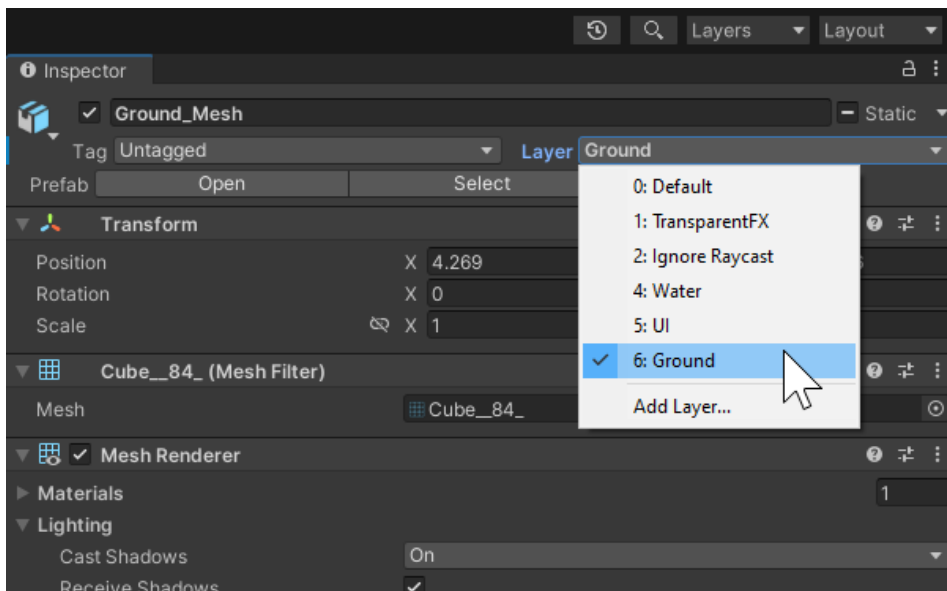
In the Inspector of the character's Prefab (PlayerArmature or PlayerCapsule), there's a section called **Player Grounded** with a dropdown to select **Ground Layers**. From here, you can specify which layers the Character Controller will consider as ground. In a new project, this will be set to **Default**, but it's a good idea to create a new layer called **Ground** in the layer menu and assign it to the objects that your player should walk on. This is most important if you want to continue

building your level or want to create more complex interactions.

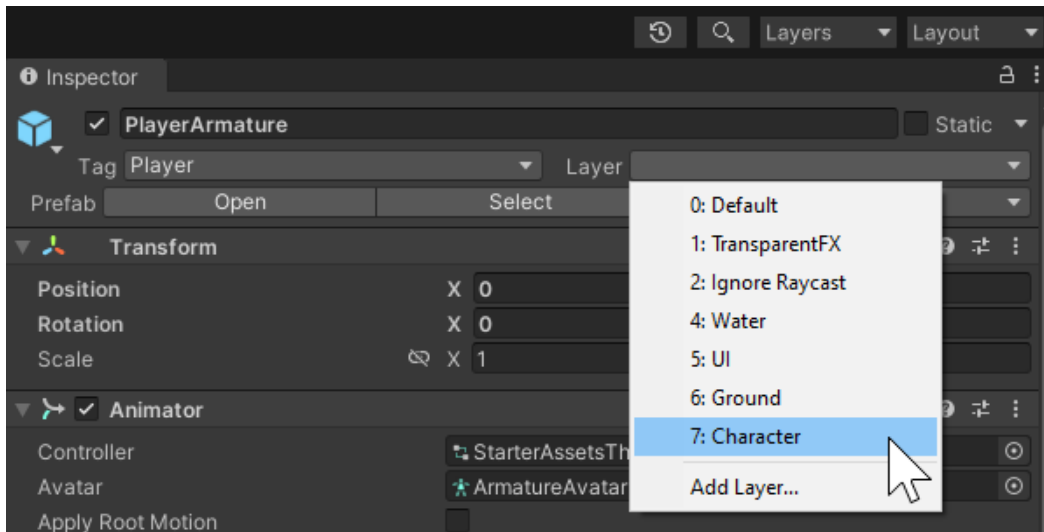
- Find the Layers menu in the top right. In a default project, it will look something like this.



