Audio Management Toolkit

Developer guide

Updated on November 8, 2025.
Written by Juan Borges Vega.

Introduction

This Audio Management Toolkit provides a solution that offers a modular and professional framework for handling the most common of your project's audio needs. From background music to sound effects, this toolkit provides a great solution for most of your common needs.

Features

The Audio Management Toolkit offers a wide range of features designed to streamline your workflow:

1. **Modular Audio Managers:** Easily configure and manage audio sources through a centralized system.
2. **Playback Control:** Play, pause, stop, and fade audio clips.
3. **Event-Driven System:** Integrate audio events with your game logic using a clean, event-based architecture.

Single Audio Management

The Single Audio Management System provides a structured way to handle audio that must remain exclusive such as background music, ambient loops, or voice lines where one clip should play at a time. The toolkit is composed of four main modules described in Table 1:

Module	Description
AudioSourcePlayer	Encapsulates playback logic for a Unity AudioSource. Handles crossfades, fade in/out, and playback notifications.
SingleAudioManager	Manages a single *AudioSourcePlayer*, acting as a controller that responds to event requests.
SingleAudioActionEventChannel	A ScriptableObject that defines and raises audio-related events, serving as a communication bridge between systems.
SingleAudioActionRequester	Triggers playback requests by raising events through the event channel, allowing external scripts to control audio indirectly.

Table 1. Single audio management modules.

System Architecture

Figure 1 illustrates the structure of this subsystem. The SingleAudioManager composes an AudioSourcePlayer, which in turn owns a Unity AudioSource component responsible for actual playback. Both the manager and the SingleAudioRequester maintain associations with a shared SingleAudioActionEventChannel; a ScriptableObject that defines and broadcasts audio events.

The requester raises playback, pause, or stop events, while the manager listens for those events and delegates the appropriate actions to its AudioSourcePlayer. This design ensures high cohesion within each module and loose coupling between systems.

Although SingleAudioActionRequester serves as an example implementation to demonstrate how another script can trigger audio actions through the event channel, it can also be extended or used as base class for more specific gameplay use cases.

