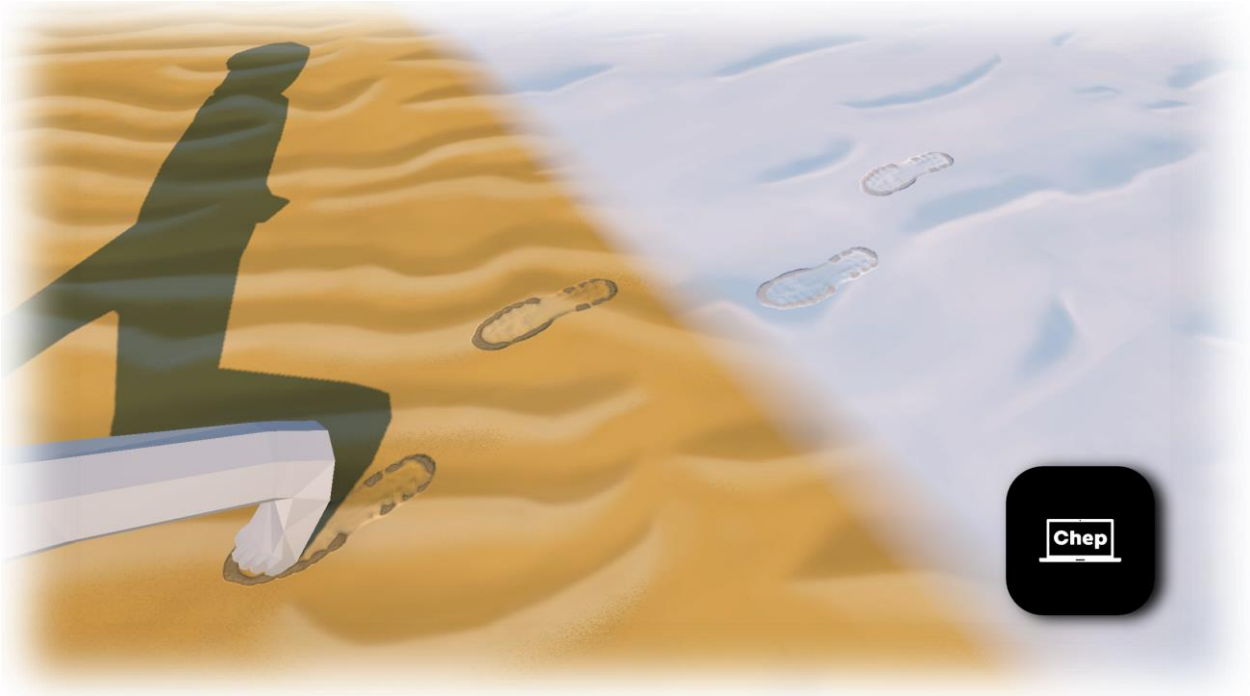
Modular Footstep System v1.0

Manual



Contents

Description

This plugin allows adding walking effects to entities. It does not include resource packs but provides a pre-built system (logic) for creating them. The plugin works with the **Animator** component to generate effects when the entity's foot touches the ground during animation.

There are three types of effects (modules) that can be added:

- Footstep traces
- Particle effects
- Sound playback

These modules can be combined with each other. For example:

- Add all three modules to an entity
- Add only footstep sounds
- Combine footstep traces and particle effects

The implemented system allows replacing generated effects depending on the surface the entity moves on.

There is also a module for replacing effects depending on the entity's state type. By "state type" we mean the action the entity is currently performing. For example, the entity is in a "Walking" or "Running" state. Depending on these states, different effects should play on the same surface. How to configure the system is described later.

A demo scene with a pre-configured system is available, demonstrating the system’s functionality. If you encounter difficulties during setup, it is recommended to study it.

Configuring Surfaces

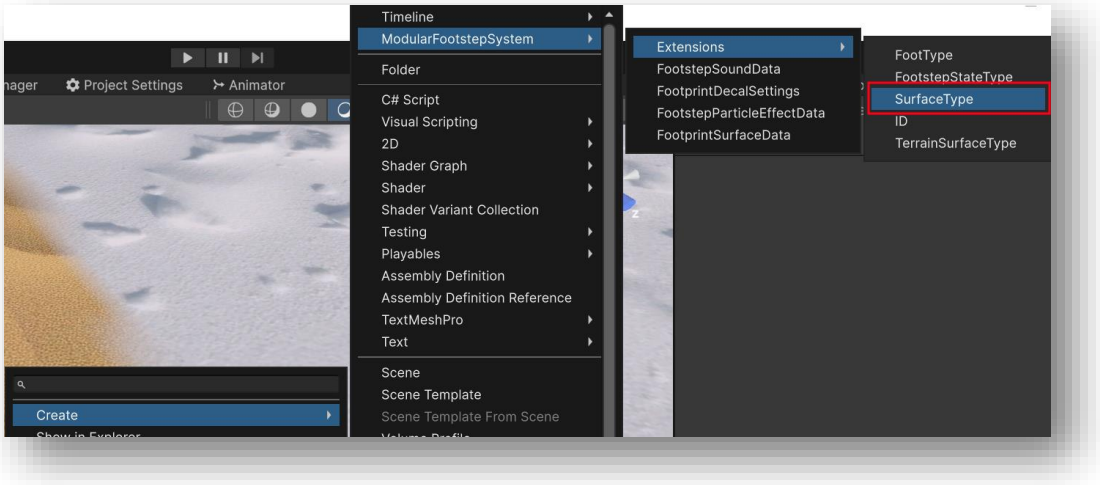
To determine the type of surface and play effects accordingly during movement, you need to create surface types and assign them to game objects representing surfaces. The plugin provides two types of surfaces:

Simple Surface (Homogeneous)

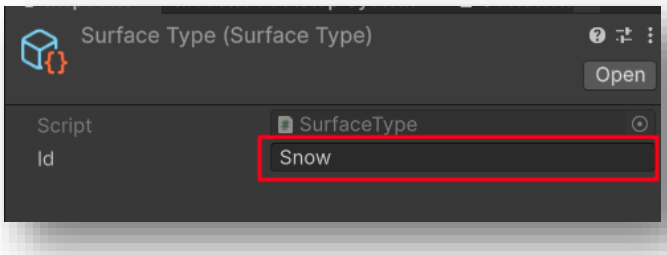
A surface with a uniform material. For example, wooden planks, stone tiles, snow, or sand.

To define such a surface:

1. Create the surface type via the context menu in the **Project** tab: **Create → ModularFootstepSystem → Extension → SurfaceType**.

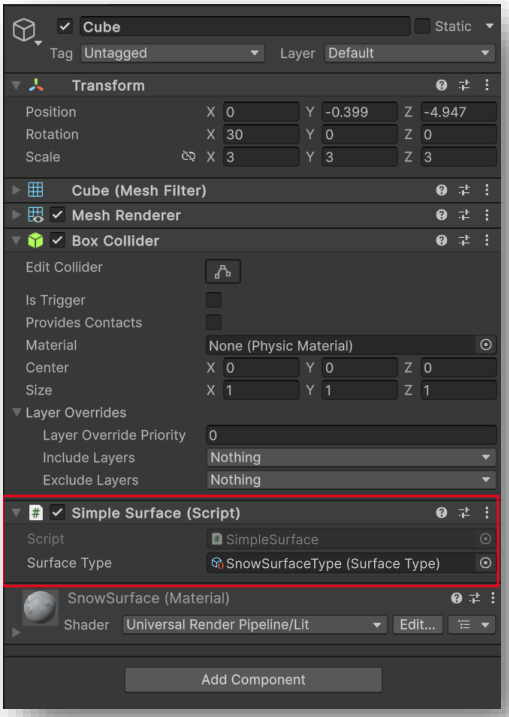


2. In the created **ScriptableObject**, fill in the **ID** text field with a unique value.



To allow the footstep system to detect this surface type later:

1. Assign it to a game object. Select the required object and add the **SimpleSurface** component.
2. In the **SurfaceType** field, specify the appropriate surface type for the object.

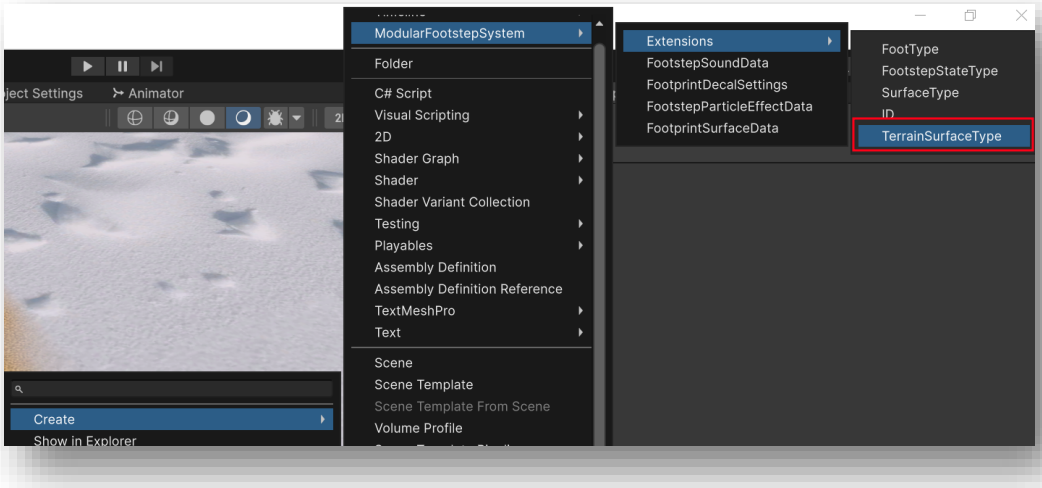


Non-Homogeneous Terrain Surface

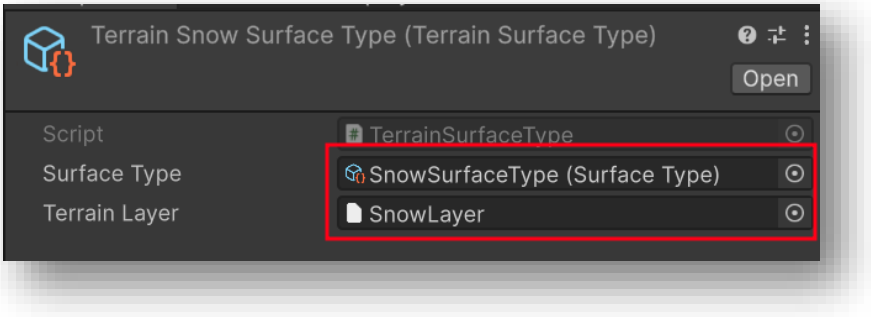
This represents a layer of a **Terrain** object.

To define such a surface:

1. Create the terrain surface type via the context menu in the **Project** tab: **Create → ModularFootstepSystem → Extension → TerrainSurfaceType**.

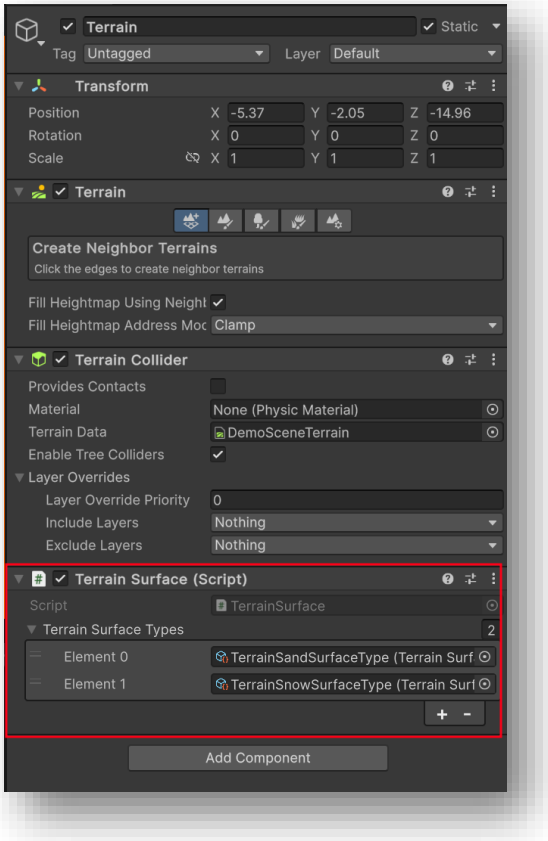


2. Fill in the **SurfaceType** and **TerrainLayer** fields in the created **ScriptableObject**.



To allow the footstep system to detect this surface type from **Terrain** later:

1. Add the **TerrainSurface** component to the **Terrain** object.
2. In the **TerrainSurfaceTypes** list, specify the required surface types used by the **Terrain**.



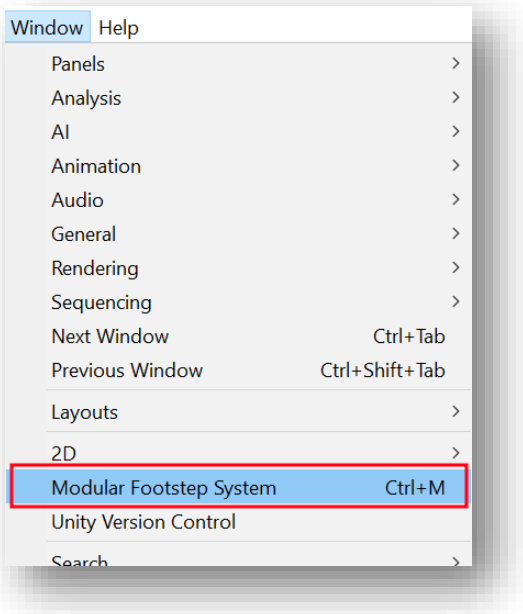
In further system configuration, for the created surface types, you will define whether footprints, particle effects, or sounds need to be generated for each specific entity.

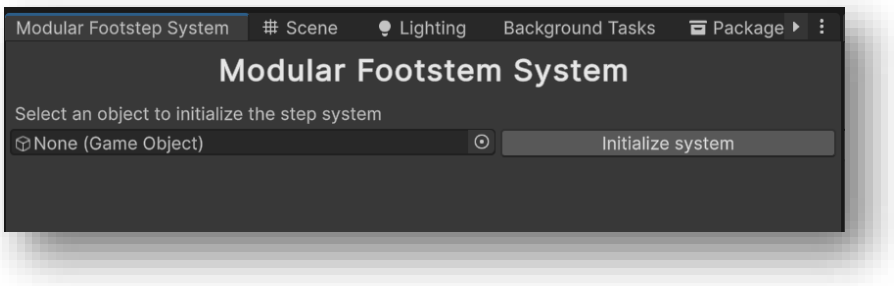
Adding Footstep Effects to Entities

There are two ways to add footstep effects to an entity. The first method involves using the configuration window, while the second involves manually adding and configuring components. Both methods are described below.

