Snog's Audio Manager

Documentation

Table of Contents

What this is:

A lightweight, production-oriented audio framework for Unity that provides a central AudioManager, name-based clip libraries, pooled 3D SFX, a fully featured ambient system (profiles/stack/emitters/zones), gameplay triggers, and editor tooling (preview + auto-generation of audio assets).

1. Key Features

- **Centralized audio control** via AudioManager (singleton) with Music, Ambient, and FX routing.
- **Mixer volume control** mapped to dB and applied to AudioMixer exposed parameters.
- **Mixer snapshot transitions** via a **user-defined snapshot list**. Add any number of snapshots (Name + AudioMixerSnapshot reference) in the AudioManager inspector and transition by snapshot name at runtime.
- **SFX playback**
    - 2D OneShot SFX (PlaySfx2D) via a dedicated 2D FX AudioSource.
    - 3D spatial SFX (PlaySfx3D) via a pooled AudioSource system.
- **Music playback** with optional delay and fade-in/out.
- **Ambient system**
    - Ambient **profiles** made of layers (tracks + volumes).
    - A **stack** of profiles (push/pop tokens, replace/clear).
    - **World emitters** that are scored (priority + distance) and voice-capped.
    - **Ambient zones** to apply profiles by trigger volumes (Replace or Stack modes).
- **Gameplay trigger component** (AudioTrigger) to fire SFX/Music/Ambient actions and snapshot transitions when a tag enters/exits a trigger collider.
- **Editor tooling**
    - Custom inspectors with runtime testing and editor clip preview.
    - “Scan → Generate → Assign” pipeline to create ScriptableObject track assets and fill libraries automatically.

2. Requirements

- **Unity AudioMixer** workflow (AudioMixer + groups + snapshots). The manager routes sources to mixer groups and transitions snapshots.
- A Unity version that includes:
    - FindFirstObjectByType<T>() (used for listener discovery).
    - FindAnyObjectByType<T>() (used by editor inspectors).

If your Unity version does not have these APIs, replace them
with FindObjectOfType<T>() equivalents in the few call sites (runtime is otherwise unaffected).

3. Installation

1. Import the package into your Unity project.
2. Ensure the scripts compile (Editor scripts are guarded by #if UNITY_EDITOR).
3. Create an AudioManager GameObject and add required components (see next section).

4. Project Setup (Mixer + Manager)

4.1 Create / Configure the AudioMixer

Create an AudioMixer with (at minimum) these **exposed parameters**:

- MasterVolume
- MusicVolume
- AmbientVolume
- FXVolume

AudioManager.SetVolume() writes dB values into these parameters.

4.2 Create Groups

Create mixer groups to route the system's sources:

- Music group → assign to musicGroup
- Ambient group → assign to ambientGroup
- FX group → assign to fxGroup

4.3 Create Snapshots

1. Create any mixer snapshots in your AudioMixer (e.g., Default, Combat, Stealth, Underwater, Menu, LowHealth, etc.).
2. In AudioManager, add entries to Snapshots:
   - Name: the key you'll call at runtime (must be unique)
   - Snapshot: the AudioMixerSnapshot reference
3. At runtime, call TransitionToSnapshot("SnapshotName", transitionTime).

Important: Snapshot names should be **unique**. If two entries share the same name, the system will pick one (and your implementation should warn)

4.4 Create the AudioManager GameObject

Create a GameObject (e.g., Audio) and add:

- AudioManager (singleton)
- SoundLibrary
- MusicLibrary
- AmbientLibrary

AudioManager has [RequireComponent] attributes for all three libraries.

In the AudioManager inspector, assign:

- mainMixer
- musicGroup, ambientGroup, fxGroup
- snapshot references (if used)

On Awake, the manager creates internal 2D FX and Music AudioSources and applies routing.

4.5 3D FX Pool

- If fxPool is not assigned, AudioManager auto-creates an AudioSourcePool child GameObject (FX Pool) and initializes it with fxPoolSize and fxGroup.

5. Quick Start

5.1 Play SFX

AudioManager.Instance.PlaySfx2D("ButtonClick");

AudioManager.Instance.PlaySfx3D("Explosion", transform.position);

SFX names are resolved via SoundLibrary.GetClipFromName() and then played as 2D OneShot or via the 3D pool.

5.2 Play Music

AudioManager.Instance.PlayMusic("MainTheme", delay: 0f, fadeIn: 1.0f);

AudioManager.Instance.StopMusic(fadeOut: 1.0f);

Music names are resolved via MusicLibrary.GetClipFromName(). Fade-in/out is handled by coroutines inside AudioManager.