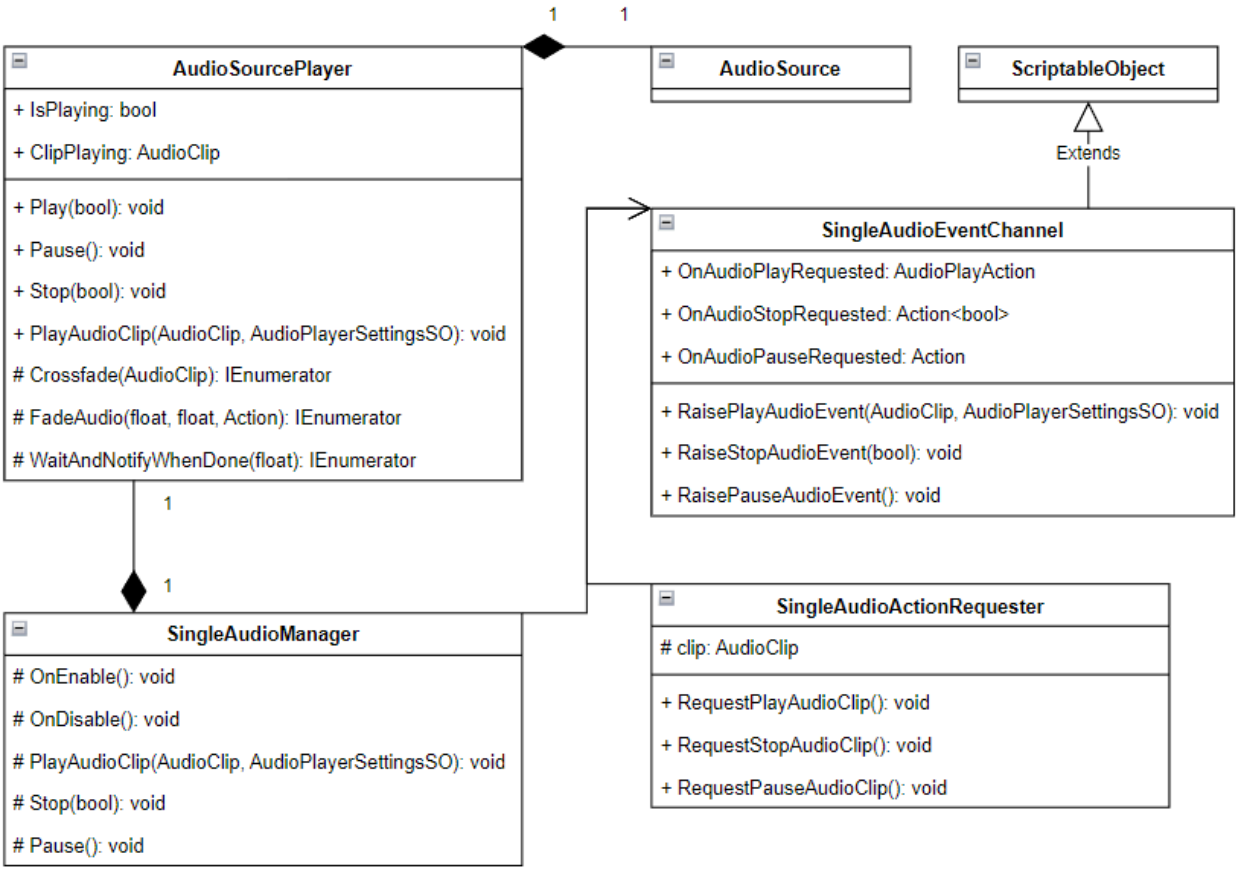

Figure 1. Single Audio Management UML Class Diagram.

Workflow

To complement how these modules interact, the following workflow outlines generally the sequence of events and method calls that occur when audio playback is requested and managed:

1. **Initialization:** SingleAudioManagement component is configured in the scene by referencing an AudioSourcePlayer and SingleAudioEventChannel.
2. **Event Subscription:** When enabled, the manager subscribes to the event channel's play, pause and stop events.
3. **Requesting Audio Actions:** Any game object or system can reference the SingleAudioEventChannel that a desired SingleAudioManager uses.
4. **Event Handling and playback:** SingleAudioManager receives the event callbacks and internally calls its methods. These methods delegate to the AudioSourcePlayer, which executes the actual playback, fade or transition behavior.

Group Audio Management

The Group Audio Management System provides a flexible and efficient way to handle multiple simultaneous audio sources, making it ideal for managing sound effects (SFX) or any audio that can overlap during gameplay. Unlike the Single Audio Management System, which focuses on exclusive playback, this system leverages a pool of audio players to manage concurrent sounds dynamically while maintaining performance.

The toolkit is composed of six main modules described in Table 2:

Module	Description
AudioSourcePlayer	Encapsulates playback logic for a Unity AudioSource. Handles crossfades, fade in/out, and playback notifications.
GroupAudioManager	Oversees a pool of AudioSourcePlayer instances, managing their
AudioSourcePlayerContainer	Stores references between unique integer IDs and their corresponding AudioSourcePlayer instances for playback control.
AudioSourcePlayerPoolSO	A ScriptableObject that defines and prewarms a pool of reusable audio players.
GroupAudioActionEventChannel	A ScriptableObject that defines and raises audio-related events, serving as a communication bridge between systems.
GroupAudioActionRequester	Sends playback, stop, and pause requests through the event channel, using IDs to target specific sound instances.

Table 2. Single audio management modules.

System Architecture

Figure 2 illustrates the main structure of the Group Audio Management system, which is designed to handle multiple simultaneous audio sources. Unlike the Single Audio Management system, this maintains and controls a pool of AudioSourcePlayer instances rather than a single one.