Using the Configuration Window

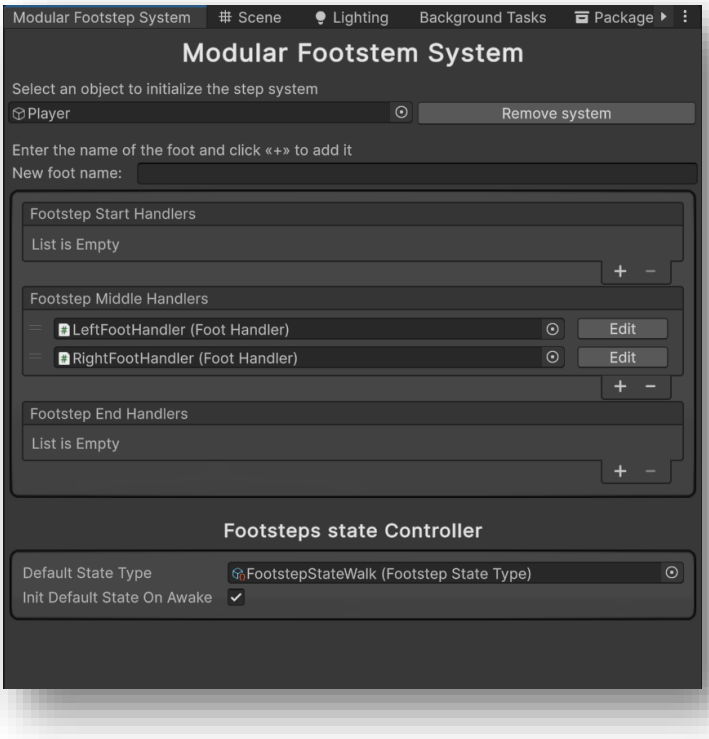
1. **Open the Configuration Window**
    - Use the shortcut **CTRL + M** or open the menu: **Window → Modular Footstep System**.





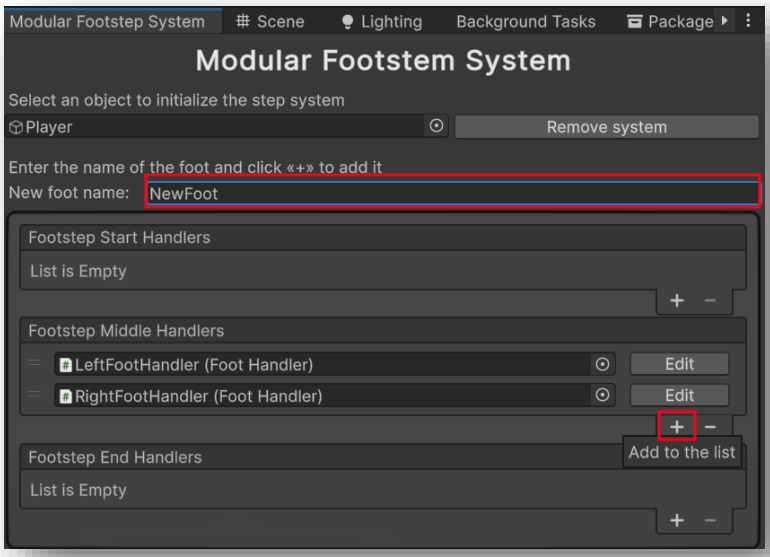
2. **Assign the Root Object**
    - In the opened window, set the root object of the entity from the game scene in the empty field. This is necessary to configure the footstep system for the selected object.
    - If the system has not been installed on this object before, click the **Initialize** button.
    - If the system was already installed, reinitialization is unnecessary. After selecting the object, the settings for the footstep system will load automatically.

Important! The entity or one of its child objects must have an **Animator** component; otherwise, the system will not initialize. This is because the system works with animation events.



3. **Add Foot Handlers**
    - After initialization, enter a logical name for the foot and add it to the desired handlers list by clicking the **+** button.
    - There are three handlers lists:
        - **Start Handlers** – Handles the creation of walking effects at the **START** of a step.
        - **Middle Handlers** – Handles the creation of walking effects in the **MIDDLE** of a step.
        - **End Handlers** – Handles the creation of walking effects at the **END** of a step.

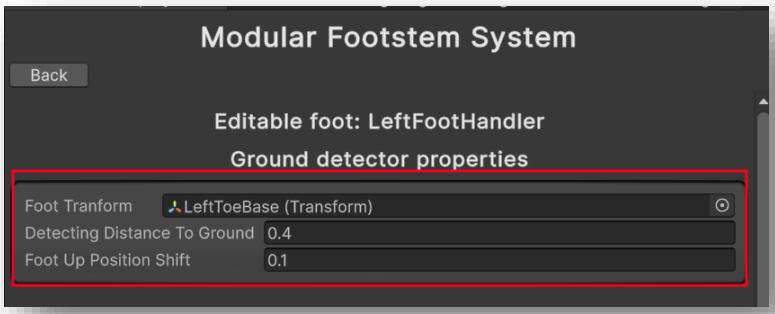
Important! The timing for triggering the **Start**, **Middle**, and **End** handlers is determined by the developer and must be set up in the entity's movement animations.



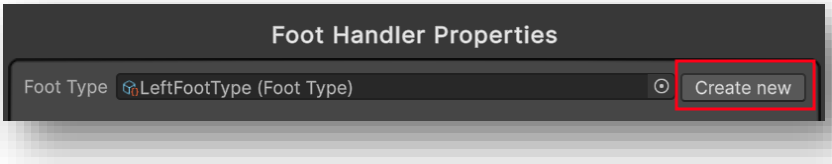
4. **Edit the Foot Handler**
    - After adding a foot handler, configure its effect creation by clicking the **Edit** button located next to the corresponding handler.

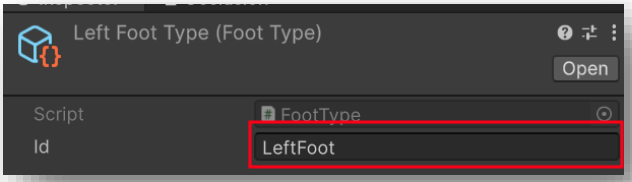
5. **Configure the Ground Detector**
    - In the updated window, configure the **GroundDetector** for proper surface detection under the foot:
        - **Foot Transform** – Specify the **Transform** component of the foot, which will be used to detect the ground surface.
        - **Detecting Distance To Ground** – Set the maximum distance for ground detection. If no surface is found within this distance, no effects will be created.
        - **Foot Up Position Shift** – Optionally adjust the vertical offset of the point used for detecting the ground.

      Important! Do not set the value in this field higher than the distance specified in the **Detecting Distance To Ground** field.

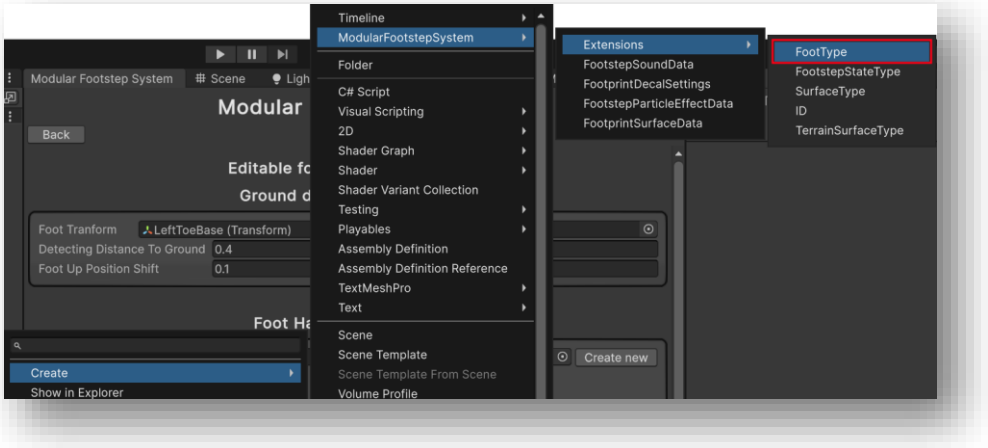


6. **Define the Foot Type**
   - Set the type of foot in the **Foot Type** field.
   - If foot types have not been created yet, click **Create New** to create a new foot type. A window will appear prompting you to choose a save path for the **ScriptableObject**. Enter a unique **ID** for the foot type to help identify which foot is stepping and configure the corresponding effects.





Alternatively, create a **ScriptableObject** manually via the context menu: **Create → ModularFootstepSystem → Extensions → FootType**. After creation, enter a unique textual **ID** and assign the **ScriptableObject** to the **FootType** field.



7. **Add Effect Modules**
    - The system supports three effect creation modules:
        - **Footprints Module**
        - **Footstep Sounds Module**
        - **Footstep Effects Module**

    Add the required modules and configure their settings as described below.

Effect Modules Configuration

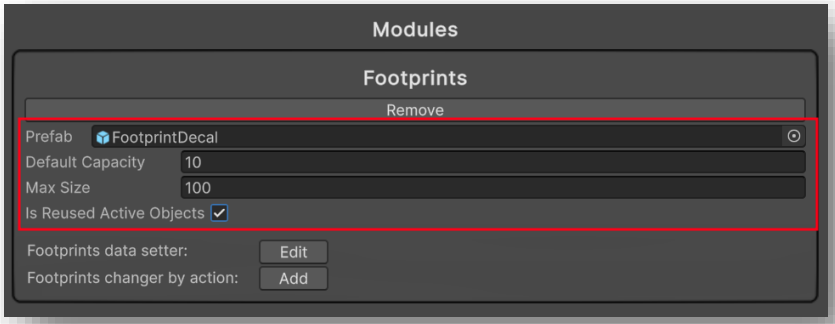
Footprints Module

This module generates footprints using decals.

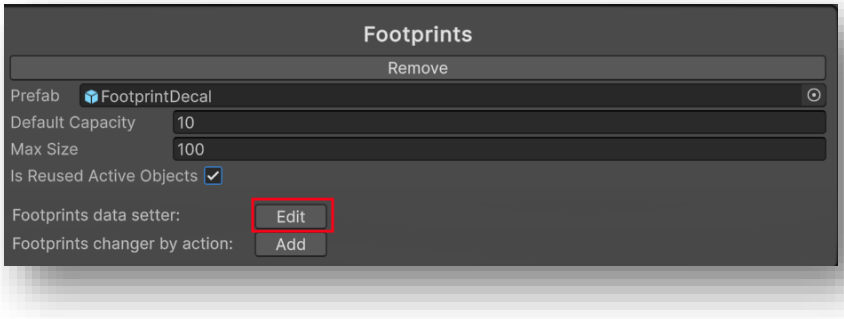
1. Add the module and configure its settings:
    - **Prefab** – Assign the footstep decal prefab. The prefab must include a **Projector** component based on the rendering pipeline:
        - For **URP**: **URPDecalProjector**
        - For **HDRP**: **HDRPDecalProjector**
        - For **Built-in**: **Projector**

      Detailed examples of decal configurations for each rendering pipeline can be found in the plugin's demo implementation. The prefab must also include the **PooledObject** component, which allows reusing decals if their quantity exceeds the pool's capacity.

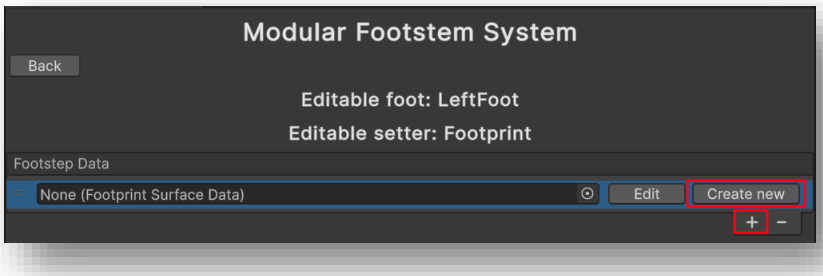
    - **Default Capacity** – Set the default size of the object pool. **Important!** This value must not exceed the value in the **Max Size** field.
    - **Max Size** – Set the maximum number of objects in the pool (the maximum number of footprints that can be created). **Important!** This value must not be smaller than the value in the **Default Capacity** field.
    - **Is Reused Active Objects** – Enable this option to reuse active footprint objects if new ones cannot be created due to the pool's size limit.

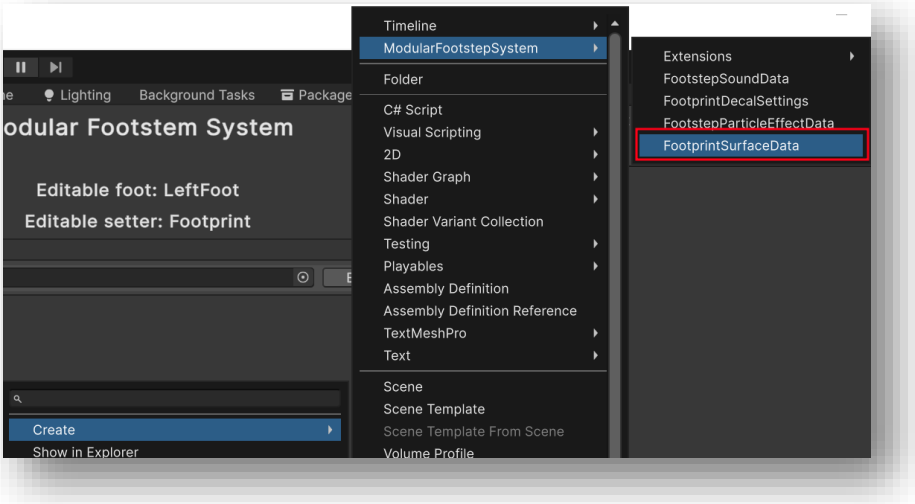


2. To specify different footprints for various surfaces, edit the **Footprints Data Setter**:
    - Click the **Edit** button next to this setting.

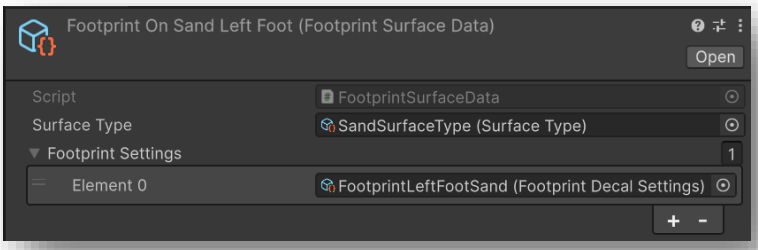


    - In the updated window, add surface-specific footprint data:
        - Click the **+** button and create a new **ScriptableObject** by selecting **Create New** or manually via the context menu: **Create → ModularFootstepSystem → FootprintSurfaceData**.



