An audio manager template

Audio Collection Guide

Teal
Games

Table of Contents

Contents

This asset contains a variety of assets for audio for your game. It contains lots of simple setup for a basic audio system with an **Audio Manager**.

20+ Scripts included!

Controlling music and **ambience** with the Audio Manager!

Audio fading system!

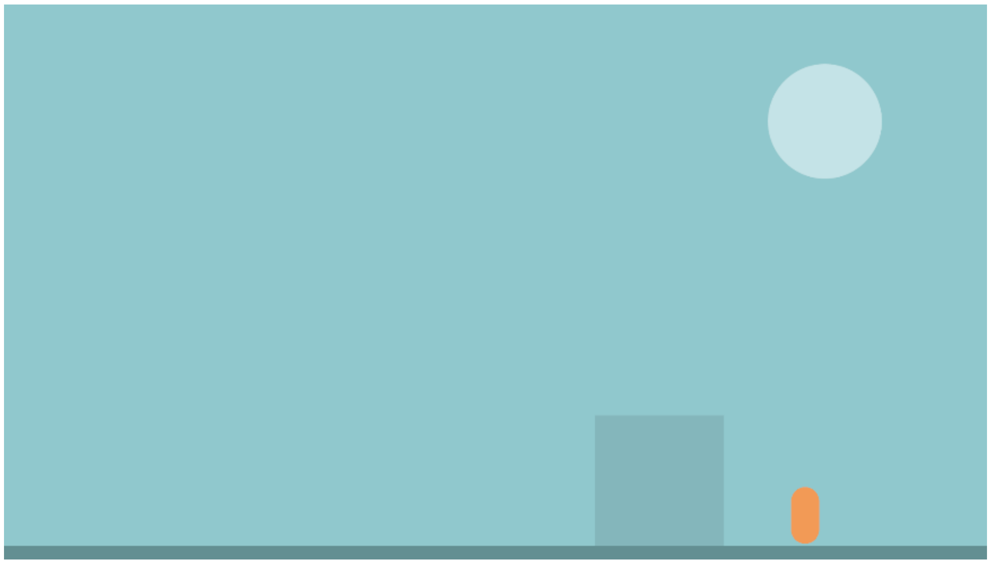
Lots of Scriptable Objects for setting lots of different presets for audio such as fade settings, background audio and others!

An **audio showcase scene** that includes a showcase of some of the features!

And so much more!

How to use this asset

This asset is meant to be used as a simple starting point for creating a basic audio system/manager for your game! Feel free to modify anything, add more functionality and change it to fit your game. This guide will teach you how the base asset works so you can understand what is happening behind the scenes.

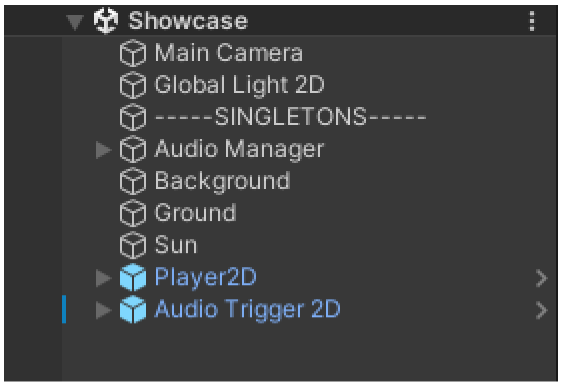


Setup/Architecture

This asset is set up in a very particular way that may vary from other assets. Firstly, it uses the **Singleton pattern** to set up managers that can be accessed via their static single instance. It allows for global managers that can be accessed anywhere. *However, it is important to note that too much reliance on singletons can result in spaghetti-like code that can be difficult to track. These should always be under the "SINGLETONS" header.*

A lot of features in this asset can be showcased in the 1 scene included in this asset: ***Showcase.*** Open the scene in order to access a showcase of one way that all the systems can be implemented. It is a great starting point for setting up the audio for your game! *When describing the systems in this guide, it is important to note that it will always reference the behaviors found for the system in the showcase scenes mentioned above.*

Tip: Use Pascal Case (ex. MainMenu) and no spaces for scenes in case you want to find them using strings (though it is better to use indices rather than strings).