- Click “Edit Layers” and create a Layer called **Ground**.
- Apply the new Ground Layer to objects that your player will walk on, for example the Ground mesh, by selecting the object and choosing the newly created Layer from the Layer dropdown.



- Additionally, we recommend creating a new layer called “Character” and applying that to your PlayerCapsule or PlayerArmature Prefab. Otherwise, there can be instances where the character is considered solid ground and never enters an in-air state. This will affect the character’s animation state and ability to jump.





## Add a custom character (Third Person only)

You can easily exchange the Player Armature for your own character model by following the steps in the [Youtube overview video](#) (timestamp 7:30).

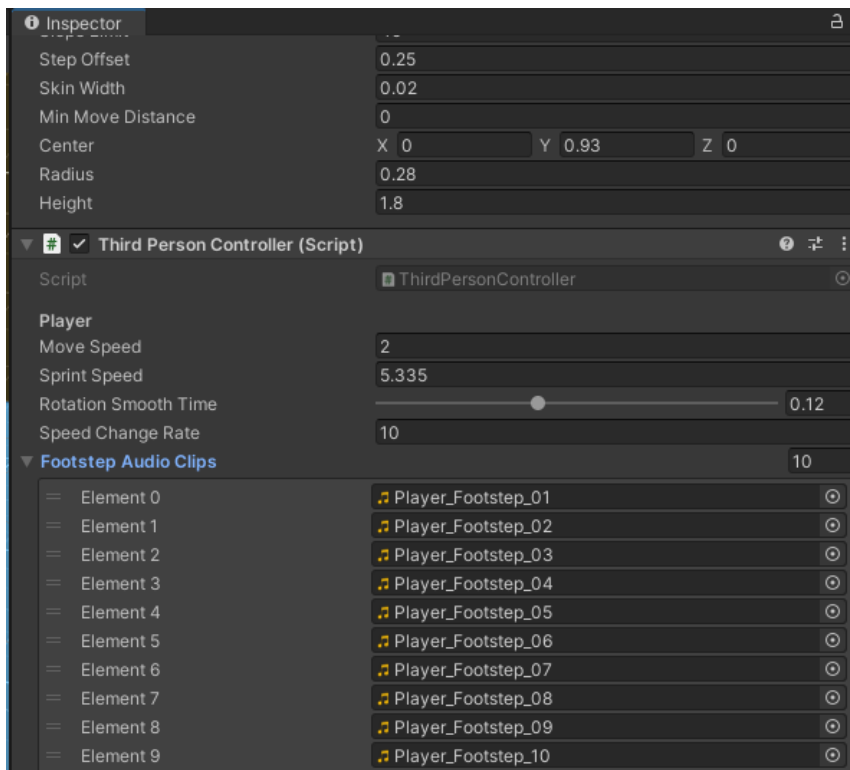
If you change the Controller in addition to the Avatar, you will be able to switch to a different animation set as well.

## Footsteps SFX (Third Person only)

The Third Person Character Controller comes with footstep sounds, which are controlled via Animation events.

Animation events are used to call methods at specific times within an animation. If an animation is playing and detects an animation event, the animation system will call that function on every MonoBehaviour attached to that game object.