5.3 Ambient with Profiles + Zones

1. Create AmbientTrack assets.
2. Place AmbientEmitter components in the scene and assign each one an AmbientTrack.
3. Create an AmbientProfile asset and add layers referencing the same AmbientTrack assets.
4. Add AmbientZone triggers in your level and assign the AmbientProfile.

The manager uses a stack of profiles to decide which tracks are desired, scores emitters, and fades allowed emitters toward the target volume.

5.4 Use AudioTrigger

Add AudioTrigger to a trigger collider and configure:

- Tag to compare (default Player)
- Fire on Enter/Exit
- Audio Type (SFX/Music/Ambient)
- Action (e.g., Play2D, Play3D, PlayFadeIn, StopMusic, SnapshotCombat…)

6. Core Concepts

6.1 AudioManager

Responsibilities

- Holds mixer references and exposed volume sliders.
- Creates core sources:
    - FX 2D AudioSource (fx2DSource, 2D)
    - Music AudioSource (musicSource, 2D)
- Initializes / uses a pooled 3D FX system via AudioSourcePool.
- Runs the ambient loop coroutine that rescoring emitters at ambientRescoreInterval and stepping volumes every frame.

Public API

Volumes

- SetVolume(float volume01, AudioChannel channel) → converts volume01 into dB and writes to AudioMixer parameters.
- GetMixerVolumeDB(string parameterName) → reads a mixer parameter.

Snapshots

- TransitionToSnapshot(string snapshotName, float transitionTime)

SFX

- PlaySfx2D(string soundName) → uses fx2DSource.PlayOneShot() with scaled volume (fxVolume * masterVolume).
- PlaySfx3D(string soundName, Vector3 position) → uses fxPool.PlayClip() with scaled volume.

Music

- PlayMusic(string trackName, float delay = 0f, float fadeIn = 0f) → sets clip and optionally fades in.
- StopMusic(float fadeOut = 0f) → stops immediately or fades out then stops.

Ambient

- RegisterEmitter(AmbientEmitter emitter) / UnregisterEmitter(AmbientEmitter emitter)
- Stack operations:
    - SetAmbientProfile(AmbientProfile profile, float fade = -1f) (replace stack)
    - ClearAmbient(float fade = -1f)

- PushAmbientProfile(AmbientProfile profile, int priority = 0, float fade = -1f) returns a token.
- PopAmbientToken(int token, float fade = -1f)
- PopAmbientProfile(AmbientProfile profile, float fade = -1f)

Helpers

- TryGetSoundNames(out string[] names) / TryGetMusicNames(...) / TryGetAmbientNames(...) return sorted name lists from libraries.

6.2 Libraries (SFX/Music/Ambient)

Libraries provide **name → clip** resolution.

SoundLibrary

- Stores List<SoundClipData> tracks and optional InlineSoundData[] inlineSounds.
- Builds Dictionary<string, AudioClip[]> and returns a random clip variation for a given sound name.

MusicLibrary

- Stores List<MusicTrack> tracks and builds Dictionary<string, AudioClip>.

AmbientLibrary

- Stores List<AmbientTrack> tracks and builds Dictionary<string, AudioClip>.

All libraries provide

- AudioClip GetClipFromName(string name)
- string[] GetAllClipNames()
- Editor context menu: “Rebuild ... Dictionary”.

6.3 ScriptableObject Clip Assets

SoundClipData

- soundName + AudioClip[] clips (variants).

MusicTrack

- trackName, moodTag, clip, loop, description.

AmbientTrack

- trackName, moodTag, clip, description.

Note: In the current runtime implementation, MusicTrack.loop controls whether this track loops at runtime. If loop is enabled, the AudioManager sets the music AudioSource to loop for this track; if disabled, it plays once and stops

6.4 3D SFX Pool

AudioSourcePool provides pooled spatial AudioSources.

- Pre-builds poolSize sources and assigns them to fxGroup.
- Applies default spatial settings (spatialBlend=1, rolloff, min/max distance, doppler).
- PlayClip(AudioClip clip, Vector3 pos, float volume) plays the clip and returns the source to the pool once finished.

6.5 Ambient System

The ambient system is built from 4 pieces:

1. **AmbientTrack**: the clip data asset.
2. **AmbientEmitter**: a world object that can play an AmbientTrack and is controlled by the manager.
3. **AmbientProfile**: a ScriptableObject holding layers that reference tracks and define desired volumes.
4. **AudioManager Ambient Stack**: determines desired tracks and voice budget, scores emitters, and fades them in/out.