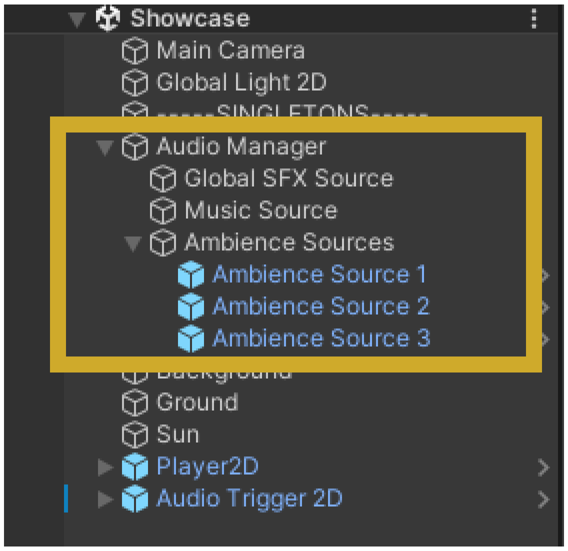
Audio Manager

The Audio Manager is a special Singleton that contains all the global audio sources. If you want to play a global sound that is not associated with any object, this is the way to do it. However, it is highly recommended to play most sounds that are associated with an object on that object so it can have control over its own sound. Most of the time, only music and ambience as well as background audio should really only be played with the Audio Manager. The Audio Manager also has a bunch of simple and utility methods for audio that can be used in other classes.

Audio Manager Setup

The Audio Manager is set up to contain all the Audio Sources as children. It is advised to follow the same structure so that all the children will stay with the parent. There are 3 types of Audio Sources available (but you can add more): a global sound effect source, a music source and 3 ambience sources.

There are 3 ambience sources because you might have multiple background ambience tracks while most of the time you will only ever play one music track at a time. However, this can easily be changed so that there are multiple music sources or sound effects sources. *It is important to note that you should try to only play one clip at a time on an Audio Source* (since you can play multiple clips with PlayOneShot()) since it makes it more manageable and clear what is going on.



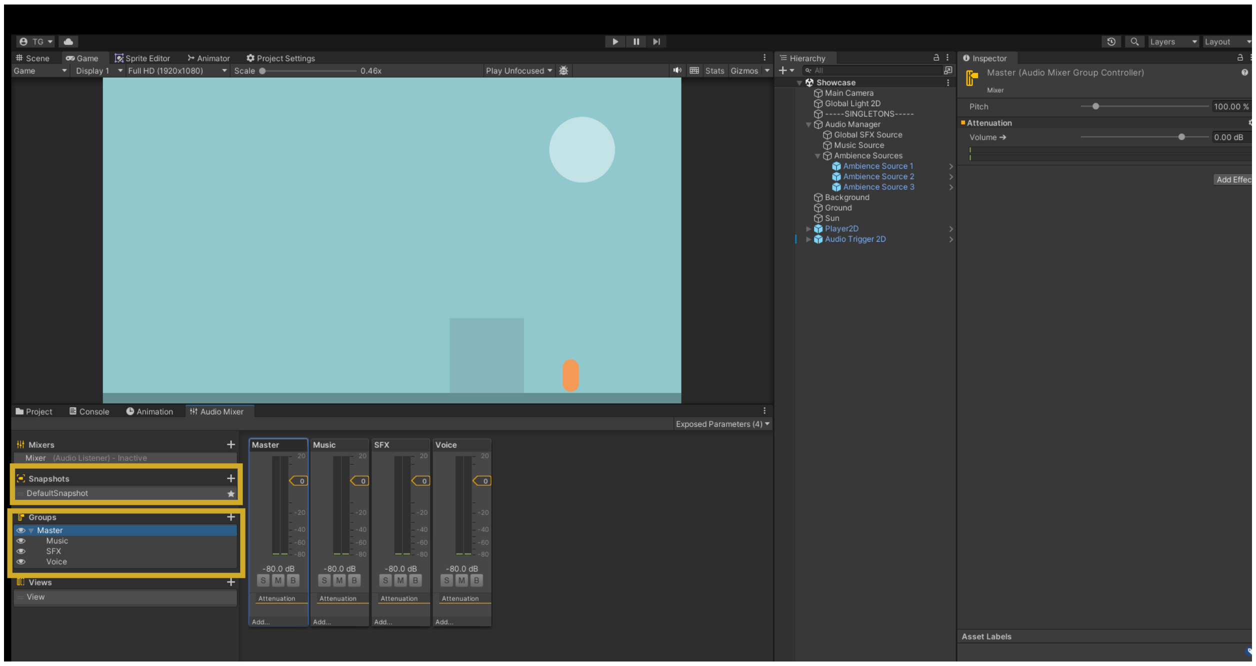
Adding a Custom Source Type

To add another source type, you should first add the type to the **GlobalSourceType** enum. Then, you should add an Audio Source to the hierarchy under the AudioManager. *If there are multiple Audio Sources for that type, they should all be grouped under one parent.* Then, make a serialized field in the Audio Manager so you can add a reference to the Audio Source(s) in the inspector. Finally, make sure to add the source/type when retrieving it with the **GetAvailableAudioSourceFromType()** method. Follow the structure in that method with your own type.

