

AudioServer in Godot – Complete Guide

 gamedevacademy.org/audioserver-in-godot-complete-guide

November 30, 2023

Audio is the heartbeat of any game. It can turn a simple gameplay experience into an immersive adventure that captures the imagination and emotions of the player. In this tutorial, we're diving deep into the world of sound with Godot 4's AudioServer class, an essential tool for any game developer looking to add depth and character to their game through audio.

Table of contents

- [What is AudioServer?](#)
- [What is AudioServer for?](#)
- [Why Should I Learn It?](#)
- [Creating and Managing Audio Buses](#)
- [Playing Sounds on Audio Buses](#)
- [Adjusting Audio Playback](#)
- [Applying Effects to Audio Buses](#)
- [Where to Go Next in Your Godot Journey](#)
- [Conclusion](#)

What is AudioServer?

The AudioServer class in [Godot 4](#) represents the low-level server responsible for audio operations within your games or applications. It's where the magic happens for sound creation, management, and performance. At its core, AudioServer manages audio samples and their playback, oversees audio buses (paths through which audio is processed and routed), and interacts with effects to shape the final sound output.

What is AudioServer for?

AudioServer is equipped to handle different aspects of sound in a game. Whether it's playing music tracks, sound effects, or recording audio, AudioServer serves as the backbone for all things audio in [Godot](#). It allows developers to control sounds at a granular level, managing everything from playback speeds to environmental effects. The class offers functionalities such as adding or configuring audio buses, enabling or disabling effects, and even selecting the output device for sound playback.

Why Should I Learn It?

Learning AudioServer opens up a world of possibilities to enhance your game's ambience and mood. By mastering audio manipulation, you can craft custom soundscapes that bring your game's world to life and wow your players. Aside from its capacity to enchant, knowledge of AudioServer is a valuable addition to any developer's toolkit, providing the foundation necessary for high-level audio operations and a deeper understanding of how games leverage sound to deliver memorable experiences. It's a skill that can significantly impact game quality and player engagement.



Creating and Managing Audio Buses

To begin, we'll need to set up audio buses. An audio bus is essentially a channel through which audio is sent and can have various effects applied. Godot allows for multiple audio buses to create sophisticated audio systems within your game. Here's how you create and manage audio buses using AudioServer:

```
// Create a new audio bus
AudioServer.add_bus(index)
// You can set the name of the bus for easy reference
AudioServer.set_bus_name(index, "Effects")
// To change the volume of the bus, use the following:
AudioServer.set_bus_volume_db(index, volume_db)
// Mute or unmute an audio bus with:
AudioServer.set_bus_mute(index, true) // Mute
AudioServer.set_bus_mute(index, false) // Unmute
// You can also bypass effects with:
AudioServer.set_bus_bypass_effects(index, true) // Bypass
```

With these commands, you can start shaping the way audio is played in your game.

Playing Sounds on Audio Buses

Once your buses are set up, you're ready to play some sounds. Every sound in Godot is tied to an audio stream, and we use `AudioStreamPlayer` nodes to play these sounds. Here is how you can connect these nodes to your buses and play a sound:

```
// Assuming you have an AudioStreamPlayer node named 'soundPlayer' and an audio file 'sound.wav'
var soundPlayer = AudioStreamPlayer.new()
soundPlayer.stream = load("res://sound.wav")
soundPlayer.bus = "Effects" // The name of your bus
add_child(soundPlayer)
soundPlayer.play()
```

Note that `'res://sound.wav'` is a placeholder for your audio resource. You'll need to replace it with the path to your actual audio file.

Adjusting Audio Playback

Next, let's see how to adjust different audio parameters during playback. Godot makes it straightforward to control aspects such as volume and pitch, which can add dynamic effects to your audio.

```
// Set the volume of an audio stream in decibels
soundPlayer.set_volume_db(volume_db)
// Change the pitch of the audio stream, where 1.0 is the normal pitch
soundPlayer.set_pitch_scale(pitch)
// You can also loop a sound by enabling the loop property
soundPlayer.loop = true
soundPlayer.play()
```

Through these manipulations, you can create interesting audio effects such as doppler effects, character voices, or machinery noises.