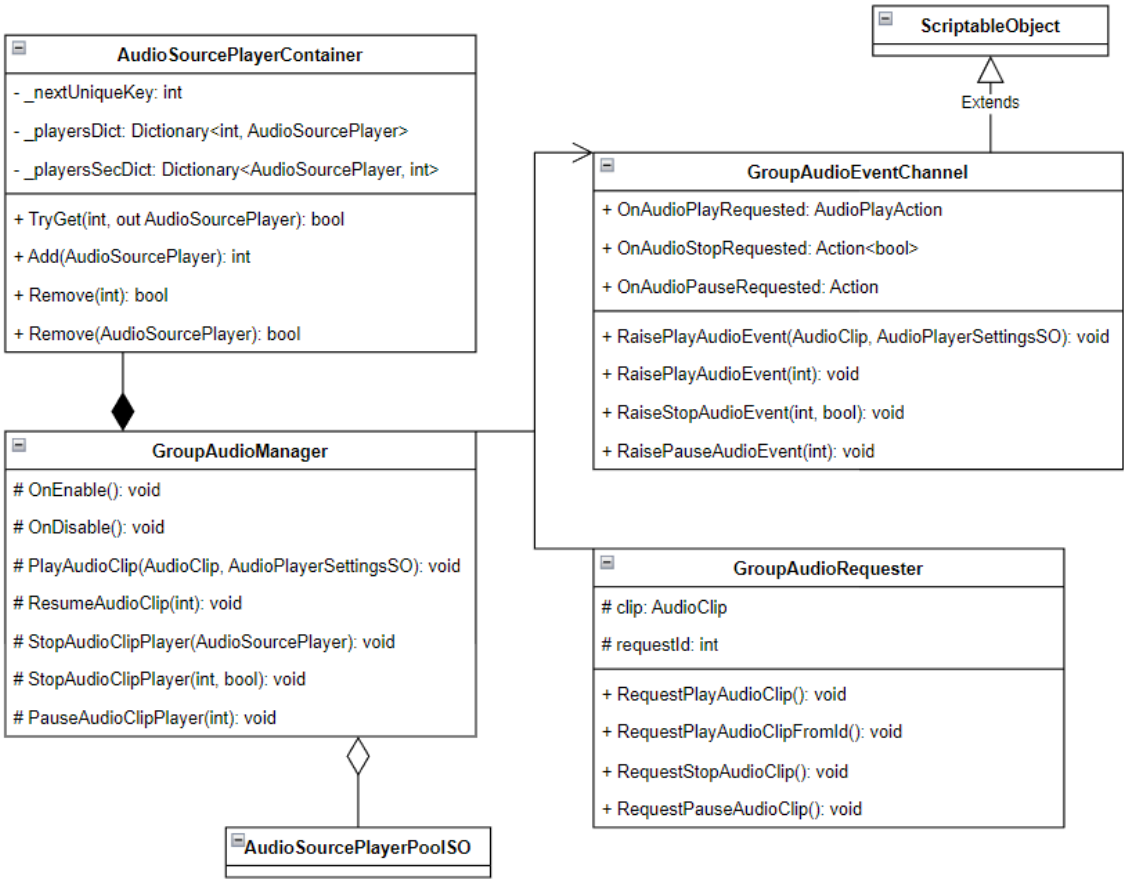

Figure 2. Group Audio Management UML Class Diagram.

The GroupAudioManager tracks all AudioSourcePlayer instances that are being used through AudioSourcePlayerContainer. This container holds unique integer IDs for each AudioSourcePlayer, allowing it to reference specific playback instances. The manager also maintains a reference to an AudioSourcePlayerPoolSO, which defines the pool of reusable audio players.

Both the GroupAudioManager and the GroupAudioRequester interact with a shared GroupAudioEventChannel. The event channel supports both global events.

Although GroupAudioActionRequester serves as an example implementation to demonstrate how other scripts can trigger audio actions through the event channel, it can also be extended or used as a base class for more specialized gameplay scenarios where multiple sound effects or instances need to be controlled individually.