Audio Mixer

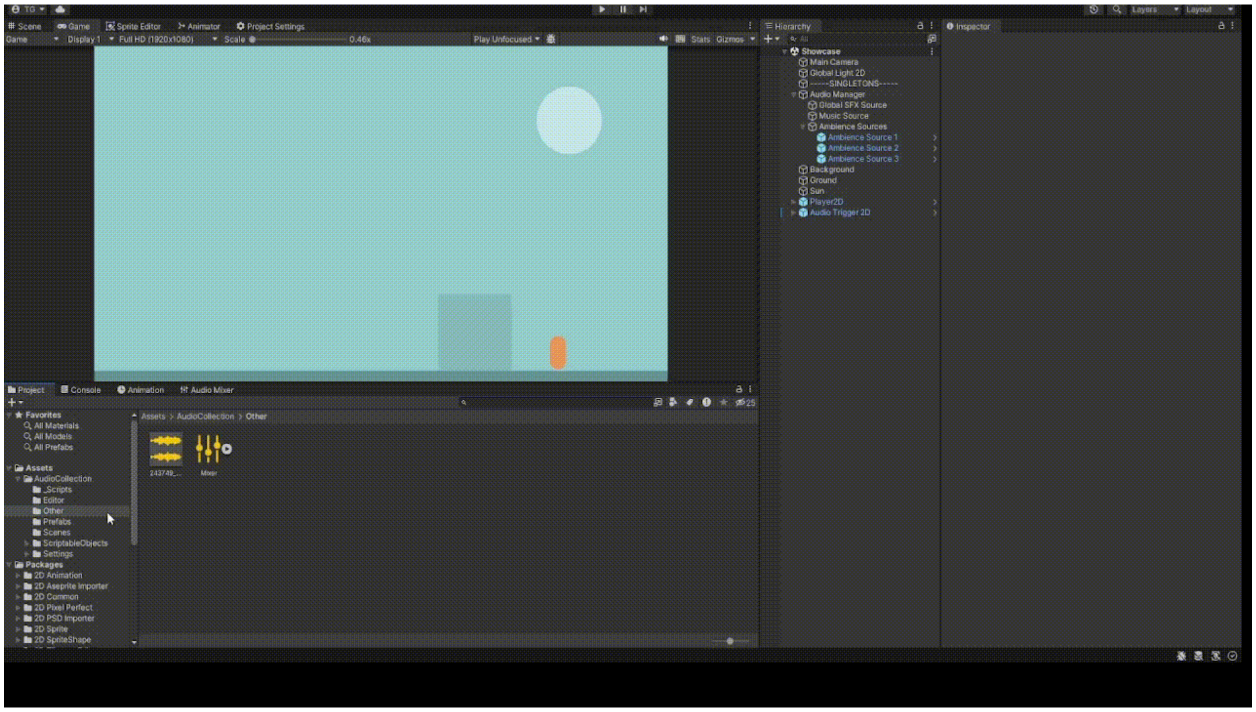
With the use of Unity's Audio Mixer, it is really easy to control global audio. *Audio Mixer Groups can be used to modify Audio Source output before it reaches the Audio Listener.* This system only works if each Audio Source has an output to one of the groups. That way you can group the output of different sources together and control them. This allows for creating volume settings in a Settings menu or for adding additional effects to the output of some sources. For example, an underwater area has different additional effects compared to a city.

Audio Mixer Snapshots allow you to transition between different Audio Mixer group settings. This can allow you to set presets or "snapshots" of certain instances of the game and transition between them. For example, you can set a snapshot of the settings for an underwater area and then transition when you enter that area to that snapshot.



Adding a new Audio Mixer Group

When you add another Audio Mixer Group, make sure it is under the “Master” parent group because the Master group controls the overall audio of the whole game. When you add a Mixer Group, you might want to be able to set the volume of that group in a settings menu. To do that, you might want to expose the volume parameter to a script. **The “Master” group should not change because it is used as a constant in the AudioManager code.** _For your convenience, you can also declare constants for any of the other mixers if you want to use them in the code in order to only have one location that holds the data._