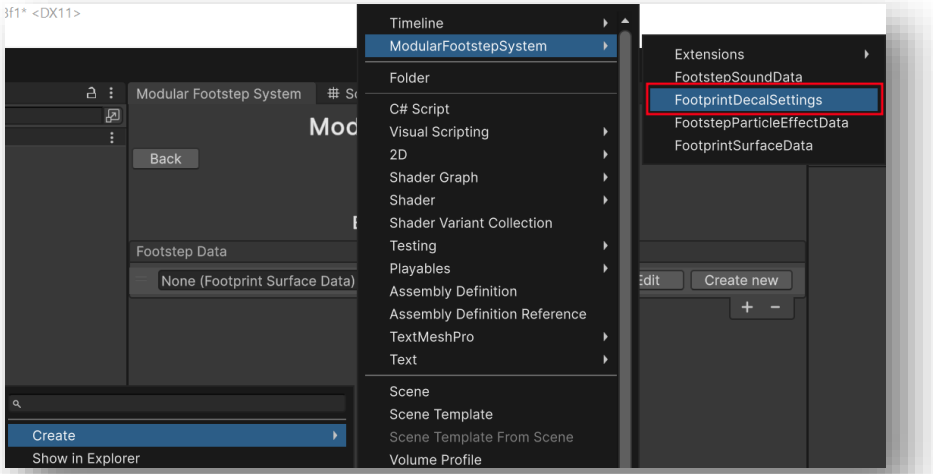


- Assign a surface type in the **SurfaceType** field.
- Add decal settings to the **FootprintSettings** list for surface-specific variation. If no variation is needed, add only one entry.

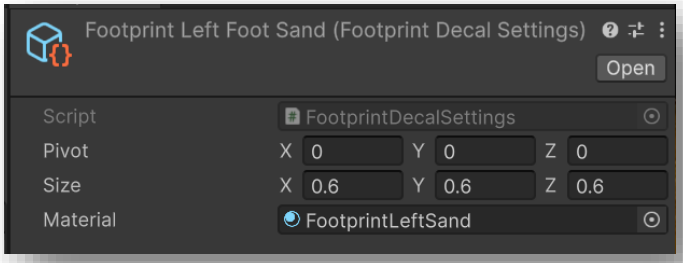


These settings are configured in the **DecalProjector** component (or **Projector** for **Built-in**) of the footprint object. This way, only one type of footprint object is used, but with modified decal component parameters for each object.

A **ScriptableObject** for decal settings can be created via the context menu in the Project tab: **Create → ModularFootstepSystem → FootprintDecalSettings**.



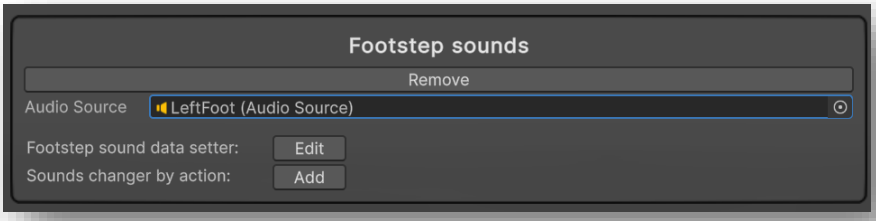
Depending on the rendering pipeline used (**Built-in**, **URP**, or **HDRP**), you need to specify the parameters of the **DecalProjector** (or **Projector** for **Built-in**) component and the decal material in the **FootprintDecalSettings**.



Footstep Sounds Module

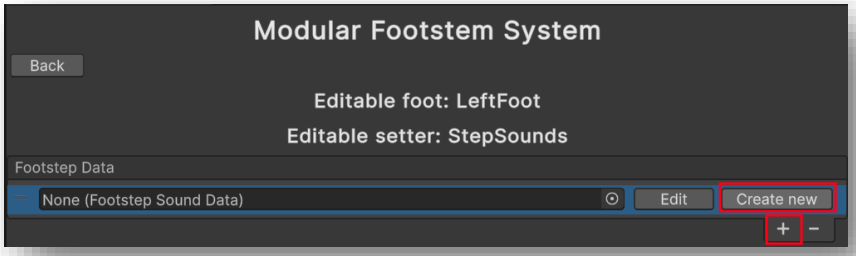
This module plays footstep sounds for different surfaces.

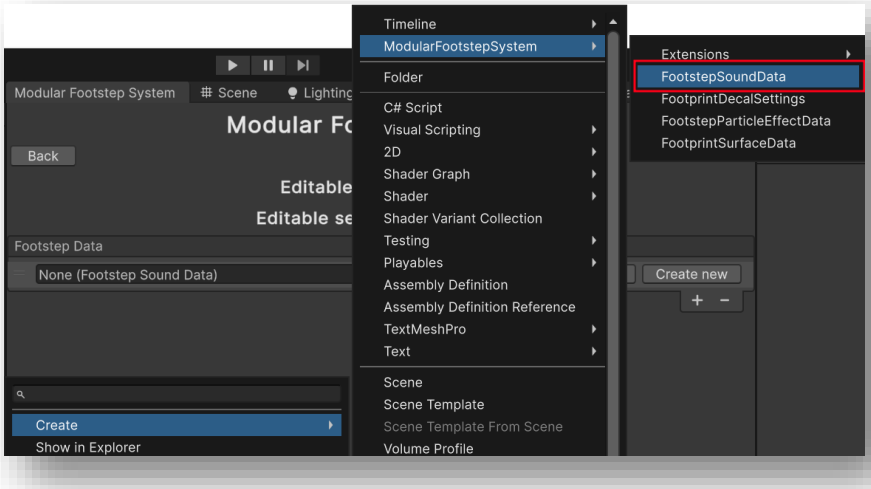
1. Add the module and specify an **AudioSource** component in the **Audio Source** field. Place the **AudioSource** near or on the foot.



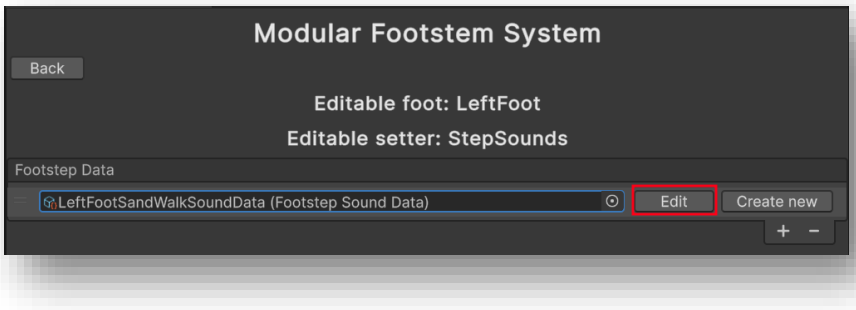
2. Edit the **Footstep Sound Data Setter**:
    - Click the **Edit** button and add sound data for various surfaces.

    - Click **+** to create a new **ScriptableObject** (**Create New**) or manually via the context menu: **Create → ModularFootstepSystem → FootstepSoundData**.

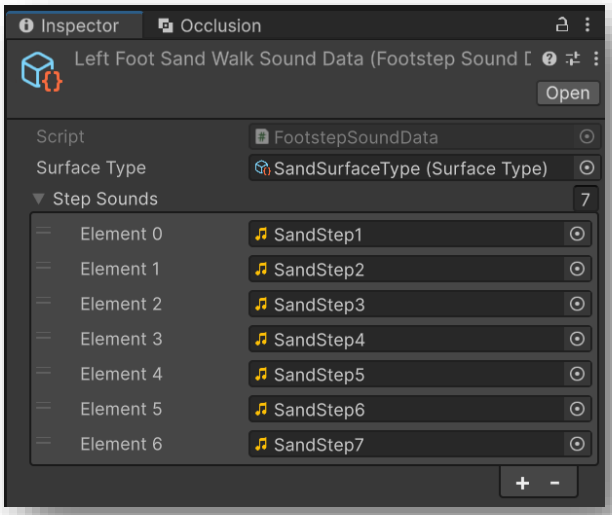




    - After creation, you need to select the **ScriptableObject** or click the **Edit** button next to the required data in the editing window.



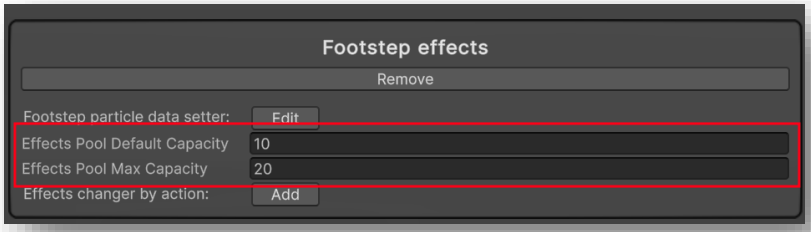
- Assign a surface type in the **SurfaceType** field and add **AudioClips** to the **StepSounds** list for sound variation on the same surface. The list is required for varied alternation of sounds on the same surface. If alternation is not needed, only one entry should be added to the list.



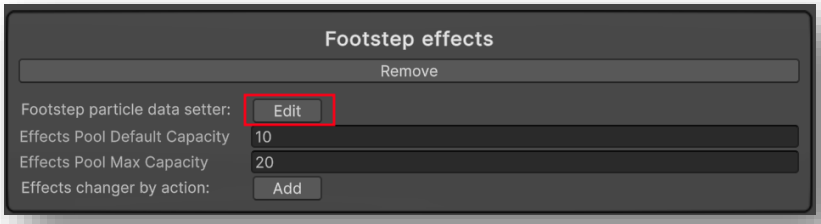
Footstep Effects Module

This module generates particle effects for different surfaces.

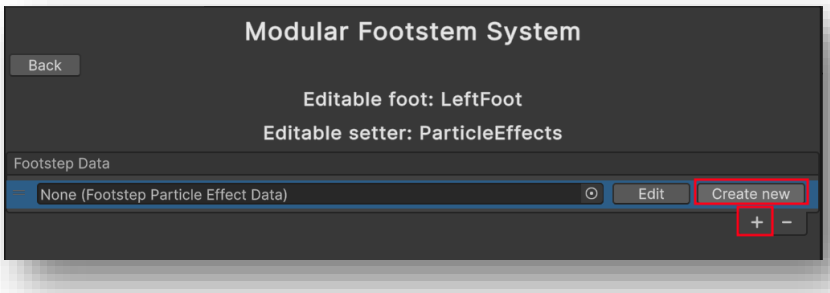
1. Add the module and configure the object pool:
   - **Effects Pool Default Capacity** – Set the default pool size. **Important!** This value must not exceed the **Effects Pool Max Size**.
   - **Effects Pool Max Size** – Set the maximum number of particle effects that can be created.
     Important! This value must not be smaller than the **Effects Pool Default Capacity**.

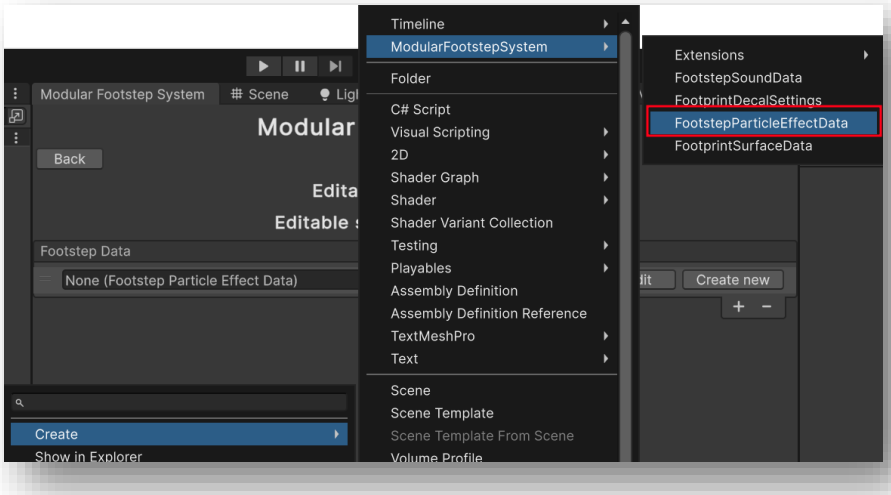


2. Edit the **Footstep Particle Data Setter**:
    - Click the **Edit** button and add particle data for various surfaces.

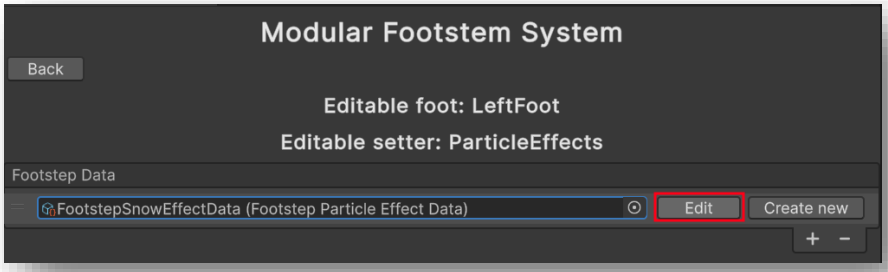


    - Create a new **ScriptableObject** (**Create New**) or manually via the context menu: **Create → ModularFootstepSystem → FootstepParticleEffectData**.

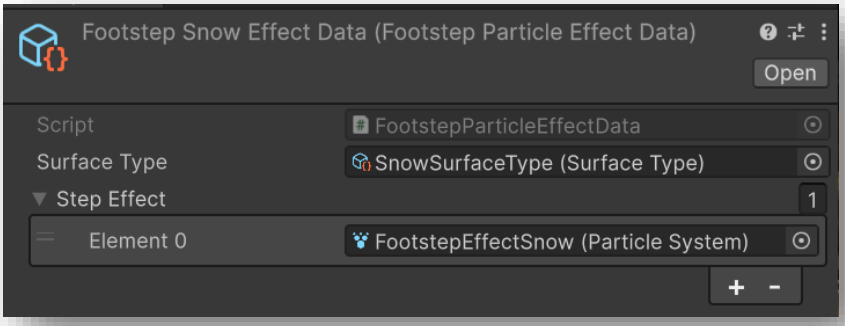


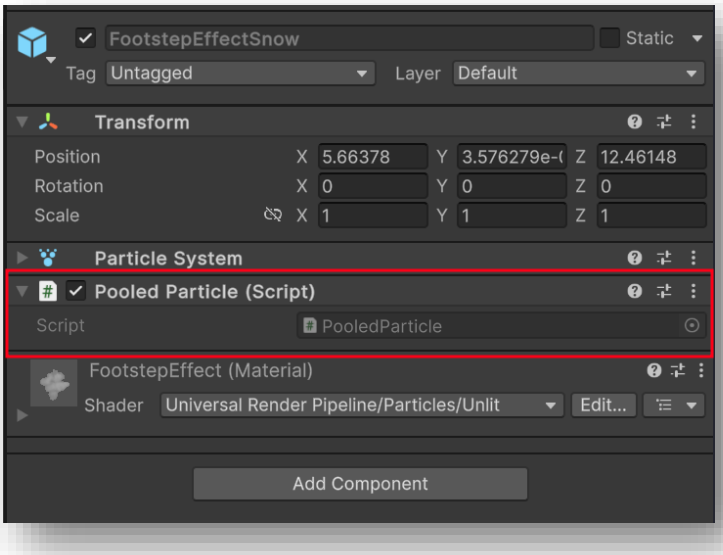


- After creation, you need to select the **ScriptableObject** or click the **Edit** button next to the required data in the editing window.