Applying Effects to Audio Buses

Now, let's apply some effects to an audio bus. Effects can add depth and space to your audio, simulating environments like concert halls or dungeons. Here's how to add a reverb effect to an audio bus in Godot:

```
// First, create an AudioEffectReverb instance
var reverb_effect = AudioEffectReverb.new()
// Setup properties for the reverb effect
reverb_effect.predelay_ms = 150.0
reverb_effect.room_size = 100.0
```

```
reverb_effect.damp = 0.5
reverb_effect.wet = 1.0
// Add the effect to the bus (index 0)
AudioServer.add_bus_effect(0, reverb_effect)
// Optionally, you can set the effect to be enabled or disabled
AudioServer.set_bus_effect_enabled(0, effect_index, true) // Enable
```

With this, the audio bus will now have a reverb effect applied to all audio that is routed through it. Experiment with different effects to create the perfect sound for your game.

Stay tuned for the next part where we'll continue with more advanced audio functionalities and how to create a fully featured audio system within your Godot game!

Moving deeper into the capabilities of Godot's AudioServer, we will now explore recording audio at runtime, effective sound management through [coding](#), and how to interact with the devices available for audio playback. These techniques empower developers to create dynamic and responsive audio environments for players.

Recording Audio with AudioServer

To record audio using Godot's AudioServer, follow these steps:

```
// Start recording to a buffer (44.1 kHz and stereo)
var format = AudioStreamSample.Format_16Bits
AudioServer.start_recording_stream(format, 44100, true)
// Stop recording and obtain the recorded audio stream
var recorded_stream = AudioServer.get_recording_stream()
// You can then save this stream or play it back using an AudioStreamPlayer
```

Remember, you must handle permissions for recording on each platform accordingly.

Managing Sound Playback

Effective sound management often requires stopping or pausing sounds based on in-game events. Here's how you can control the sound playback:

```
// To stop all sounds on a specific bus
AudioServer.stop_bus(index)
// To pause all sounds on a bus
AudioServer.set_bus_pause(index, true)
// To resume sounds on a paused bus
AudioServer.set_bus_pause(index, false)
```

These commands let you respond to in-game events, such as pausing the game, by pausing all sound effects or music appropriately.

Altering Audio Bus Layouts

To further modify how audio is routed in your game, you may want to add, remove, or move audio buses dynamically:

```
// To move an audio bus to another index position
AudioServer.move_bus(old_index, new_index)
// If you need to remove an audio bus, you can do so with:
AudioServer.remove_bus(index)
// When your sound system requires a completely new bus layout:
AudioServer.swap_bus(index1, index2)
```

These commands allow for dynamic audio routing that can be adjusted based on gameplay mechanics or player preferences.

Interacting with Audio Playback Devices

Players may have specific audio output preferences or devices. Here's how you can list and set playback devices using Godot's AudioServer:

```
// List available playback devices
var devices = AudioServer.get_device_list()
// You can set the desired device for playback
AudioServer.set_device("desired_device_name")
```

To provide players with the best experience, consider integrating an options menu that includes the choice of audio output devices.

These advanced techniques offer a glimpse into the power and flexibility of Godot's audio system. By mastering the AudioServer class, you can offer your players an audio experience that greatly enhances immersion and engagement in your game. As with any system, it is important to test thoroughly to ensure that your audio behaves as expected across different scenarios and platforms.

Whether it's creating dynamic soundscapes, reacting to in-game triggers, or providing user choices for audio output, the knowledge you've gained here will serve as a strong foundation for all your future Godot projects. We're passionate about equipping you with the skills you need to bring your creative visions to life, and we hope this guide has been a valuable step on your development journey.

Taking the audio experience further, we can tap into the spatial and environment aspects of sound within Godot's 3D space. By implementing 3D sound positioning and environmental effects, we can create a truly immersive audio environment that breathes life into the game world. Let's explore some code examples of how to work with 3D sounds and environmental effects in Godot.

Positioning 3D Sounds

For 3D games, positioning audio in a three-dimensional space is crucial to add realism. Godot's `AudioStreamPlayer3D` node is designed for this purpose:

```
// Assuming you have an AudioStreamPlayer3D node with an attached audio file
var soundPlayer3D = AudioStreamPlayer3D.new()
soundPlayer3D.stream = load("res://3D_sound.ogg")
// Set the position of the sound in 3D space
soundPlayer3D.translation = Vector3(10, 0, 5)
// Add the node to the scene and play the sound
add_child(soundPlayer3D)
soundPlayer3D.play()
```

By modifying the 'translation' property, you can position your sound in any location within the 3D space of your game world.