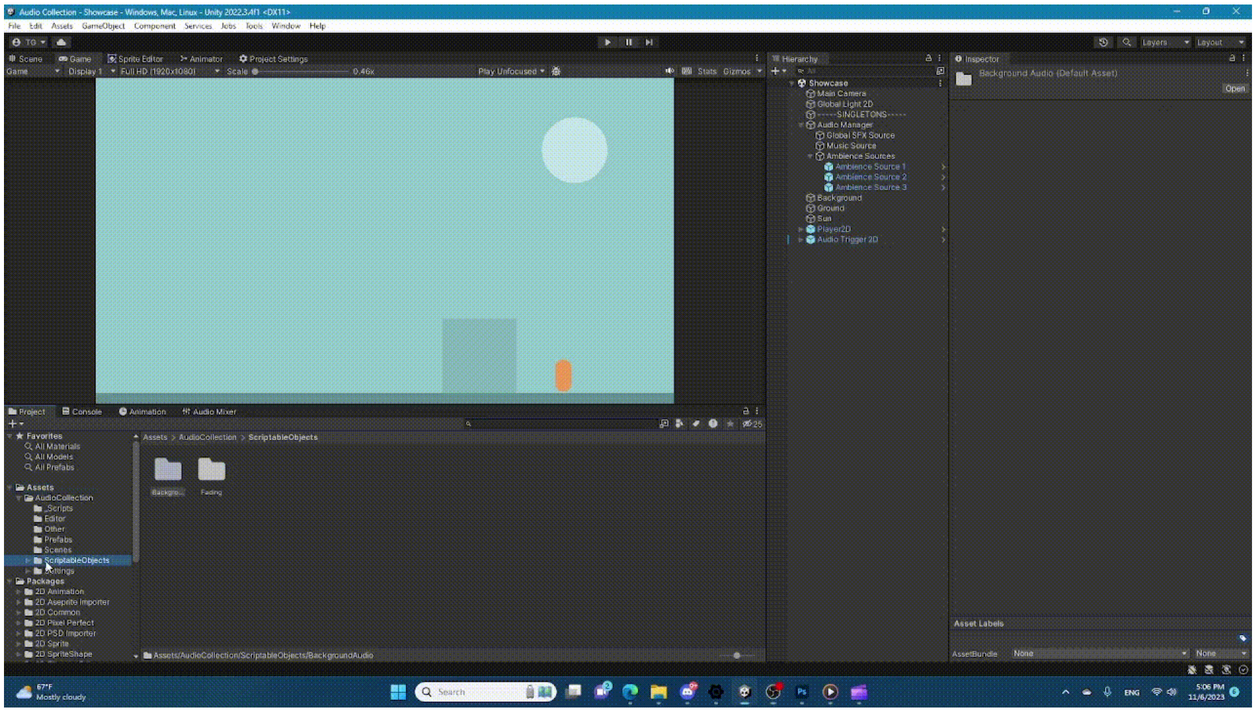


Background Audio

Background Audio is any type of audio that can be played in the background by the Audio Manager, such as music or ambience. It is highly customizable and easy to make, created simply by making a Scriptable Object.

Setting Up Background Audio

To set it up, simply make a Scriptable Object and then set up all the preferences you want.



Now it is ready to be used when triggering it!

Random Background Audio

Background Audio has an option to be played randomly in the background at random intervals. This means that based on the settings in the Audio Manager for generic settings for random background audio, as well the settings in the Scriptable Object can change how it works! *It is important to note that Background Audio can only be either played in the background or played normally, and not both.*

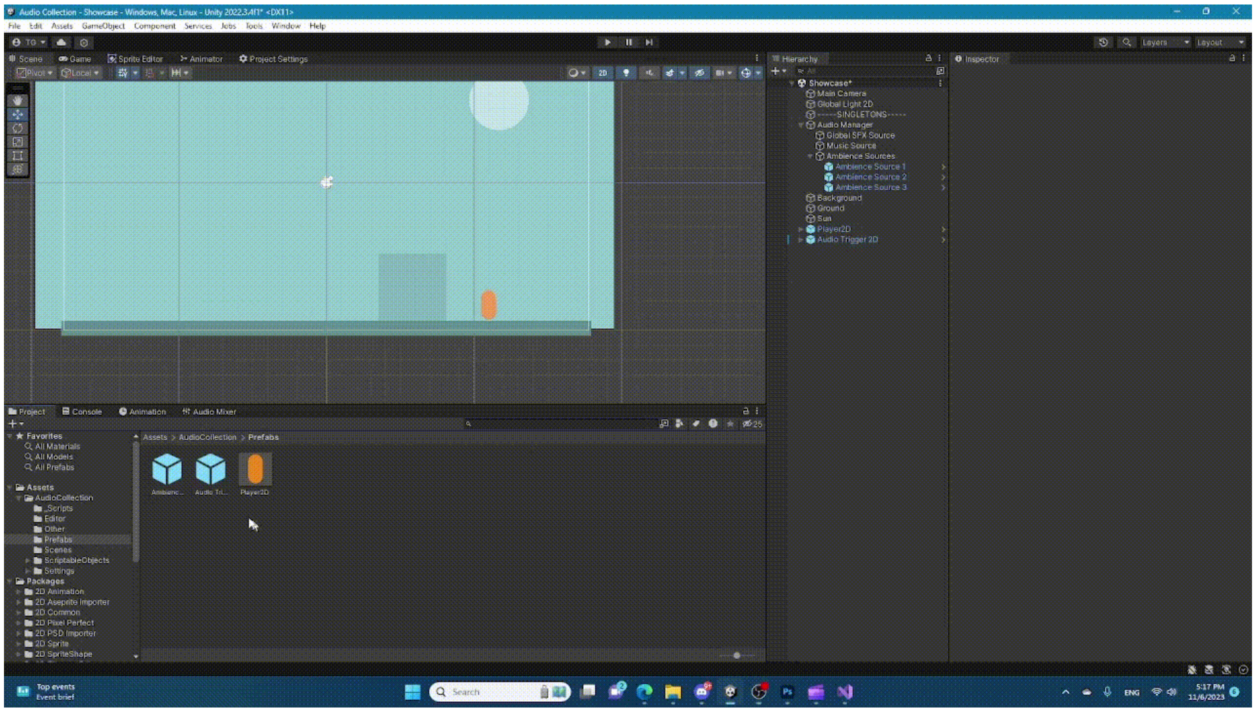
When a group of Background Audio is selected to start being played in the background, a random interval is chosen before any audio is played. Once that time is up, it chooses a random Background Audio (from the ones that have enabled it to be used in random background audio) based on the play chance. *If the play chance of all the audio does not add up to 100, it means silence can also be randomly chosen as an option.* If it is, a random amount of silence is chosen, otherwise a track is played. Another interval time is once again chosen based on the minimum and maximum interval time and once it is over, another random audio is chosen.