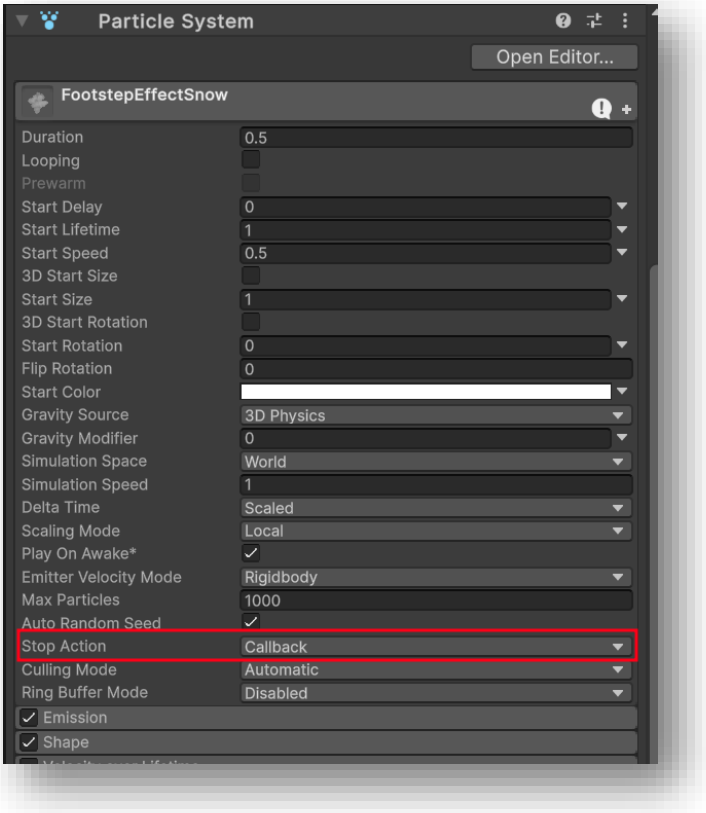


- Assign a surface type in the **SurfaceType** field and add particle effect prefabs to the **StepEffect** list. Each prefab should include a **ParticleSystem** component and a **PooledParticle** component to ensure it returns to the pool after playback. The list is required for varied alternation of sounds on the same surface. If alternation is not needed, only one entry should be added to the list.





Important! To ensure the **PooledParticle** component returns the object to the pool, you need to set the **StopAction** parameter to **Callback**.



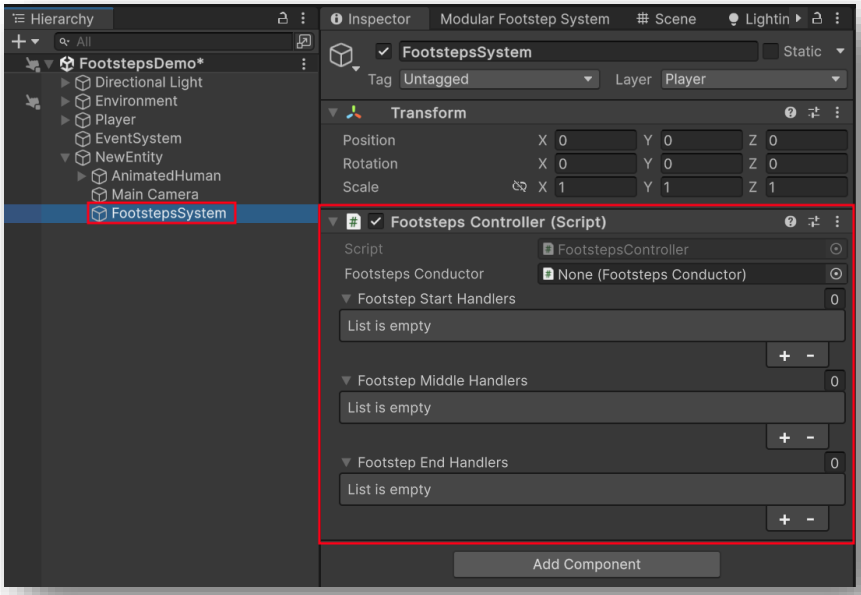
Manual Addition and Configuration of Components

Manual addition of components is useful for understanding the relationships between scripts, allowing you to create custom effect modules based on abstractions or modify existing ones if their logic does not suit your needs.

The creation and configuration process is divided into the following steps:

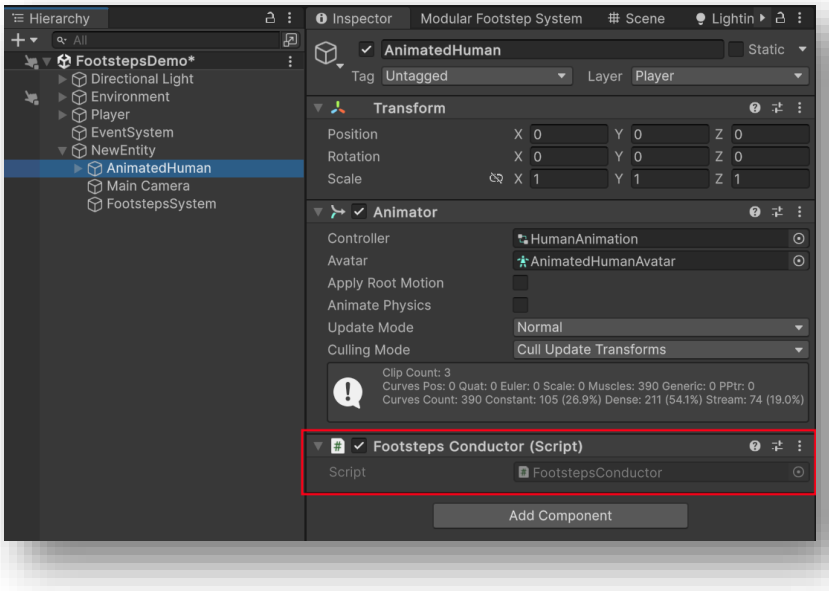
Step 1: Creating and Configuring the System Controller

1. **Add the FootstepsController Component**
    - Create an empty object under the entity to house the footstep system.
    - Add the **FootstepsController** component to this object.

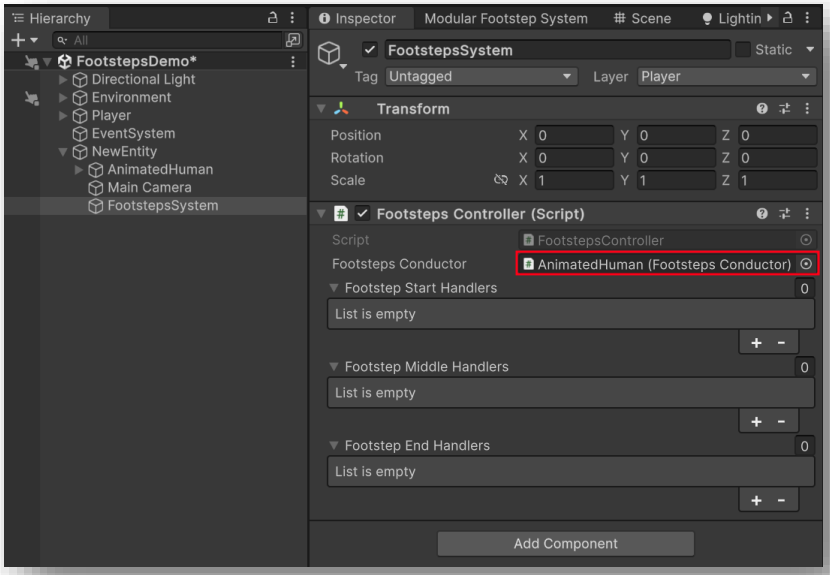


2. **Add the FootstepsConductor Component**

   - Add the **FootstepsConductor** component to the object containing the Animator.

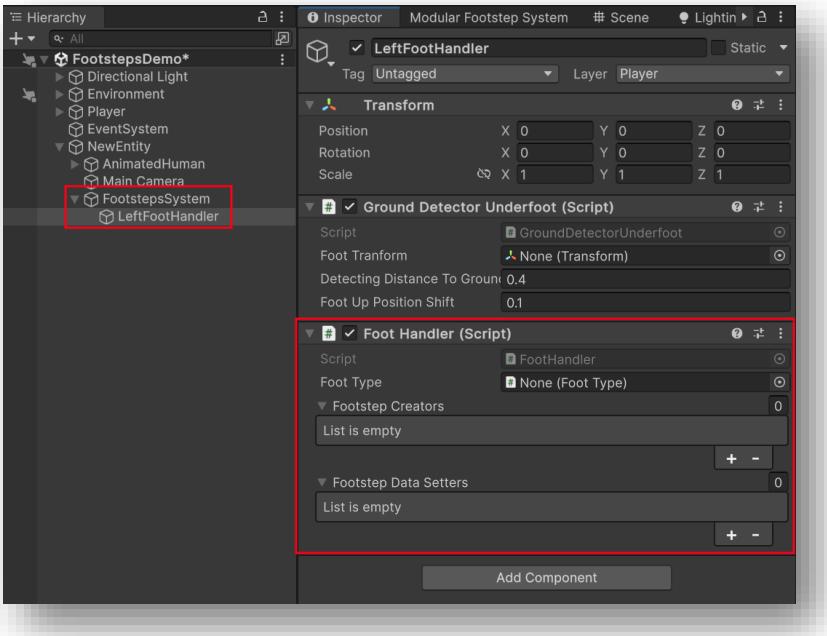


   - Link this component to the **FootstepsController** by assigning it in the **FootstepsConductor** field of the **FootstepsController** component.

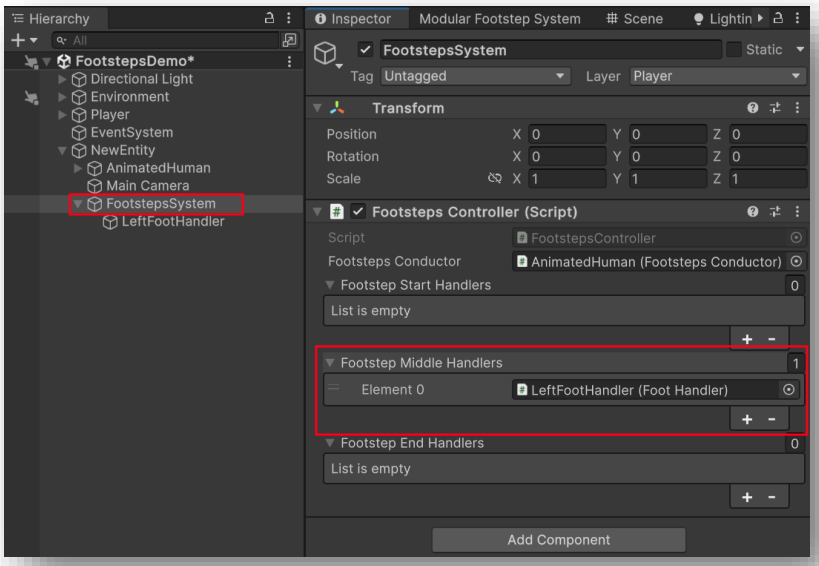


Step 2: Creating and Configuring a Foot Handler

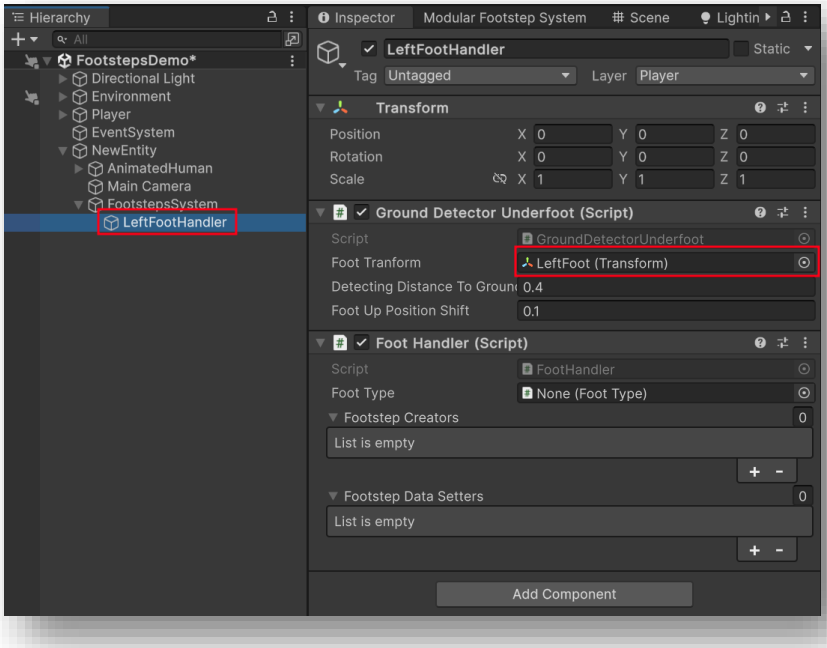
1. **Add the FootHandler Component**
    - Create a new child object under the entity.
    - Attach the **FootHandler** component to this object.



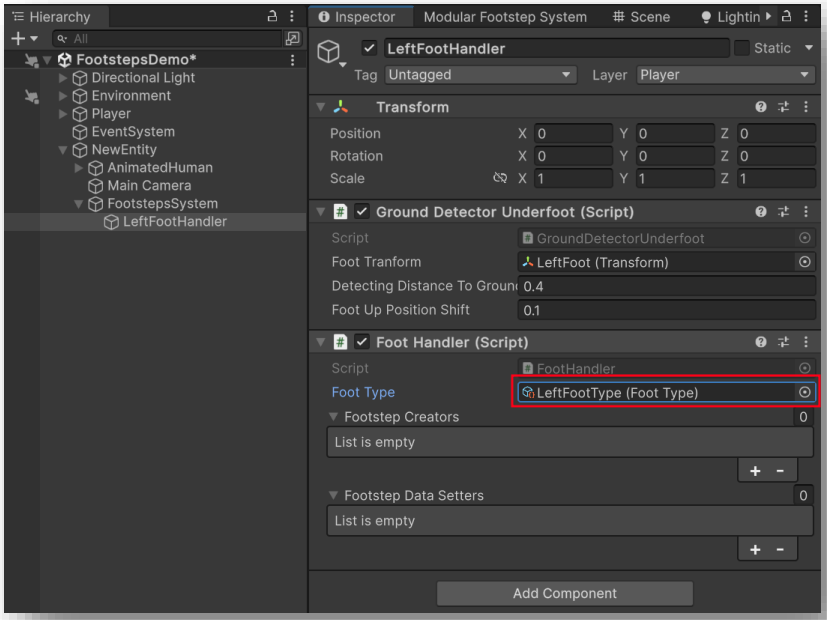
    - Add the **FootHandler** to one of the three handler lists in the **FootstepsController** component:
        - Start Handlers
        - Middle Handlers
        - End Handlers



2. **Configure the GroundDetectorUnderfoot Component**
    - When the **FootHandler** is created, the **GroundDetectorUnderfoot** component is automatically added.
    - In its **FootTransform** field, specify the **Transform** of the foot that will be used to detect the ground surface.