AmbientProfile & AmbientLayer

- AmbientProfile contains AmbientLayer[] layers and a defaultFade.
- Each AmbientLayer references an AmbientTrack and a target volume (0..1).
- Layers include additional metadata (priority, randomStartTime, pitchRange) and validate ranges in editor.

AmbientEmitter

- Requires an AudioSource, registers/unregisters with AudioManager on enable/disable.
- EnsurePlaying(AudioMixerGroup) assigns clip and plays (optionally random start time and pitch)..
- SetTargetVolume01() sets the desired volume; StepVolume() interpolates and stops the source when faded to ~0.

- Ambient looping is controlled by `AmbientEmitter.loop` (per emitter). `AmbientTrack` does not define looping at runtime; emitters decide whether the clip loops.

Scoring & Voice Budget (How it decides what's audible)

At ambientRescoreInterval, the manager:

- Builds **desired volume per track** from all ambient profiles in the stack (max volume wins).
- Scores each emitter using:
    - stack entry priority
    - emitter priority
    - base volume
    - audibility term derived from distance to the listener
- Selects up to maxAmbientVoices emitters (and optionally enforces one emitter per track).
- Ensures allowed emitters are playing and sets their target volumes; disallowed emitters fade out.

Listener transform is chosen from listenerOverride, or auto-detected via AudioListener / Camera.main.

AmbientZone (Level Design Workflow)

AmbientZone is a trigger volume that activates a profile on enter.

- Supports multi-collider characters by tracking colliders inside and only reacting on the first enter / last exit.
- Two modes:
    - **Replace**: SetAmbientProfile(profile, fade)
    - **Stack**: PushAmbientProfile(profile, priority, fade) storing a token for later pop.
- Exit behavior:
    - Replace: optionally ClearAmbient(fade) / ClearAmbient(0)
    - Stack: pops the stored token (optionally with fade).
- Draws gizmos for Box/Sphere/Capsule trigger colliders.

6.6 Gameplay Triggers

AudioTrigger is a simple collider trigger that calls AudioManager actions.

- Requires a Collider and sets it to trigger in Reset().

- Filters by TagToCompare (default Player) and supports firing on enter and/or exit.
- Supports delay (playDelay) and fade durations (fadeDuration).

SFX actions

- Play2D → AudioManager.PlaySfx2D(soundName)
- Play3D → AudioManager.PlaySfx3D(soundName, position) with optional local-space override position.

Music actions

- Play / PlayFadeIn → AudioManager.PlayMusic(trackName, 0f, fadeDuration)
- StopMusic → AudioManager.StopMusic(fadeDuration)
- Snapshot → AudioManager.TransitionToSnapshot(snapshotName, fadeDuration)

Ambient actions

- Play / PlayFadeIn creates a temporary AmbientProfile with one layer referencing the assigned AmbientTrack, then pushes it and stores the token.
- Stop / PopAmbient pops the stored token and clears it.

6.7 Mixer Snapshots (User-Defined)

- What it is (list of Name + Snapshot reference)
- Naming rules (unique names, avoid trailing spaces)
- How to call it (string API)
- How AudioTrigger uses it (Snapshot + snapshotName)

7. Editor Tools

7.1 AudioManager Inspector (Runtime Tools)

AudioManagerEditor adds a "🔧 Runtime Tools" panel with:

- **Utilities (Editor):**
    - Set root audio folder
    - One-click pipeline: Scan → Generate → Assign
    - Refresh clip lists / refresh emitters
- **SFX testing:** select a sound name, play 2D/3D at a position, or preview in editor.
- **Music testing:** play with delay and fade settings, stop with fade, preview in editor.
- **Ambient stack testing:** set/clear profile, push/pop profile, pop token, inspect last token.
- **Emitters browser:** lists scene AmbientEmitter objects, ping/select them, preview their clips.
- **Snapshots:** select from the configured snapshot list.
- **Debug meters:** shows ambient stack count and dB meters for mixer parameters.

7.2 AudioTrigger Inspector

AudioTriggerEditor provides a more guided inspector:

- Shows AudioManager presence status ("Found in scene / Not found").
- Displays only the relevant fields for selected Audio Type and Action.
- Adds preview buttons for assigned clips (SFX/Music/Ambient).
- Offers runtime music buttons (Play/Fade/Stop) if AudioManager is found.