Triggering Audio

There are 2 ways to trigger background audio, using a trigger as well as triggering it internally with code.

To trigger it with a trigger, there are 2 options: **AudioTrigger2D** or **AudioTrigger3D**. *Both work the same, but are suitable based on the dimension for your game.* Simply add another element to the triggered background audio list in the inspector and set it up with the correct settings that apply to your Background Audio. Now when another collider enters the trigger collider, it should activate! *Since Background Audio is only really meant for music and ambience, you cannot really use Background Audio Scriptable Objects.* However, you can still use the AudioTrigger to trigger a global sound to be played in the Audio Manager by simply *invoking PlayGlobalSound() from the Audio Manager in a UnityEvent in the Audio Trigger with the sound you want!*

The second way is to trigger it directly by using the methods. *Just be careful that you use the right one!* Sounds played in the background at random intervals have different methods than ones that are played directly and instantly! This also applies to global sounds as well!


Audio Collection - Showcase - Windows, Mac, Linux - Unity 2022.3.4f1* <DX11>
File Edit Assets GameObject Component Services Jobs Tools Window Help
Scene Game Sprite Editor Animator Project Settings
Hierarchy
Showcase*
Main Camera
Global Light 2D
-----SINGLETONS-----
Audio Manager
Global SFX Source
Music Source
Ambience Sources
Ambience Source 1
Ambience Source 2
Ambience Source 3
Background
Ground
Sun
Player2D
Audio Trigger 2D
Inspector
Project Console Animation Audio Mixer
Favorites
All Materials
All Models
All Prefabs
Assets
AudioCollection
_Scripts
Editor
Other
Prefabs
Scenes
ScriptableObjects
Settings
Packages
2D Animation
2D Aseprite Importer
2D Common
2D Pixel Perfect
2D PSD Importer
2D Sprite
2D SpriteShape
Assets > AudioCollection > Prefabs
Ambienc...
Audio Tri...
Player2D
Top events
Event brief
Search
5:17 PM
11/6/2023