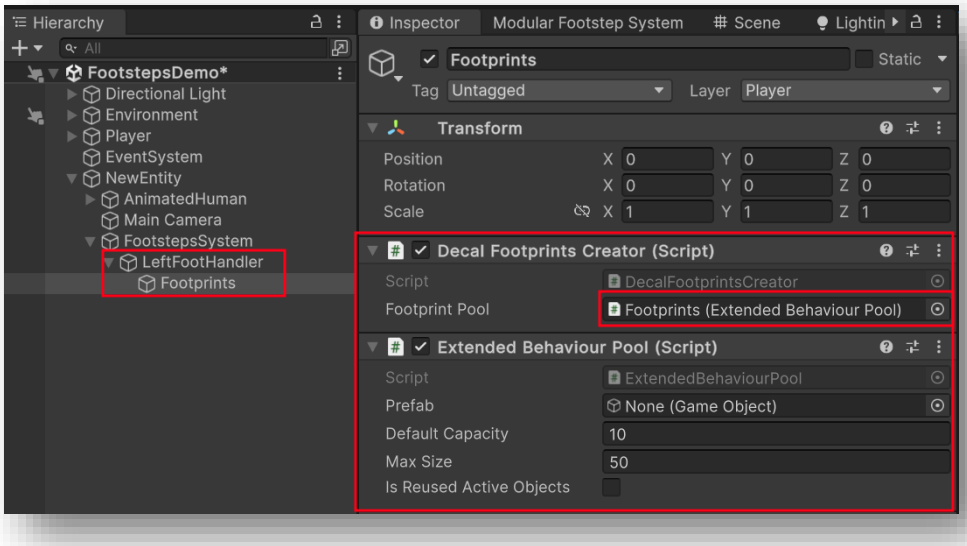
    - Assign the foot type in the **FootType** field. Details on creating foot types are provided earlier.



At this stage, the system will process the newly created foot but will not yet generate effects.

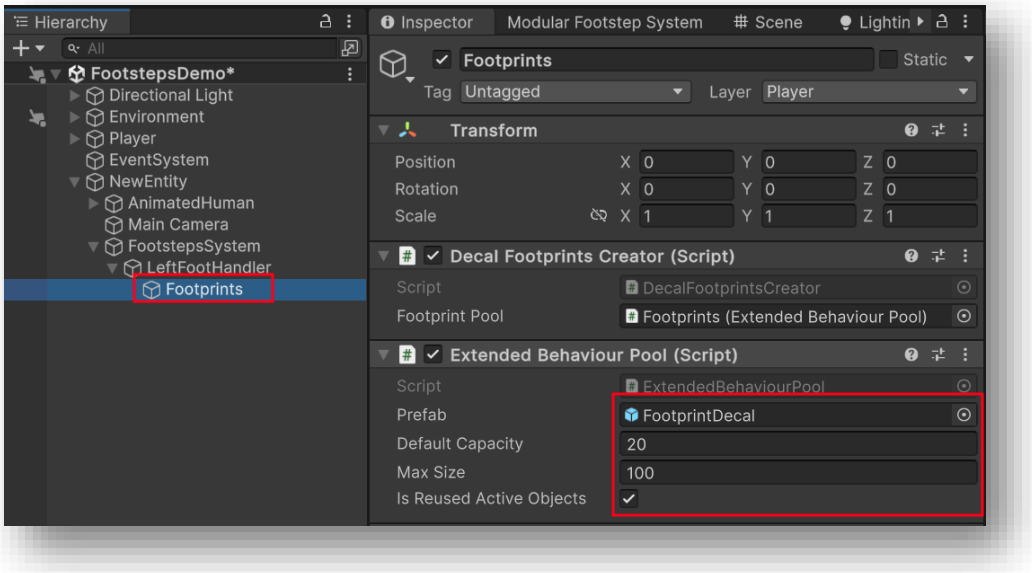
Step 3: Adding and Configuring the Footprints Module

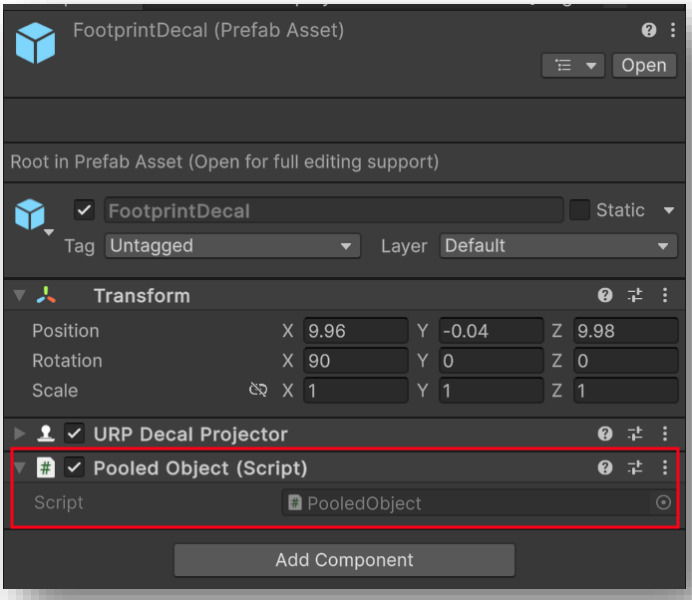
1. **Add the DecalFootprintsCreator Component**
    - Create a new child object under the **FootHandler**.
    - Attach the **DecalFootprintsCreator** and **ExtendedBehaviourPool** components to this object.
    - Link the **ExtendedBehaviourPool** to the **FootprintPool** field of the **DecalFootprintsCreator**.



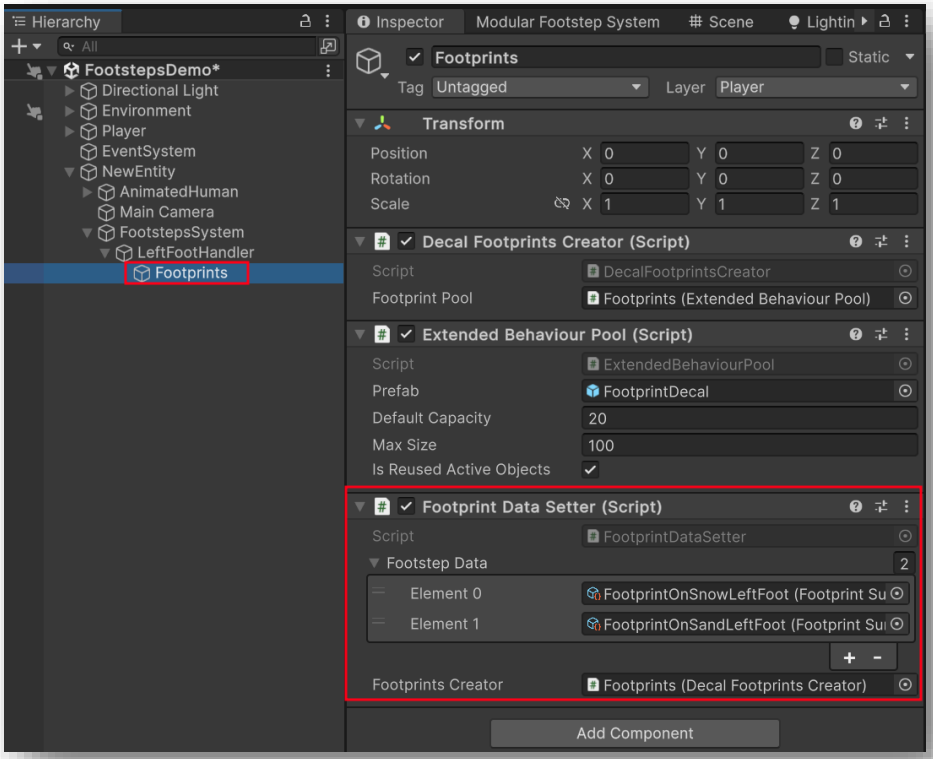
2. **Configure the Footprints Pool**
    - Add a footstep decal prefab to the pool.

- Configure pool parameters as described earlier.
- Attach the **PooledObject** component to the decal prefab so that it returns to the pool after use.

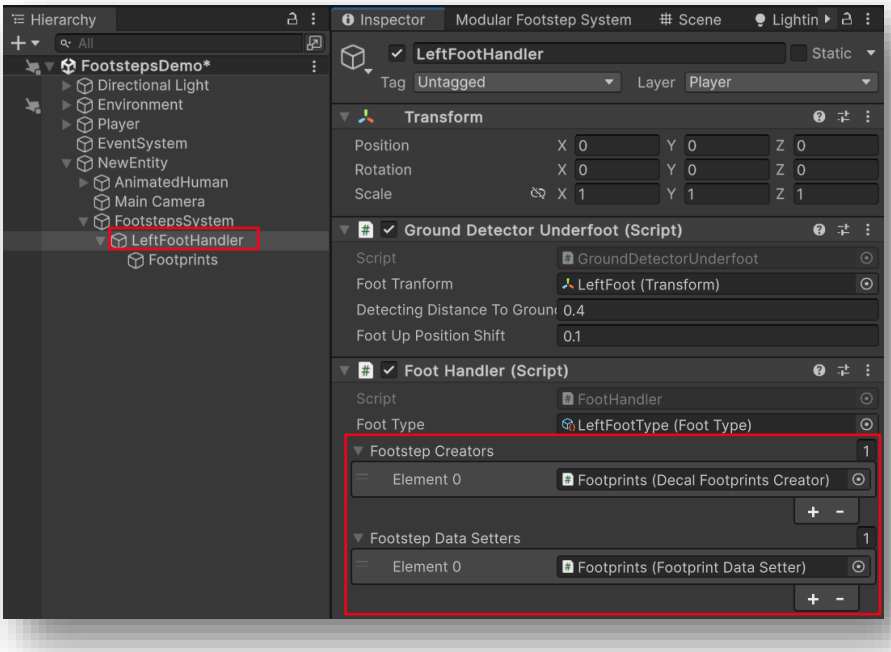




3. **Add and Configure the FootprintDataSetter Component**
   - Attach the **FootprintDataSetter** component to the same object.
   - Link the **FootprintDataSetter** to the **DecalFootprintsCreator**.
   - Add surface-specific decal settings to the **FootstepData** list. Details on creating and configuring decal settings are provided earlier.

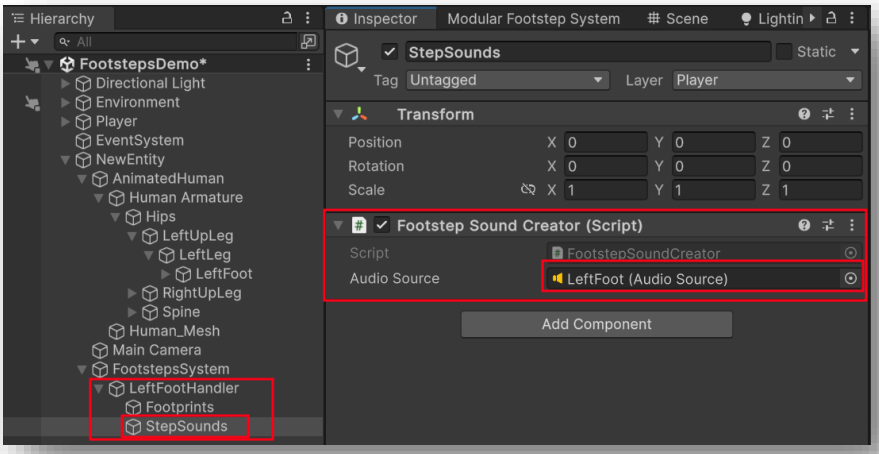


4. **Link Components to the Foot Handler**
    - Add the **DecalFootprintsCreator** and **FootprintDataSetter** components to their respective lists in the **FootHandler**.

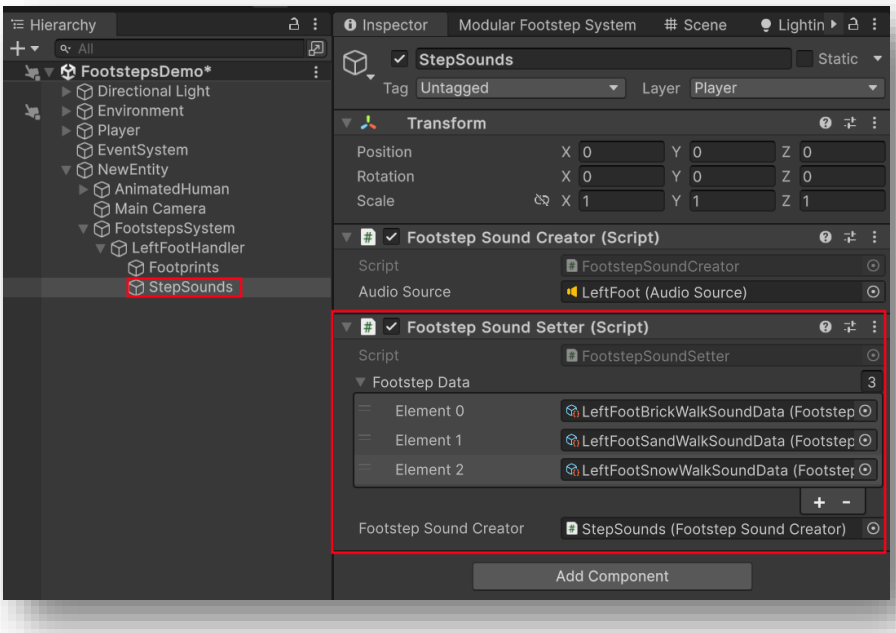


Step 4: Adding and Configuring the Footstep Sounds Module

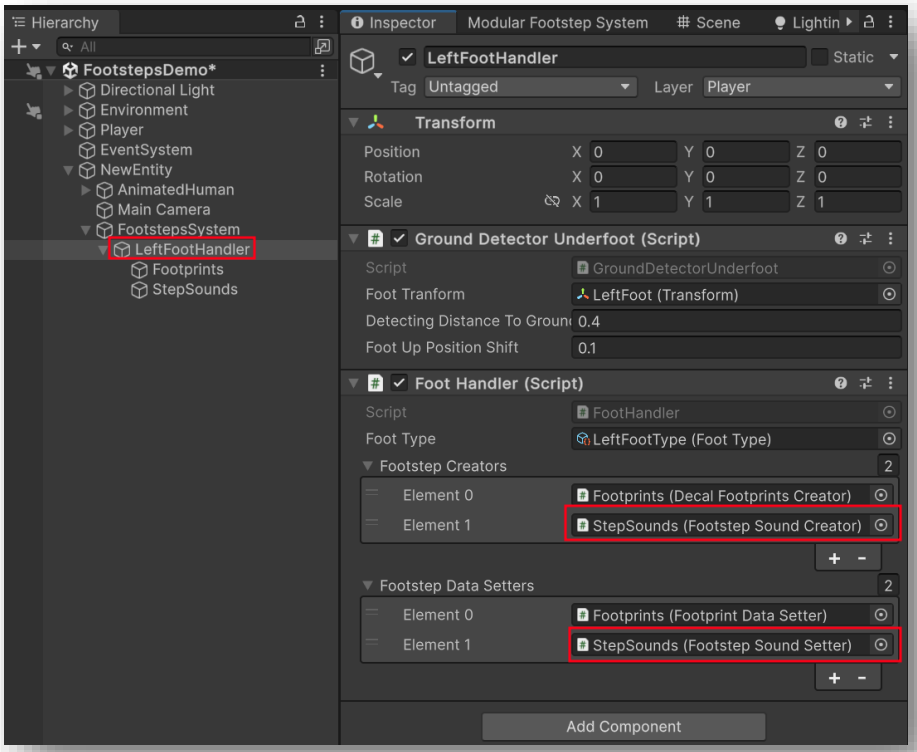
1. **Add the FootstepSoundCreator Component**
    - Create a new child object under the **FootHandler**.
    - Attach the **FootstepSoundCreator** component to this object.
    - Specify an **AudioSource** component in the **Audio Source** field. Place the **AudioSource** near or on the foot.