Using Doppler Effect in 3D

The Doppler effect can add to the realism of moving objects by altering the perceived frequency of their sound. To implement Doppler effects:

```
// Set up Doppler tracking on an AudioStreamPlayer3D node
soundPlayer3D.doppler_tracking =
AudioStreamPlayer3D.DOPPLER_TRACKING_PHYSICS_STEP
// Adjust the Doppler scale to increase or decrease the effect
soundPlayer3D.doppler_scale = 2.0
```

This effect will automatically be applied based on the velocities of the object and the listener within the 3D space.

Environmental Reverb in 3D Spaces

To create an environmental reverb effect that simulates the acoustics of a physical space:

```
// Create an Area node with ReverbBus effect applied
var area = Area.new()
var reverb_bus = AudioEffectReverbBus.new()
reverb_bus.bus = "Reverb" // A previously created audio bus with reverb effect applied
area.add_reverb_bus(reverb_bus)
// Tailor the reverb properties to the desired space
reverb_bus.uniformity = 0.5 // Adjust between 0.0 (non-uniform) to 1.0 (uniform)
reverb_bus.room_size = 0.8
```

Players within this Area node will experience the sound as if they're inside the defined acoustic environment.

Controlling the Listener

By default, Godot uses the camera as the listener for 3D sounds. If you want to change this behavior or adjust the listener's properties:

```
// Assuming you have a Spatial node named 'customListener'  
var customListener = Spatial.new()  
customListener.translation = Vector3(5, 1, 3)  
// Set your custom listener  
AudioServer.set_listener_transform(customListener.global_transform)  
// If you have multiple listeners, you can set the current one by index  
AudioServer.set_current_listener(1) // Listener index
```

Custom listeners allow for unique gameplay mechanics, such as hearing from the perspective of multiple characters.

Managing Audio Threads

To ensure that audio processing does not hinder your game's performance, Godot allows you to manage the threading mode:

```
// Set the audio thread mode for performance adjustments  
AudioServer.set_thread_safe_mode(AudioServer.THREAD_SAFE_MODE_SAFE)  
// You can check the current threading mode with  
var thread_mode = AudioServer.get_thread_safe_mode()
```

These are advanced settings that can be crucial for optimizing games with intensive audio processing.

With these examples and the ability to script complex audio behavior, Godot's audio system becomes a powerful tool in your development arsenal. The combination of spatial audio, Doppler effects, environmental [simulation](#), and performance management can contribute to creating a more authentic and engaging game world. Remember to experiment with different settings and properties to find the perfect audio balance that supports your game's design and narrative.

Where to Go Next in Your Godot Journey

Congratulations on taking your Godot skills to new heights! The knowledge you've gained about the AudioServer is a game-changer for creating immersive audio experiences in your projects. But your journey doesn't have to stop here. At Zenva, we believe in continuous learning and growing through practical experience, and we've got just the resources you need.

To further advance your game development skills with Godot, we eagerly recommend our [Godot Game Development Mini-Degree](#). This comprehensive collection of courses will guide you through building your own games using the Godot 4 engine. Dive into topics including GDScript [programming](#) language, UI systems, and mechanics for various game genres.

Tailored for beginners and providing flexible, project-based learning, this Mini-Degree will equip you with the skills to create a professional portfolio and take a confident stride towards game development mastery.

For those who want to explore the breadth of content we offer, visit our full selection of [Godot courses](#). These courses, available 24/7, cover a spectrum of topics, ensuring there's always something new to discover. Continue learning at your own pace, and create the games you've always dreamed of with Zenva.

Conclusion

As we wrap up this tutorial, you now hold the key to unlocking the full potential of Godot's audio capabilities. Whether you're a hobbyist [game developer](#) or striving to become a professional, the skills you've acquired with the AudioServer class are pivotal for breathing life into your virtual worlds. We hope you feel inspired to experiment with the dynamic audio landscapes you can create, and remember, the soundscape of your game can be as vast and varied as your imagination allows it to be.

Don't let the learning end here! With our [Godot Game Development Mini-Degree](#), you have the opportunity to expand your horizons and refine your craft even further. We at Zenva are excited to be a part of your journey and can't wait to see the amazing games you'll create with the power of Godot's audio tools. Happy game developing!

FREE COURSES



Unlock coding courses in Unity, Unreal, Python, Godot and more.