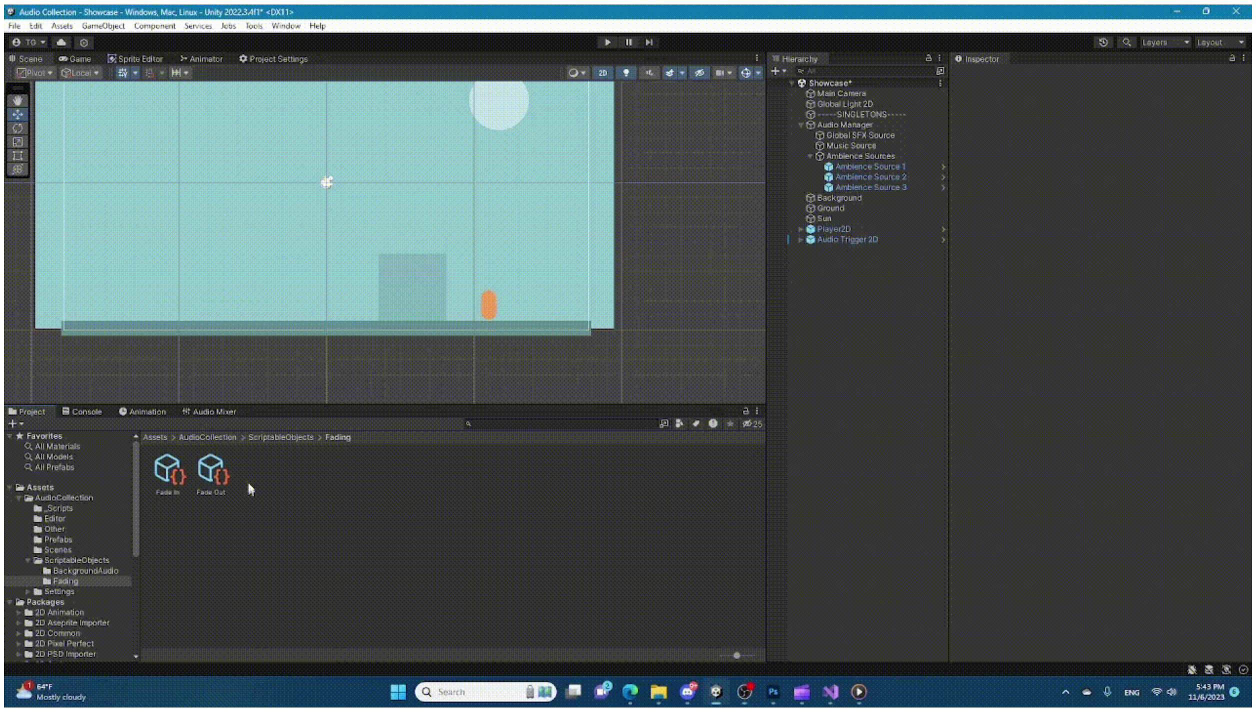
Audio Volume Fading

There are many ways to accomplish fading, whether it is fading the volume up or down. The fade methods are all located in the Audio Manager. You can fade the volume of a Mixer as well as of an Audio Clip that is being played in an Audio Source. Simply call the appropriate method from Audio Manager.

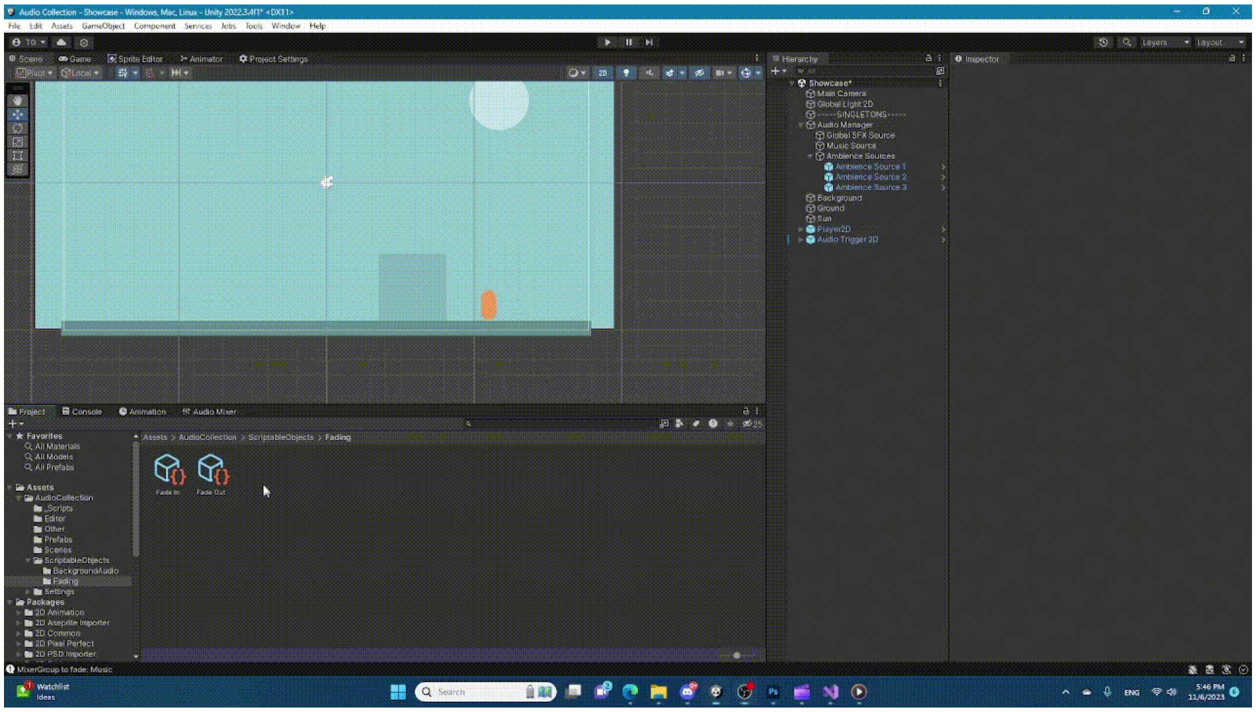
Creating Fade Settings

Both types of audio fading require fade settings. You can either create one directly by creating a new Fade Settings object based on the type you want. This allows you to create Fade Settings directly in your code or serialize them in the Inspector as fields that are set. The other option is to create a Scriptable Object so that you can store the data in one place and have multiple fade settings presets which can be shared by different Background Audio Scriptable Objects.