2. **Add and Configure the FootstepSoundSetter Component**
    - Attach the **FootstepSoundSetter** component to the same object.
    - Link the **FootstepSoundSetter** to the **FootstepSoundCreator**.
    - Add sound settings for different surfaces to the **FootstepData** list. Details on creating and configuring sound settings are provided earlier.

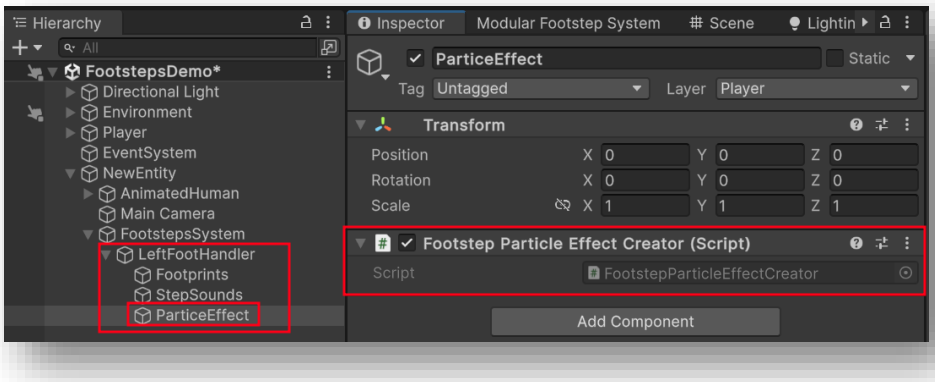


3. **Link Components to the Foot Handler**
    - Add the **FootstepSoundCreator** and **FootstepSoundSetter** components to their respective lists in the **FootHandler**.



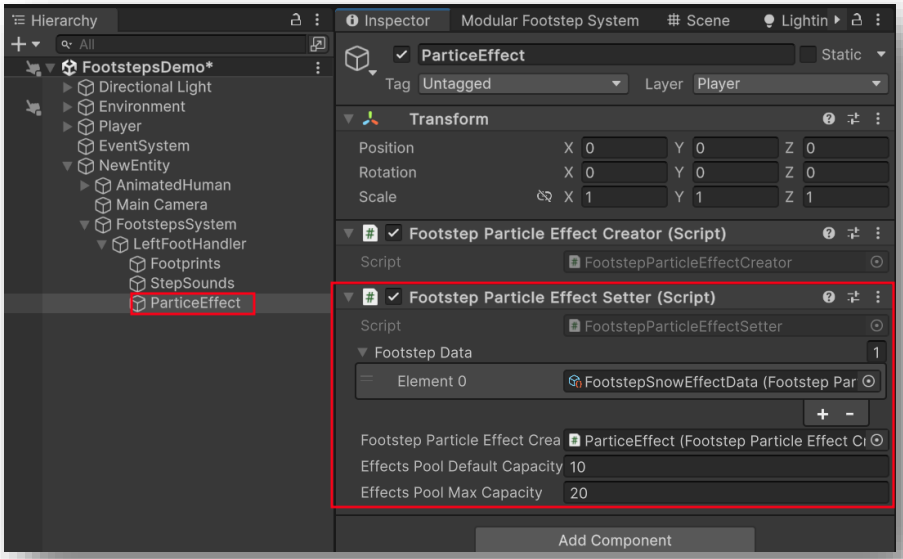
Step 5: Adding and Configuring the Footstep Effects Module

1. **Add the FootstepParticleEffectCreator Component**
    - Create a new child object under the **FootHandler**.
    - Attach the **FootstepParticleEffectCreator** component to this object.

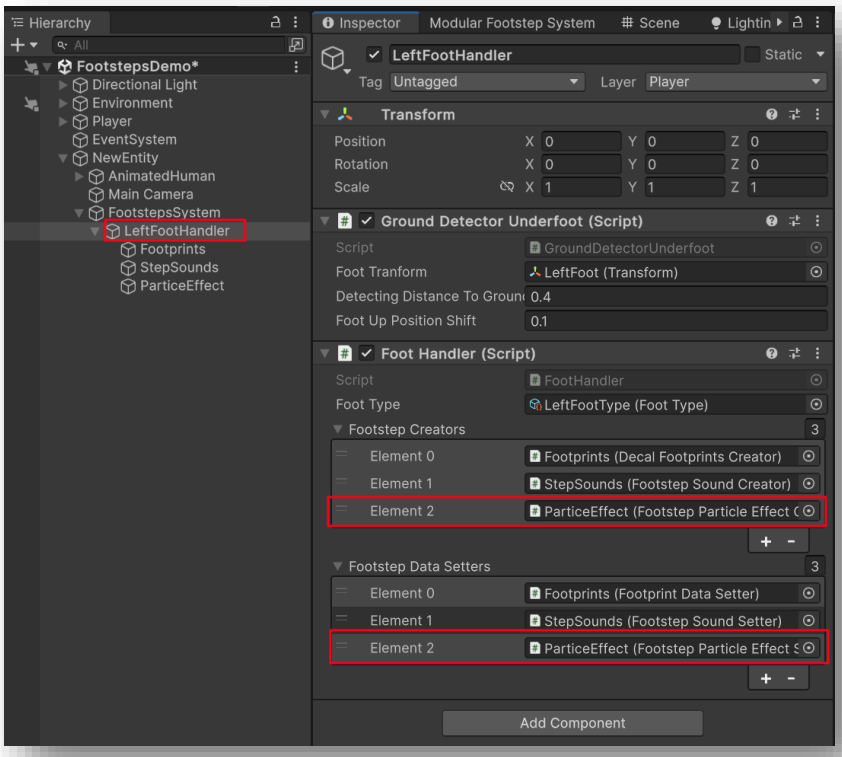


2. **Add and Configure the FootstepParticleEffectSetter Component**
    - Attach the **FootstepParticleEffectSetter** component to the same object.
    - Link the **FootstepParticleEffectSetter** to the **FootstepParticleEffectCreator**.

- Add particle effect settings for different surfaces to the **FootstepData** list. Details on creating and configuring particle settings are provided earlier.



3. **Link Components to the Foot Handler**
   - Add the **FootstepParticleEffectCreator** and **FootstepParticleEffectSetter** components to their respective lists in the **FootHandler**.



Configuring Animations to Send Events

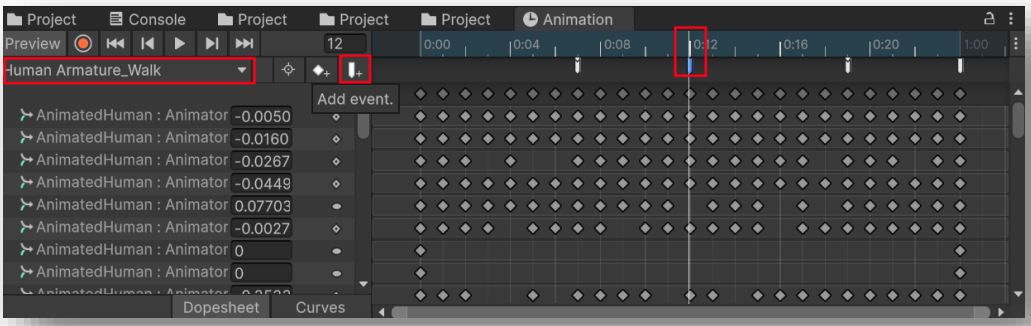
The basic setup of the footstep system is now complete. However, effects are still not being created or triggered because the system operates through events sent by animations, and these events need to be configured. Follow these steps to set up the animation events:

Step 1: Open the Animation Editor

1. Select the entity object in your scene.
2. Open the animation editing window.
3. Choose the animation you wish to modify.

Step 2: Add an Event

1. Identify the frame in the animation where the entity's foot fully contacts the ground.
2. Add an event at this frame.

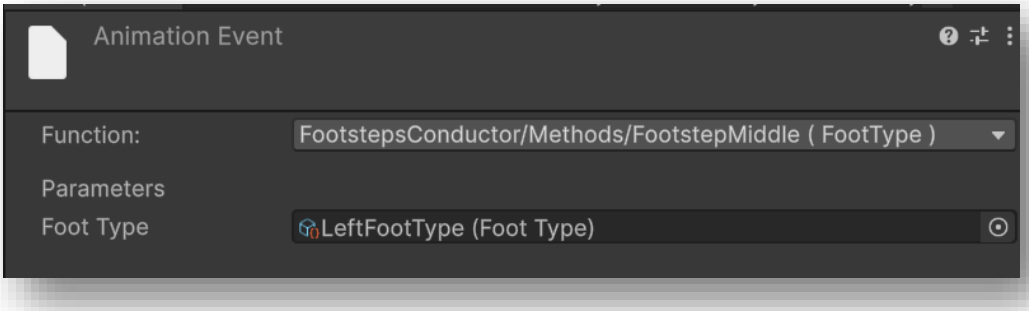


Step 3: Configure the Event

1. Select the event in the timeline.
2. In the **Function** field, choose the **FootstepMiddle** method from the **FootstepsConductor** component.

   Important! Ensure the **FootstepsConductor** component is on the same object as the **Animator**. Otherwise, the **FootstepMiddle** method will not appear in the list of functions.

3. In the **Foot Type** field, specify the foot type that is stepping (e.g., left or right).



Step 4: Additional Events (Optional)

- If necessary, repeat the steps above to create events for the beginning and end of the step.
- Use the following functions:
    - **FootstepStart** – Triggers effects at the start of the step.
    - **FootstepEnd** – Triggers effects at the end of the step.

When the events are correctly configured and aligned with the animation, the system will be ready to work. Effects will trigger as expected based on the animation events.

What If I Need Different Effects When Changing the Entity's State?

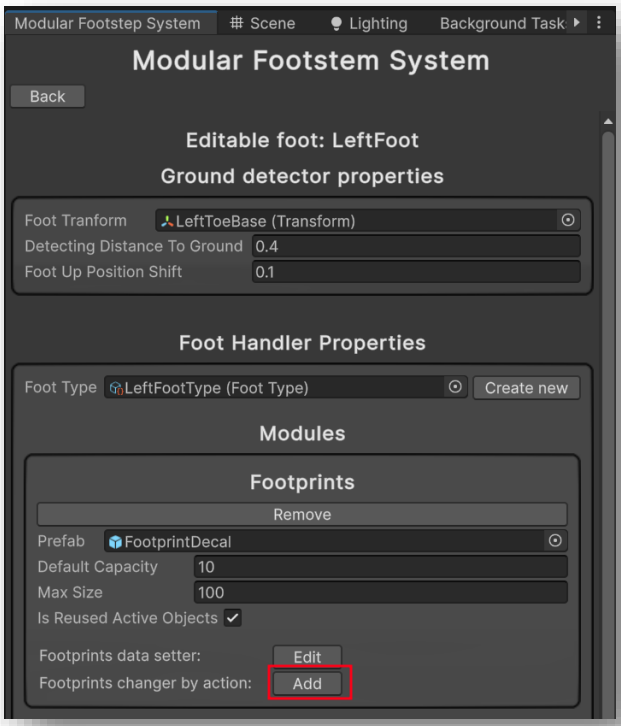
Let's consider a scenario where the entity has two or more movement animations. For example, the entity may walk slowly and run fast. In this case, you may want to use different footstep sounds or effects depending on the movement type.

To address this, you need to extend the system configuration. Similar to the previous setup, there are two methods:

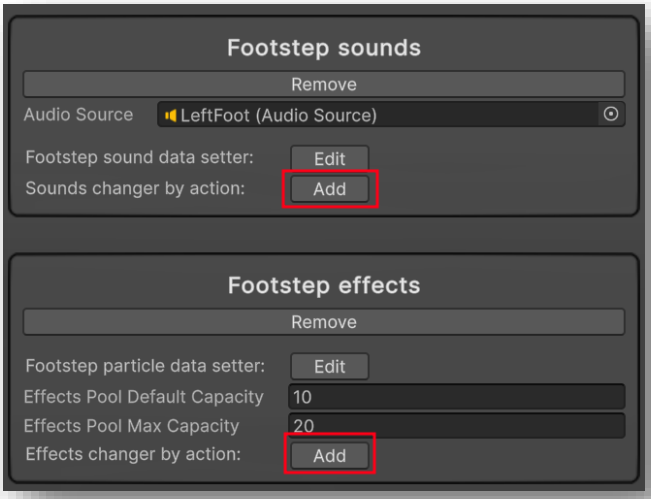
1. Using the Configuration Window
2. Manual Addition and Configuration of Components

Using the Configuration Window

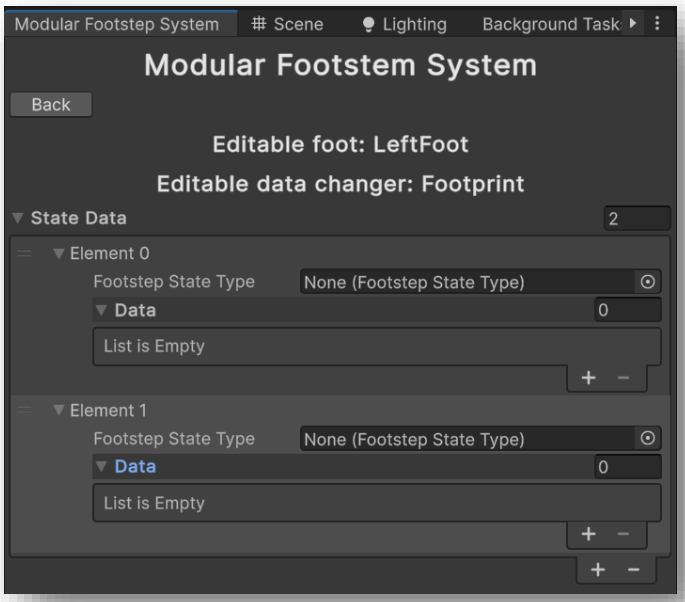
1. **Add Changers to Modules**
    - Open the configuration window.
    - To enable state-based changes in the **Footprints Module**, click the **Add** button next to the **Footprints changer by action** option.