7.3 Asset Tools: Scan → Generate → Assign

This editor-only pipeline lives in AudioManagerAssetTools.Editor.cs and can drastically speed up setup:

1. **Set Root Audio Folder** (must be under Assets/).
2. **ScanFolders**
    - Finds all AudioClips under the folder
    - Categorizes them using folder/name hints, then by clip length thresholds.
3. **GenerateScriptableObjects**
    - Creates GeneratedTracks/Music → MusicTrack

- Creates GeneratedTracks/Ambient → AmbientTrack
- Creates GeneratedTracks/SFX → SoundClipData grouped by parent folder or filename prefix.

4. **AssignToLibraries**
    - Adds generated assets to MusicLibrary, AmbientLibrary, SoundLibrary on the same GameObject.

Editor Preview Note: The preview buttons use reflection to call internal Unity editor utilities (UnityEditor.AudioUtil). If Unity changes internal APIs, preview may stop working, but runtime playback remains unaffected.

8. Recommended Folder Structure & Naming

While the system works with any layout, the Editor scanner performs best with clear folder naming conventions. It looks for hints
like /music/, /bgm/, /ambient/, /ambience/, /sfx/, /fx/ and filename hints
like theme, music, ambient, sfx.

Recommended structure:

Assets/
    Audio/
        Music/
        Ambient/
        SFX/

Recommended naming conventions:

- **SFX names**: stable keys like Footstep, Door_Open, UI_Click → stored in SoundClipData.soundName.
- **Music names**: stable keys like MainTheme, CombatLoop → stored in MusicTrack.trackName.
- **Ambient names**: stable keys like ForestWind, CaveDrips → stored in AmbientTrack.trackName.

9. Troubleshooting

9.1 “Volume sliders do nothing”

- Ensure your AudioMixer has exposed float parameters named exactly:
    - MasterVolume, MusicVolume, AmbientVolume, FXVolume
- Ensure mainMixer is assigned in the AudioManager inspector.

9.2 “No clips found in dropdowns / Runtime Tools shows ‘No SFX Found’”

- Confirm your libraries have assets assigned (tracks lists).
- Use the editor utilities: **Scan → Generate → Assign** to populate libraries automatically.
- Use each library’s context menu “Rebuild … Dictionary”.

9.3 “Ambient not playing”

Check these common causes:

- There are no AmbientEmitters in the scene (the editor panel will show a count).
- Emitters have no AmbientTrack assigned, or the track has no clip. AmbientEmitter will stop immediately in that case.
- Your ambient stack is empty (stack count shows in the AudioManager inspector).
- Your zone is not triggering (tag mismatch or collider not set as trigger). AmbientZone requires a trigger collider and filters by tagToCompare.

9.4 “Editor preview buttons don’t work”

- Preview uses reflection into Unity’s internal editor audio utilities. If Unity changes those internal APIs, preview may fail. Runtime playback is unaffected.

9.5 “AudioTrigger doesn’t compile”

If you are using a version where the SFX null/empty check was accidentally malformed, fix it as follows:

```
if (sfxClip == null || string.IsNullOrEmpty(sfxClip.soundName))
{
  return;
}
```

The trigger expects a valid SoundClipData and a non-empty soundName for SFX actions.

9.6 “Snapshots don’t work”

- Make sure the target snapshot exists in the AudioMixer.
- Also ensure it is added to AudioManager > Snapshots (User-Defined) with a unique name.
- Don’t forget to call TransitionToSnapshot("Name", time) with the exact configured name.

10. FAQ

Q: Does this require an AudioMixer?

Yes — AudioManager routes sources to AudioMixerGroups and sets exposed mixer parameters for volumes.

Q: Can I play many 3D SFX at once?

Yes — AudioSourcePool provides pooled sources, and if the pool is empty it will create an additional source to avoid missing playback.

Q: How does ambient voice limiting work?

The manager scores emitters by priority + distance and allows up to maxAmbientVoices (optionally enforcing only one emitter per track). It then fades allowed emitters toward their target volumes.

Q: Can I layer ambience (river over forest)?

Yes — use the ambient **stack** via PushAmbientProfile() or AmbientZone in **Stack** mode.

11. Support & Extending

Extending ideas (common additions)

- **Layer priority integration:** AmbientLayer contains a priority field but the provided manager scoring uses stack entry priority and emitter priority; you can incorporate layer priority into scoring if desired.
- **Listener caching:** ambient scoring calls listener discovery; you can set listenerOverride for explicit control.

For Additional Support

snogdev@gmail.com

Credits

Created by Snog / Pedro Schenegoski.