This shows how to create Audio Mixer fade settings.



This shows how to create Audio Source fade settings.

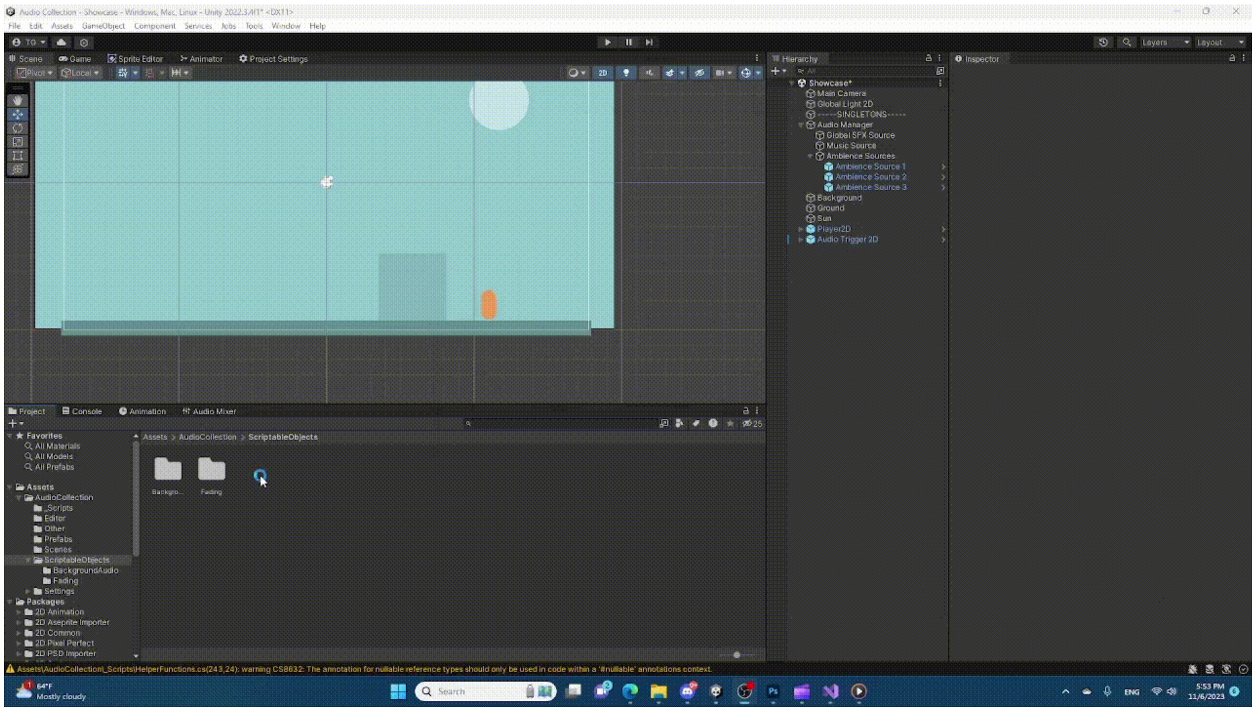


AudioClip+

AudioClip+ is a simple feature used to allow for more Audio Clip customization that is linked directly with an AudioClip so that it can then be set in the AudioSource. This does not change the Audio Clip, but rather it is a set of settings that will be applied when the clip is played in an Audio Source. This is not meant to be used with music or ambience, but instead only with sound effects.

It can be added as a group of settings that can be set in a class in the Inspector with "AudioClipPlus" class, or its variant "AudioClipPlusPreset" which allows you to set the extra settings from a Scriptable Object preset. Finally, the "AudioClipPlusPresetSO" is the Scriptable Object with only the extra settings (without Audio Clip setting).

This video shows how to make a AudioClip+ preset Scriptable Object.