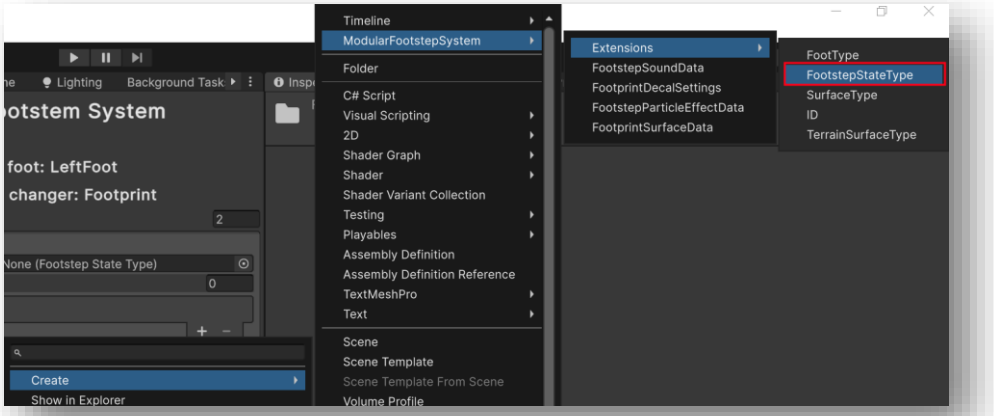
    - Similarly, add changers for other modules:
        - For the **Footstep Sounds Module**, click **Add** next to **Sounds changer by action**.
        - For the **Footstep Effects Module**, click **Add** next to **Effects changer by action**.

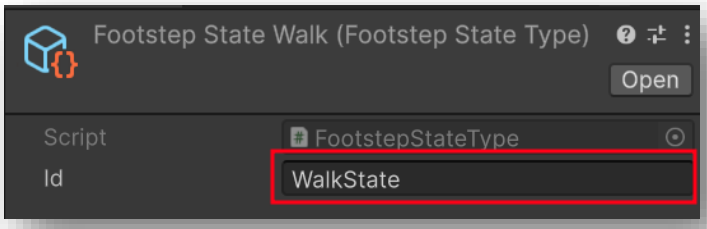


2. **Edit Changers**
    - Once a changer is added, the **Add** button will be replaced by **Edit** and **Remove** buttons.
    - Click **Edit** to open the changer's configuration window.
3. **Configure State Data**
    - Add entries to the **State Data** list for each entity state (e.g., walking, running).



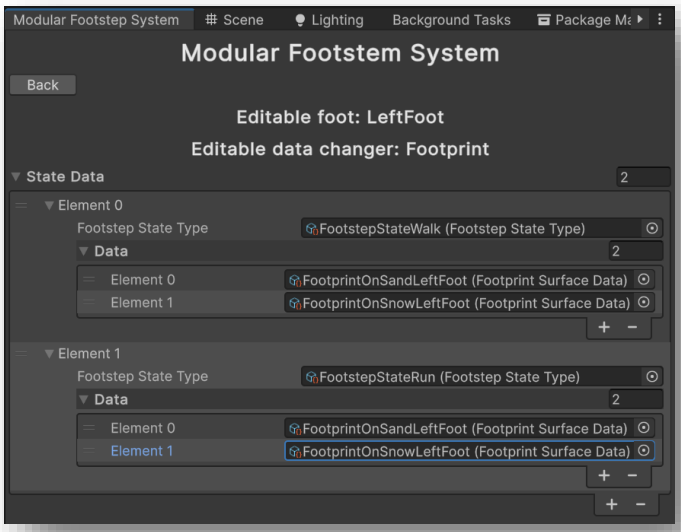
    - For each entry:
        - In the **Footstep State Type** field, assign a **ScriptableObject** representing the entity's state. Create this object via the context menu: **Create → ModularFootstepSystem → Extensions → FootstepStateType**. Assign a unique **ID** to each state.

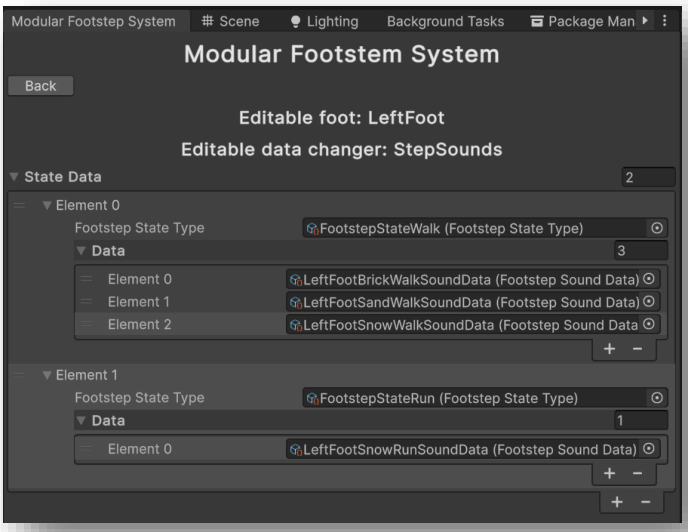


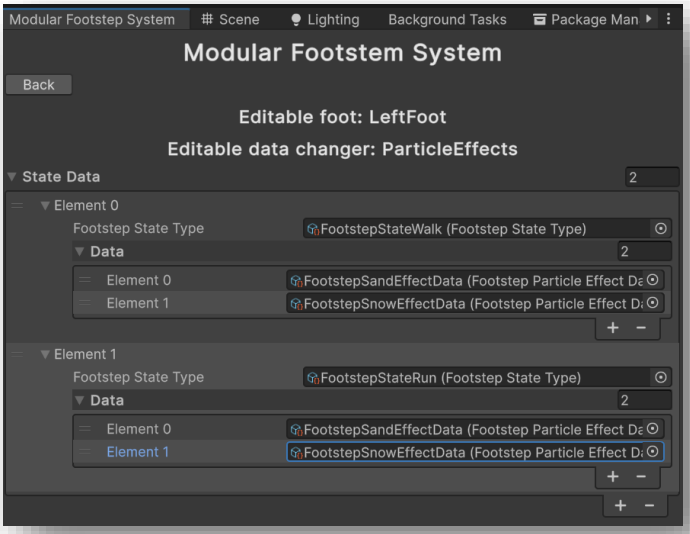


- In the **Data** list, add the **ScriptableObjects** containing settings for decals, sounds, or particle effects to be used for that state.

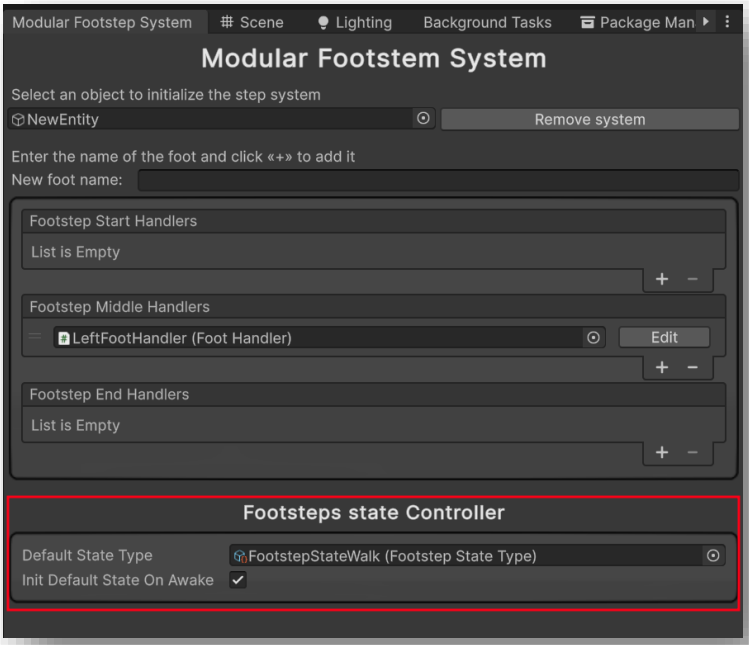
Important! Ensure that all surfaces used in the scene have settings defined for each state. Missing settings can cause issues where effects play in one state but fail in another.







4. **Set the Default State**
    - Return to the first tab in the configuration window by clicking the **Back** button in the upper left corner.
    - In the **Footsteps State Controller** section, set the default entity state in the **Default State Type** field.
    - Enable the **Init Default State On Awake** option if the default state should be set automatically when the scene starts.

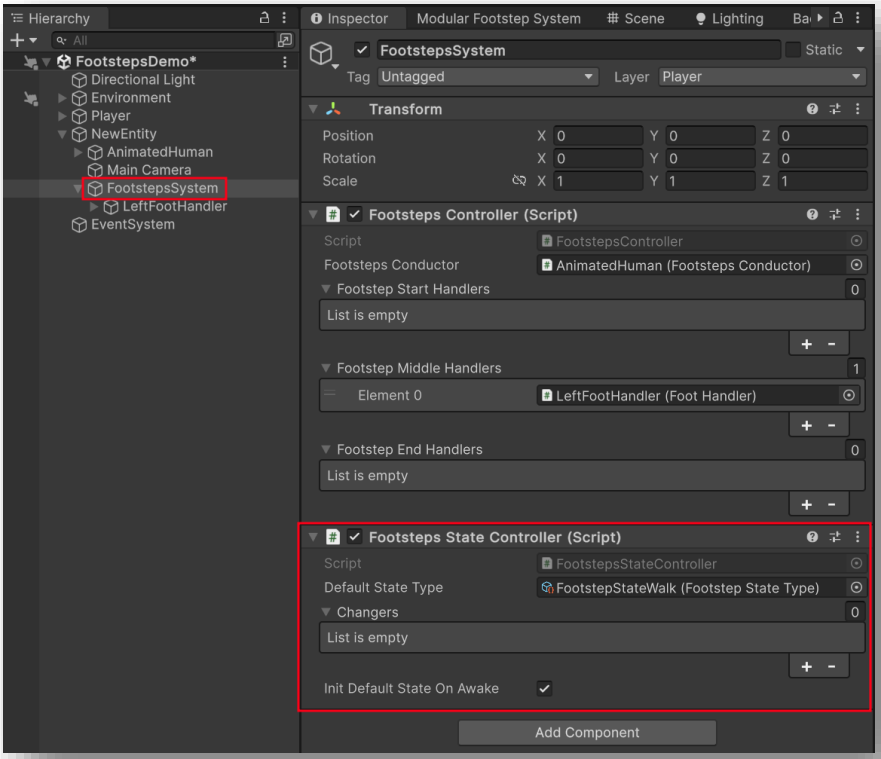


5. **Change States Programmatically**
    - To dynamically change states, use the **SetState** method of the **FootstepsStateController** component. Pass the appropriate **FootstepStateType ScriptableObject** to this method to switch the state.

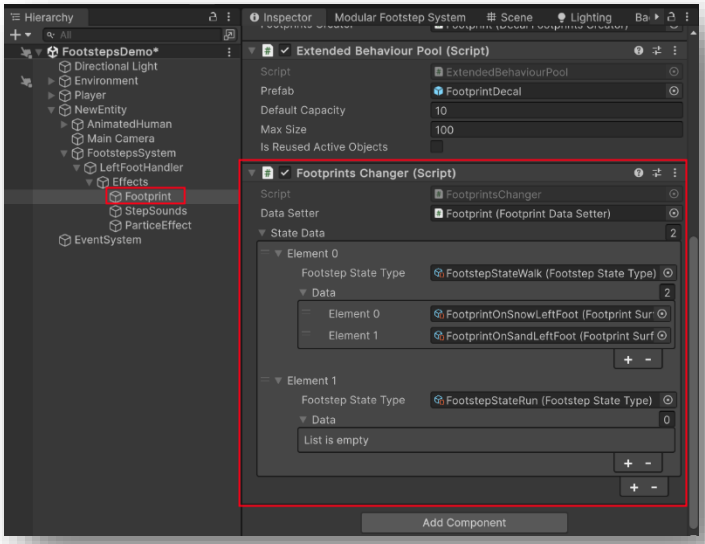
Manual Addition and Configuration of Components

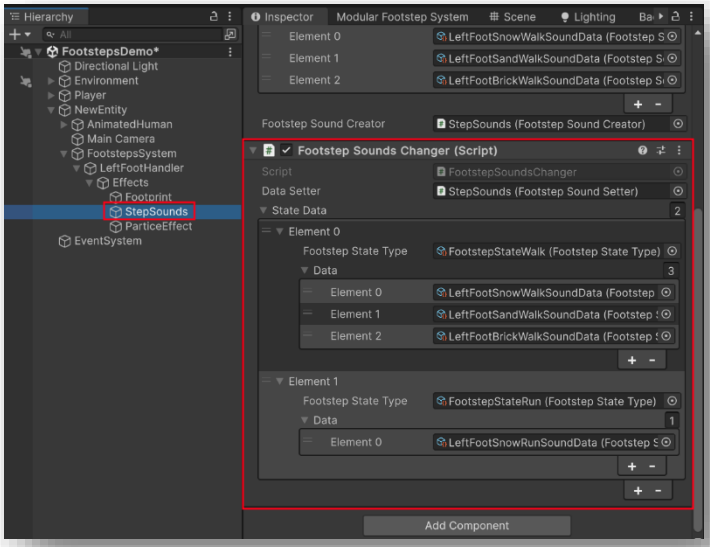
For manual configuration:

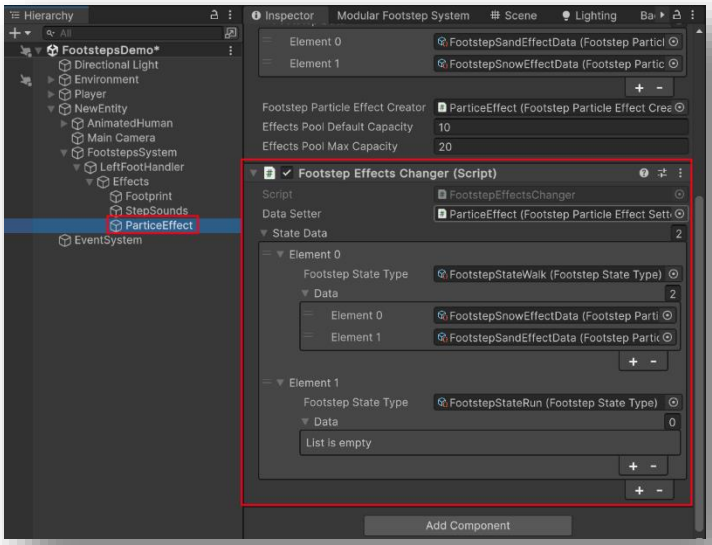
1. **Add the FootstepsStateController Component**
    - Attach the **FootstepsStateController** to the object containing the **FootstepsController**.
    - Set the default state in the **DefaultStateType** field.
    - Enable or disable the **Init Default State On Awake** flag as needed.



