Footstepper

A Lightweight footstep audio system with terrain and mesh support

Version 1.0.0 • Unity 2019+

Table of Contents

System Overview

Footstepper provides automatic surface-based footstep audio with support for both Unity Terrain and mesh-based surfaces. The system detects surface materials through texture mapping and plays appropriate footstep sounds with configurable audio parameters.

Key Features

Surface Detection
Automatic detection of terrain textures and mesh materials using raycasting

Texture Mapping
Map specific audio clips to textures with individual volume and pitch settings

Dual Trigger Modes
Support for both distance-based and animation event-based triggering

Audio Randomization
Multiple audio clips per surface with pitch variation and volume control

Visual Debugging
Comprehensive gizmo system for setup and troubleshooting

Performance Optimized
Efficient raycasting with configurable layer masks and distance limits

Component Reference

FootstepController

The main controller component responsible for detecting surfaces and playing footstep audio. Attach this component to your character GameObject.

Parameter	Type	Default	Description
`footstepData`	FootstepsDatabase	null	Reference to the footstep database ScriptableObject
`audioSource`	AudioSource	null	Audio source component for playback (auto-created if null)
`raycastOrigin`	Transform	this.transform	Origin point for ground detection raycast
`raycastDistance`	float	2.0f	Maximum distance for ground detection
`raycastOffset`	float	0.1f	Vertical offset above ground for raycast origin
`triggerMode`	FootstepTriggerMode	Distance	Triggering method: Distance or AnimationEvent
`enableDebug`	bool	false	Enable console logging for debugging

Public Methods

```
public void OnFootstep() // Called by animation events or external systems to trigger footstep
```

FootstepsDatabase

A ScriptableObject that stores footstep configurations and texture-to-sound mappings.

Global Settings

Parameter	Type	Default	Description
`stepDistance`	float	1.5f	Distance between footsteps in distance mode
`defaultVolume`	float	0.7f	Base volume for all footstep sounds
`groundLayerMask`	LayerMask	-1	Layer mask for ground detection

TextureFootstepPair Structure

Field	Type	Description
`name`	string	Custom identifier for the footstep pair
`texture`	Texture2D	The texture to match for this footstep
`footstepSounds`	AudioClip[]	Array of audio clips for this surface
`volumeMultiplier`	float	Volume adjustment for this surface (default: 1.0)
`pitchVariation`	float	Random pitch variation range (default: 0.1)
`terrainLayerIndex`	int	Terrain layer index (-1 for mesh textures)

AnimationEventProxy

A utility component for routing animation events to FootstepController methods.

Parameter	Type	Description
`events`	UserNamedUnityEvent>	List of named events that can be triggered

Setup Guide

Step 1: Create FootstepsDatabase

1. Navigate to `Tools → Hyzen Works → FootstepsDatabase`
2. Click on the ➕ button to create a new footstep data and save it
3. Configure global settings (step distance, default volume, ground layer mask)

Step 2: Configure Texture Mappings

1. In the Footstepper Window, expand the `Texture FootSteps` list
2. Add entries for each surface type you want to support
3. For **mesh surfaces**: Assign the texture, leave `terrainLayerIndex` as -1
4. For **terrain surfaces**: Set `terrainLayerIndex` to match your terrain layer order (0, 1, 2, etc.)
5. Add multiple audio clips to each entry for variation
6. Adjust volume multipliers and pitch variation as needed

Important: For terrain footsteps, ensure the `terrainLayerIndex` values match your terrain's layer order exactly. Layer 0 is the first terrain texture, layer 1 is the second, etc.

Step 3: Setup Character Controller

1. Add the `FootstepController` component to your character GameObject
2. Assign your `FootstepsDatabase` to the `footstepData` field
3. Set `raycastOrigin` to a Transform at your character's feet level
4. Configure `raycastDistance` and `raycastOffset` as needed
5. Choose your preferred `triggerMode`
6. Enable `enableDebug` for initial testing

Step 4: Animation Event Setup (Optional)

If using `AnimationEvent` trigger mode:

1. Add the `AnimationEventProxy` component to your character
2. Create a new named event (e.g., "FootstepLeft", "FootstepRight")
3. Connect the event to the FootstepController's `OnFootstep()` method
4. In your animation model's import settings, go to the Animations tab
5. Select your walk/run animation and add Animation Events
6. Set the Function to "TriggerEvent" and String Parameter to your event name
7. Position the events at appropriate times in your animation

Animation Events: You can add animation events by selecting your model in the Project window, going to the Animations tab in the Inspector, and adding events in the Animation Event track.