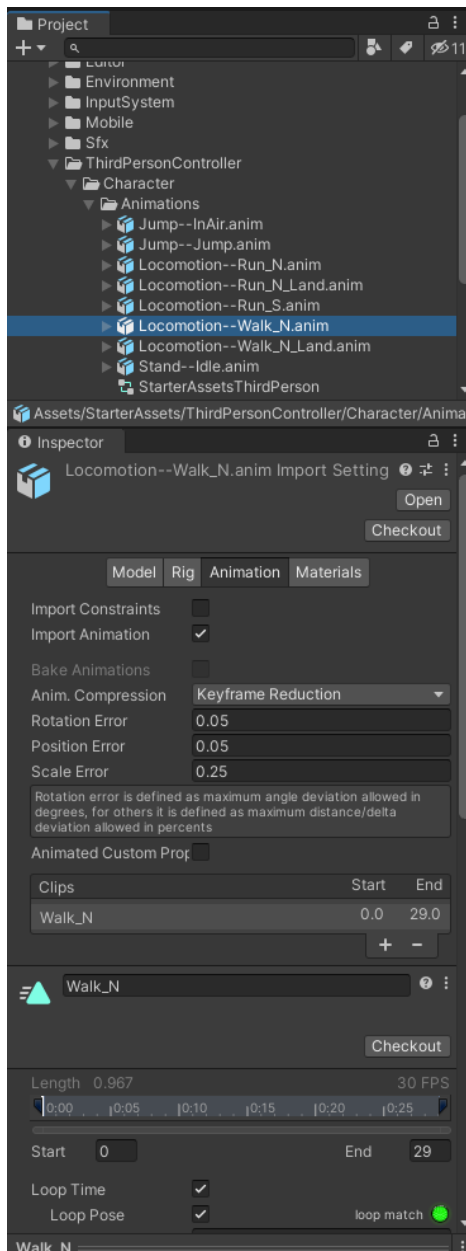
The **ThirdPersonController** component, which is attached to the **PlayerArmature**, has a method called **OnFootstep** that will play an AudioClip. It randomly selects a clip from the list of **FootstepAudioClips**, which is defined in the Third Person Controller Component's Inspector. You can modify the list of available clips through the Inspector.



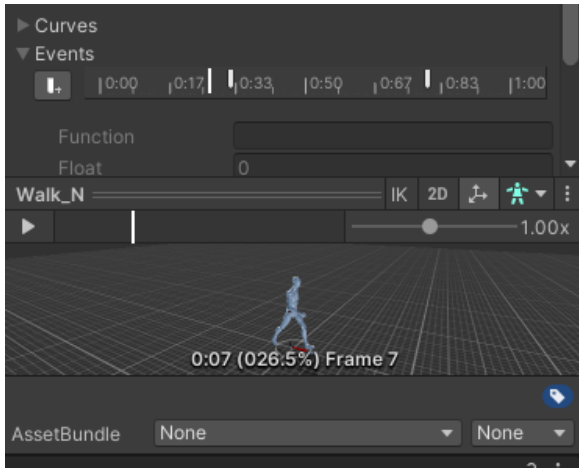
## Add footstep SFX to your custom animation

If you want to add footsteps to a new set of animations, you must define the **OnFootstep** animation events manually on your walk and run animations. This is done via the Import Settings of your animation files.

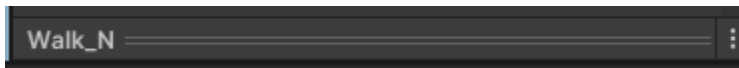
- With an animation file selected (Fbx, Blend, etc.), go to the animation tab in the Inspector.




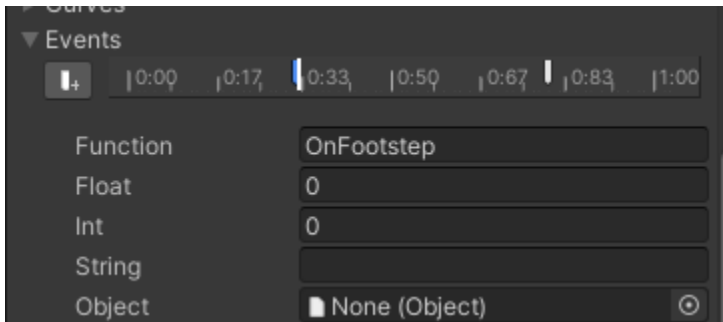
- Scroll down to the events foldout.