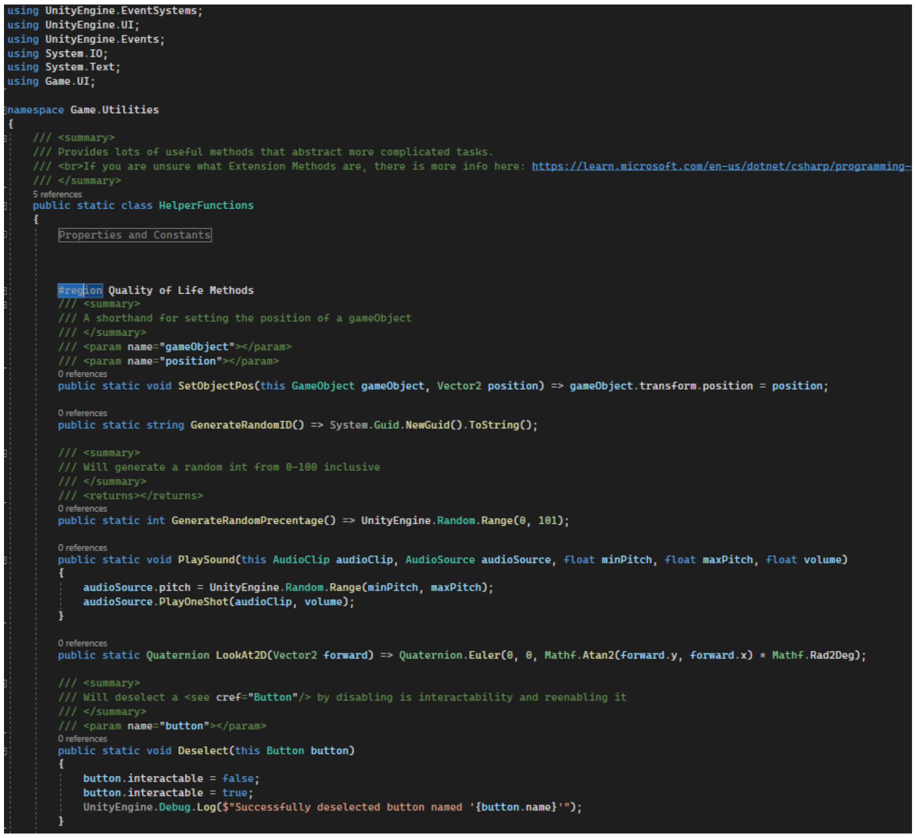


Assets\AudioCollection_Scripts\HelperFunctions.cs(243,24): warning CS8632: The annotation for nullable reference types should only be used in code within a '#nullable' annotations context.

Helper Functions

The Helper Functions class is not a MonoBehavior but rather it is a static class that has lots of utility methods. Meant at avoiding repetition, it has methods that do commonly used tasks as well as methods that abstract more complicated tasks. Simply call a method from Helper Functions by first using the name of the class, Helper Functions, followed by the method name. There is no need to create objects since only one instance ever exists!

Each method has a summary above it describing it and giving some more information!



```
using UnityEngine.EventSystems;
using UnityEngine.UI;
using UnityEngine.Events;
using System.IO;
using System.Text;
using Game.UI;

namespace Game.Utilities
{
    /// <summary>
    /// Provides lots of useful methods that abstract more complicated tasks.
    /// <br>If you are unsure what Extension Methods are, there is more info here: https://learn.microsoft.com/en-us/dotnet/csharp/programming-
    /// </summary>
    5 references
    public static class HelperFunctions
    {
        Properties and Constants


        #region Quality of Life Methods
        /// <summary>
        /// A shorthand for setting the position of a gameObject
        /// </summary>
        /// <param name="gameObject"></param>
        /// <param name="position"></param>
        0 references
        public static void SetObjectPos(this GameObject gameObject, Vector2 position) => gameObject.transform.position = position;

        0 references
        public static string GenerateRandomID() => System.Guid.NewGuid().ToString();

        /// <summary>
        /// Will generate a random int from 0-100 inclusive
        /// </summary>
        /// <returns></returns>
        0 references
        public static int GenerateRandomPrecentage() => UnityEngine.Random.Range(0, 101);

        0 references
        public static void PlaySound(this AudioClip audioClip, AudioSource audioSource, float minPitch, float maxPitch, float volume)
        {
            audioSource.pitch = UnityEngine.Random.Range(minPitch, maxPitch);
            audioSource.PlayOneShot(audioClip, volume);
        }

        0 references
        public static Quaternion LookAt2D(Vector2 forward) => Quaternion.Euler(0, 0, Mathf.Atan2(forward.y, forward.x) * Mathf.Rad2Deg);

        /// <summary>
        /// Will deselect a <see cref="Button"/> by disabling is interactability and reenabling it
        /// </summary>
        /// <param name="button"></param>
        0 references
        public static void Deselect(this Button button)
        {
            button.interactable = false;
            button.interactable = true;
            UnityEngine.Debug.Log($"Successfully deselected button named '{button.name}'");
        }
```

Leaving Feedback

Feedback is very important in order for me to improve the quality and value of the asset. I wanted to price this as low as possible so that developers can easily buy it. I understand the quality or value may not be at the highest level, so giving me feedback will make this asset better overall. Don't hesitate to refund it, but if you do, leave a review telling me why you did so. Reviews always help out, so please leave one no matter if it is positive or negative! I plan on adding updates in the future as well as making more similar assets focused on different major systems of a game.

If you want to stay anonymous and submit feedback, you can do it here.

Thank you!