Important: Make sure the you add the AnimationEventProxy component to the Player Model (the one with the animator and animations). Otherwise the Animation Events work.

Configuration

Trigger Modes

Distance Mode

Footsteps are triggered automatically based on distance traveled.

- **Advantages:** Works with any movement system, no animation setup required
- **Disadvantages:** May not sync perfectly with character animation
- **Best for:** Simple movement systems, prototyping

Animation Event Mode

Footsteps are triggered by animation events.

- **Advantages:** Perfect synchronization with character animation
- **Disadvantages:** Requires animation event setup
- **Best for:** Polished character systems, complex animations

Surface Detection

Terrain Detection

The system automatically detects the dominant terrain texture at the character's position using Unity's terrain alphamap system.

```
// Terrain layer indices correspond to the order of textures in your terrain // Layer 0 = First terrain texture // Layer 1 = Second terrain texture // etc.
```

Mesh Detection

For mesh surfaces, the system uses the main texture from the material of the hit surface.

```
// The system checks: material.mainTexture // Ensure your materials have the correct texture assigned as the main texture
```

Audio Configuration

Volume Control

- **Global Volume:** Set via `defaultVolume` in the database
- **Per-Surface Volume:** Use `volumeMultiplier` in each TextureFootstepPair
- **Final Volume:** `defaultVolume * volumeMultiplier`

Pitch Variation

- Controlled by `pitchVariation` in each TextureFootstepPair
- Applied as: `basePitch + Random.Range(-pitchVariation, pitchVariation)`
- Recommended range: 0.1 to 0.2 for natural variation

Debugging

Console Logging

Enable `enableDebug` on the FootstepController to see detailed console output:

```
Playing terrain footstep for layer 2 Playing mesh footstep for texture GrassTile No footstep found for texture RockTile, using default No ground detected for footstep
```

Visual Gizmos

The FootstepController provides comprehensive visual feedback in the Scene view:

Gizmo	Color	Description
Raycast Line	Green/Red	Green when ground detected, red when no ground
Origin Sphere	Yellow	Raycast origin point
Step Distance Circle	Blue	Maximum step distance (distance mode only)
Progress Circle	Blue to Cyan	Current progress toward next step
Last Position Cube	Magenta	Position of last triggered footstep
Ground Hit Sphere	Green	Exact point where raycast hit the ground
Surface Normal	Green	Direction of surface normal at hit point

Common Issues

No Footstep Sounds

- Check that `footstepData` is assigned
- Verify `groundLayerMask` includes the ground layer
- Ensure `raycastDistance` is sufficient to reach the ground
- Check that audio clips are assigned in the database

Wrong Surface Detected

- For terrain: Verify `terrainLayerIndex` matches your terrain layer order
- For meshes: Check that the material's main texture is correctly assigned
- Use debug mode to see which textures are being detected

Animation Events Not Working

- Ensure `AnimationEventProxy` is attached to the same GameObject as the Animator
- Check that animation event function is set to "TriggerEvent"
- Verify string parameter matches your event name exactly
- Confirm the animation clip has proper event timing

Best Practices

Performance Optimization

- Use specific layer masks instead of checking all layers
- Set appropriate raycast distances - avoid unnecessarily large values

Audio Quality

- Use 3-5 different audio clips per surface for good variation
- Keep pitch variation between 0.1-0.2 for natural sound
- Balance volume multipliers so all surfaces have similar perceived loudness
- Use high-quality audio files with consistent levels

Setup Tips

- Position raycast origin at the character's feet or slightly above
- Use a small raycast offset (0.1-0.2) to avoid ground clipping issues
- Test with debug mode enabled before deployment
- Create a dedicated layer for ground objects

Content Organization

- Use descriptive names for TextureFootstepPair entries
- Group similar surfaces together in the database
- Maintain consistent naming conventions for audio files
- Document terrain layer indices for team members

Pro Tip: Use the visual gizmos to fine-tune your raycast settings. The step distance visualization helps you understand how the distance-based triggering works.