Workflow

To complement how these modules interact, the following workflow outlines generally the sequence of events and method calls that occur when multiple audio clips are requested, played, and managed through the pooling system:

1. **Initialization:** GroupAudioManagement component is added to the scene and references the AudioSourcePlayerPoolSO and GroupAudioEventChannel.

2. **Event Subscription:** When enabled, the manager subscribes to the event channel's play, resume, pause and stop events.
3. **Requesting Audio Actions:** Any game object or system can reference the GroupAudioEventChannel that a desired GroupAudioManager uses.
5. **Event Handling and playback:** GroupAudioManager receives the event callbacks and internally calls its methods. These methods delegate to an available AudioSourcePlayer to use given by the AudioSourcePlayerPoolSO, which executes the actual playback, fade or transition behavior, and returns the AudioSourcePlayer back to the pool once it's done playing the Audio Clip.

Demo

The Audio Management demo scene showcases the use of both the Single Audio Management and Group Audio Management systems. In this example, the Single Audio Management module handles Music playback, while the Group Audio Management module manages SFX (sound effects).

The user interface is divided into two sections, Music and SFX, each illustrating different use cases of the toolkit (see Figure 3).

- **Music Section:** Users can select between two songs (Jazz and Ukulele) to play. Below the song selection, standard playback controls (Play, Stop, and Pause) with the active track.
- **SFX Section:** The SFX panel displays four buttons representing different sound effects (Wolf, Thunder, Car, and Chime). Initially, playback control buttons are hidden. Once an SFX clip has been requested to play, the corresponding controls become visible, allowing users to manage each sound effect individually.

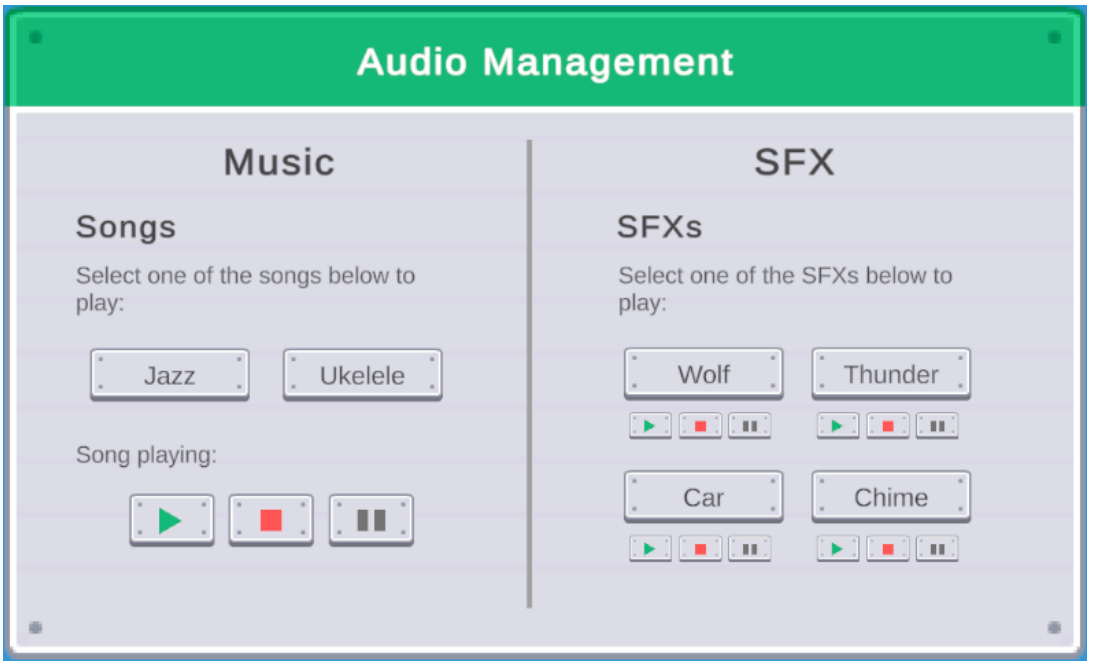

Figure 3. Audio Management Demo Scene with Music and SFX playback controls.