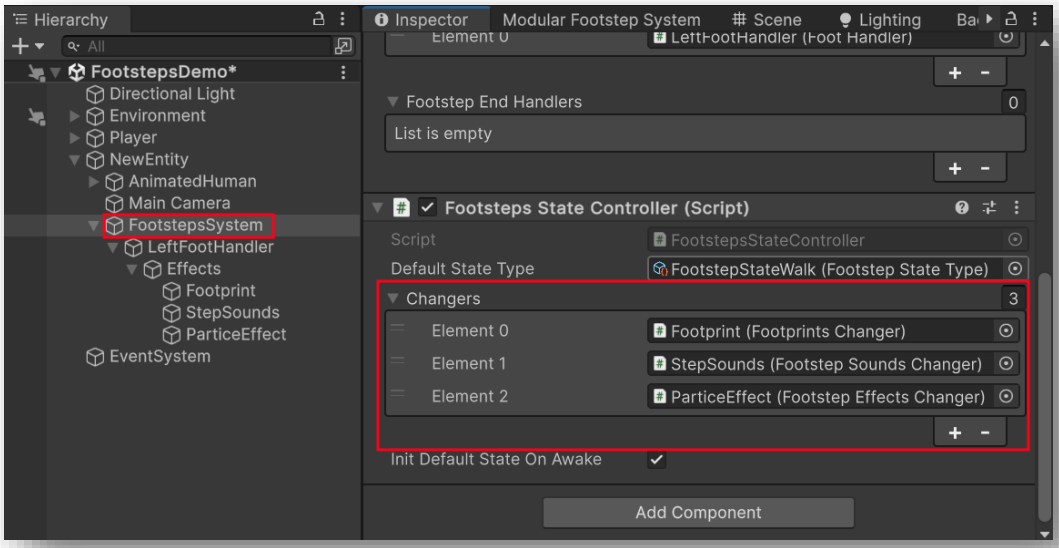
2. **Add Changers to Modules**
    - For the **Footprints Module**, add the **FootprintsChanger** component.
    - For the **Footstep Sounds Module**, add the **FootstepSoundsChanger** component.
    - For the **Footstep Effects Module**, add the **FootstepEffectsChanger** component.
3. **Link Changers to Setters**
    - In the **Data Setter** field of each changer, assign the corresponding setter component.
    - Example:
        - Link the **FootprintDataSetter** to the **FootprintsChanger**.
        - Link the **FootstepSoundSetter** to the **FootstepSoundsChanger**.
        - Link the **FootstepParticleEffectSetter** to the **FootstepEffectsChanger**.
4. **Configure State Data**
    - Populate the **State Data** list in each changer with entries for different states.
    - Each entry should include:
        - A **Footstep State Type** representing the state.
        - Settings for the effects (decals, sounds, or particles) to be used for that state.


Hierarchy
Inspector
Modular Footstep System
Scene
Lighting
FootstepsDemo*
Directional Light
Environment
Player
NewEntity
AnimatedHuman
Main Camera
FootstepsSystem
LeftFootHandler
Effects
Footprint
StepSounds
ParticeEffect
EventSystem
Extended Behaviour Pool (Script)
Script
ExtendedBehaviourPool
Prefab
FootprintDecal
Default Capacity
10
Max Size
100
Is Reused Active Objects
Footprints Changer (Script)
FootprintsChanger
Data Setter
Footprint (Footprint Data Setter)
State Data
2
Element 0
Footstep State Type
FootstepStateWalk (Footstep State Type)
Data
2
Element 0
FootprintOnSnowLeftFoot (Footprint Sur
Element 1
FootprintOnSandLeftFoot (Footprint Surf
Element 1
Footstep State Type
FootstepStateRun (Footstep State Type)
Data
0
List is empty
Add Component


Hierarchy
Inspector
Modular Footstep System
Scene
Lighting
FootstepsDemo*
Directional Light
Environment
Player
NewEntity
AnimatedHuman
Main Camera
FootstepsSystem
LeftFootHandler
Effects
Footprint
StepSounds
ParticeEffect
EventSystem
Element 0
LeftFootSnowWalkSoundData (Footstep S
Element 1
LeftFootSandWalkSoundData (Footstep S
Element 2
LeftFootBrickWalkSoundData (Footstep S
Footstep Sound Creator
StepSounds (Footstep Sound Creator)
Footstep Sounds Changer (Script)
FootstepSoundsChanger
Data Setter
StepSounds (Footstep Sound Setter)
State Data
2
Element 0
Footstep State Type
FootstepStateWalk (Footstep State Type)
Data
3
Element 0
LeftFootSnowWalkSoundData (Footstep
Element 1
LeftFootSandWalkSoundData (Footstep
Element 2
LeftFootBrickWalkSoundData (Footstep
Element 1
Footstep State Type
FootstepStateRun (Footstep State Type)
Data
1
Element 0
LeftFootSnowRunSoundData (Footstep
Add Component


Hierarchy
Inspector
Modular Footstep System
Scene
Lighting
FootstepsDemo*
Directional Light
Environment
Player
NewEntity
AnimatedHuman
Main Camera
FootstepsSystem
LeftFootHandler
Effects
Footprint
StepSounds
ParticeEffect
EventSystem
Element 0
FootstepSandEffectData (Footstep Particl
Element 1
FootstepSnowEffectData (Footstep Partic
Footstep Particle Effect Creator
ParticeEffect (Footstep Particle Effect Cree
Effects Pool Default Capacity
10
Effects Pool Max Capacity
20
Footstep Effects Changer (Script)
FootstepEffectsChanger
Data Setter
ParticeEffect (Footstep Particle Effect Sett
State Data
2
Element 0
Footstep State Type
FootstepStateWalk (Footstep State Type)
Data
2
Element 0
FootstepSnowEffectData (Footstep Parti
Element 1
FootstepSandEffectData (Footstep Partic
Element 1
Footstep State Type
FootstepStateRun (Footstep State Type)
Data
0
List is empty
Add Component

5. **Add Changers to FootstepsStateController**
    - Add all changers to the **Changers** list in the **FootstepsStateController**.



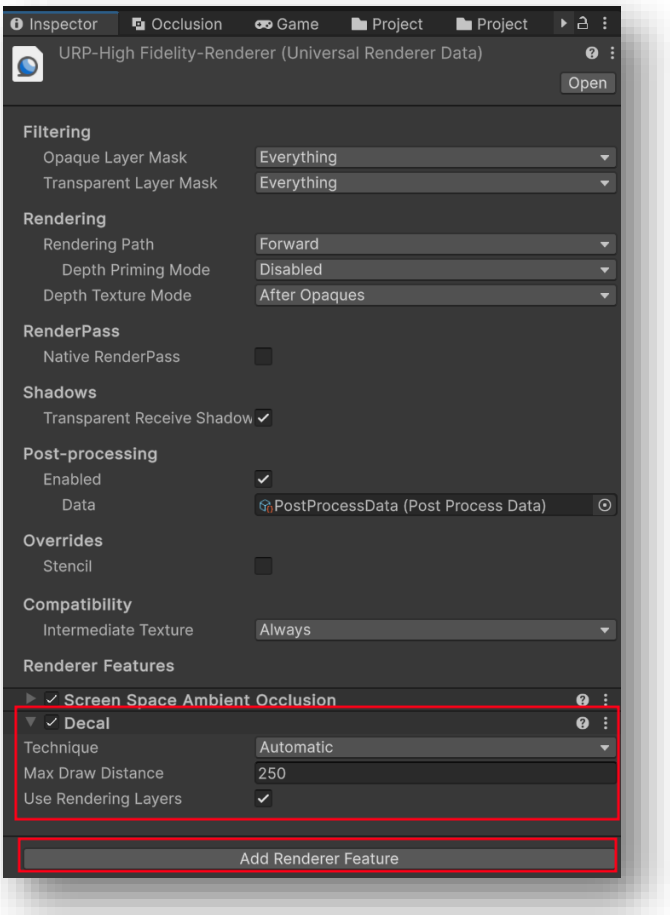
FAQ

1. Why aren't decals appearing?

- **Possible issue with the decal material:**
  - For **URP** and **HDRP**, textures used by the material must have a transparent background.
  - For **Built-in**, textures must have a black background if you are using the plugin's shader.
- **Rendering layer mismatch:**
  - In **HDRP** and **URP**, check the rendering layers (**Rendering Layers** or **Rendering Layers Mask**) in the **DecalProjector** component of the decal object and the **MeshRenderer** component of the objects where decals should appear.
- **Incorrect DecalProjector settings:**
  - Verify parameters such as **Opacity**, **Draw Distance**, and **Start Fade**.
  - Check other settings in the **DecalProjector** component that might vary depending on your **Render Pipeline Compatibility**.

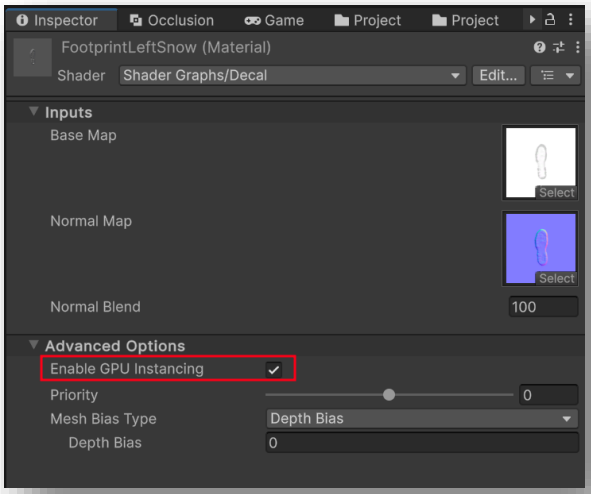
2. Why is the DecalProjector component missing in URP?

- Add the **Decal** feature to the Universal Renderer Data configuration in your graphics settings.



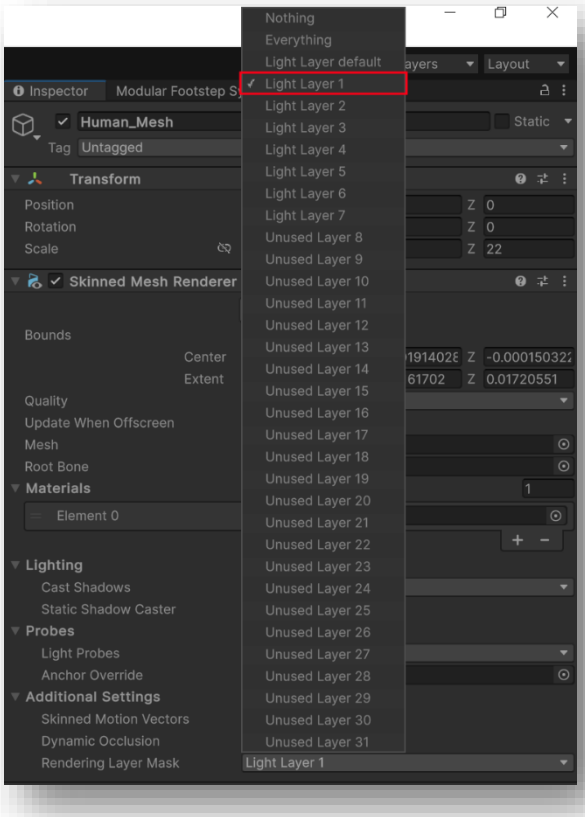
3. Why is the batch count high with many footprints?

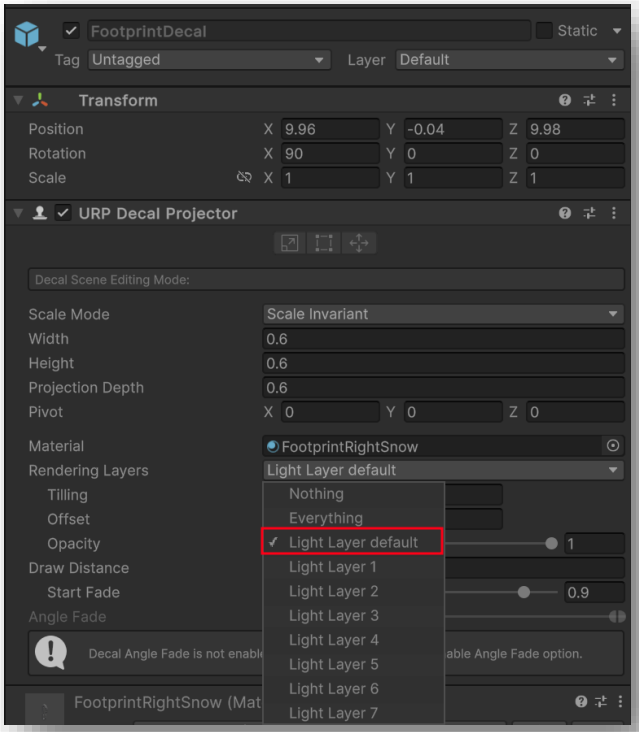
- Instances with decal materials are rendered individually, which increases draw calls. Enable **Enable GPU Instancing** in your decal materials to optimize performance.



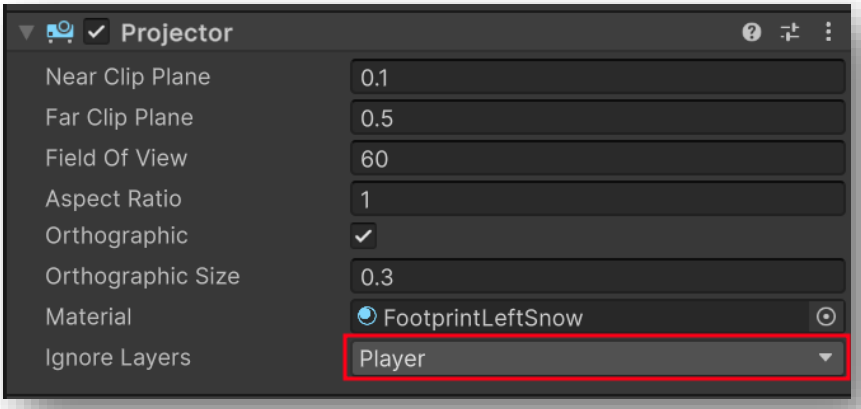
4. Why do footprints appear on the entity itself?

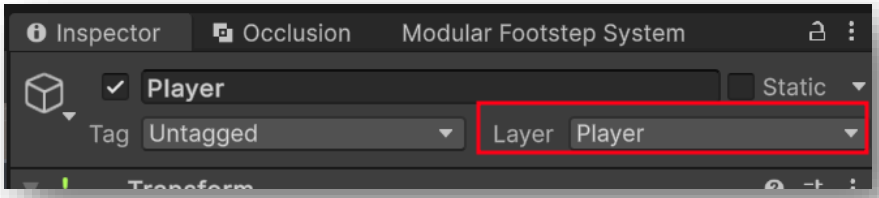
- This happens due to overlapping rendering layers between the **DecalProjector** and **MeshRenderer** components.
- Adjust the **MeshRenderer** settings on the entity to exclude layers shared with the **DecalProjector**.





- For **Built-in**, in the **Projector** component, you need to configure the **Ignore Layers** parameter by adding the layer used by the player.





Afterword

The implemented system is indeed not simple to configure. However, this complexity is compensated by its modularity, which allows adding only the required modules. It also enables parameter customization for modules based on the entity's state type.

The recommendations for manually adding components are merely suggestions. You can place and configure components wherever it is convenient for your project.

We hope you will not be disappointed with this system and will not encounter any problems during its setup and usage.

For any questions, issues, or suggestions regarding this plugin, please send your emails to **chepgamesystemsfeedback@gmail.com**