- With the animation preview window open, you may scrub through the timeline to find the frame when the foot makes contact with the ground. If you don't see the animation preview, click on this bar at the bottom of the Import Settings inspector



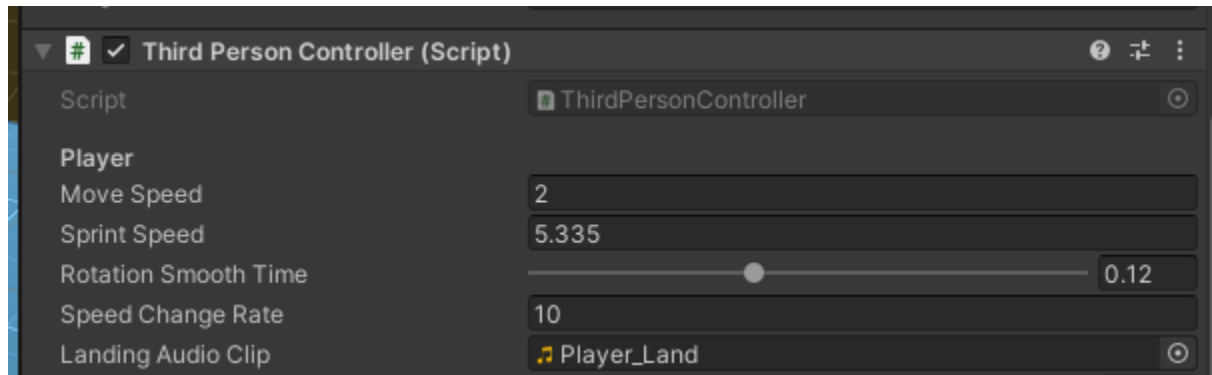
- Once you've found the correct frame, click the **Add Event** button  and set the function name to **OnFootstep**.



- Add **OnFootstep** events on each frame in the animations where you would like the sound to play.

## Landing SFX (Third Person only)

The **ThirdPersonController** component, which is attached to the **PlayerArmature**, has a method called **OnLand** that will play an AudioClip. The sound effect is defined by **Landing Audio Clip** in the Third Person Controller Component's Inspector.



### Add landing SFX to your custom animation

If you want to add footsteps to a new animation, you must define the **OnLand** animation events manually on your new landing animation. This is done via the Import Settings of your animation files.

You can follow the same steps as in [Add footstep SFX to your custom animation](#) to define the point in the animation where you'd like the landing SFX to play, just make sure you set the Function in the Event to **OnLand** instead of **OnFootstep**.

## Adding a new Input Action

The character controller reads the input from StarterAssetsInput.cs. To add a new Input Action to your project, follow the steps below.

### First, do this:

In ThirdPersonController.cs (or FirstPersonController.cs) you'll read the bool for your new action from StarterAssetsInput.cs. So open StarterAssetsInput.cs and add a public bool named Fire next to the other Character Input Values.

Then, add a new function to StarterAssetsInput.cs, for example:

```
1. public void FireInput(bool newFireState)
2. {
3.     fire = newFireState;
```

```
4. }
```

This function will set a fire boolean inside StarterAssetsInput.

### **Then, for PC / Console / Input System-based devices:**

If you want to add a new action, for example "Fire", you'd add an action called Fire in the InputActions window (open the window by opening StarterAssets.inputactions at the path: StarterAssets/InputSystem/StarterAssets.inputactions) and assign a key binding.

To make it work for other devices that use the Input System (PC, console), create a new function inside StarterAssetsInput.cs. The Input System will automatically look for a function called "On[ActionName]" so in your case "OnFire":

```
1. public void OnFire(InputValue value)
2. {
3.     FireInput(value.isPressed);
4. }
```

You can query the InputValue argument to check if the button is pressed (value.isPressed) to check if the button was pressed or released.

### **Finally, for mobile:**

Now, because mobile controls are virtual, we need to create another function inside UICanvasControllerInput.cs, if we want this function to also work on mobile devices:

```
1. public void VirtualFireInput(bool virtualFireState)
2. {
3.     starterAssetsInputs.FireInput(virtualFireState);
4. }
```

Create a new UI Virtual Button from the prefab inside StarterAssets/Mobile/Prefabs/VirtualInputs, or duplicate one of the existing UI Virtual Buttons in the scene. Set that Button's "Button State Output Event" to the newly created "VirtualFireInput" function.

Since the mobile controls are virtual, they don't go through the New Input System, so at this point you're all set for mobile.

## Technical documentation

For more details about how the different Scripts work, please refer to the comments within the Scripts themselves.