# Audio Manager (Offline)

ℹ️ For full docs, please see the online documentation! What is provided here is just a summary giving you just the info to use the asset.

**Online Documentation** | **Unity Asset Store** | **Carter Games**

## 🔴 Summary

The Audio Manager package is designed, like most of my assets is free and aimed to be user friendly and easy to use. Once installed into your project, you may call for sounds to be played pretty much anywhere you want. With options to control the location, volume, pitch, time and delay and more when calling for a sound to play.

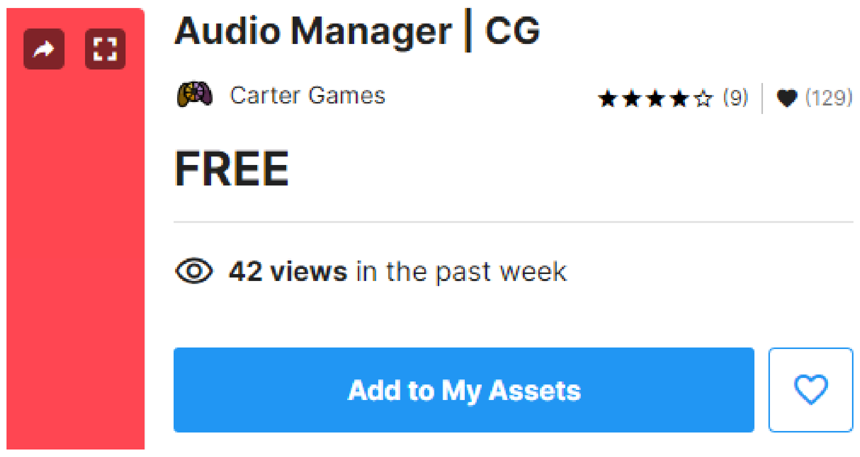
## 🌟 Thanks

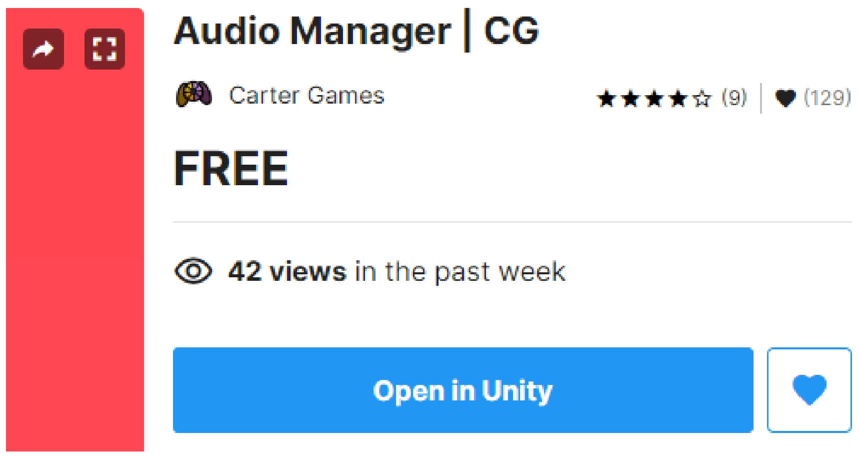
Thank you for deciding to use my asset for your project. If you like my asset, feel free to leave a `⭐⭐⭐⭐⭐` review! If you find that our asset is not up to scratch or find and issue, please do let me know either via our email: **hello@carter.games** and I will do my best to help you with the issues you are facing. I can't read minds, so if you don't speak up, it won't get fixed 😆

# ℹ️ Essential Information

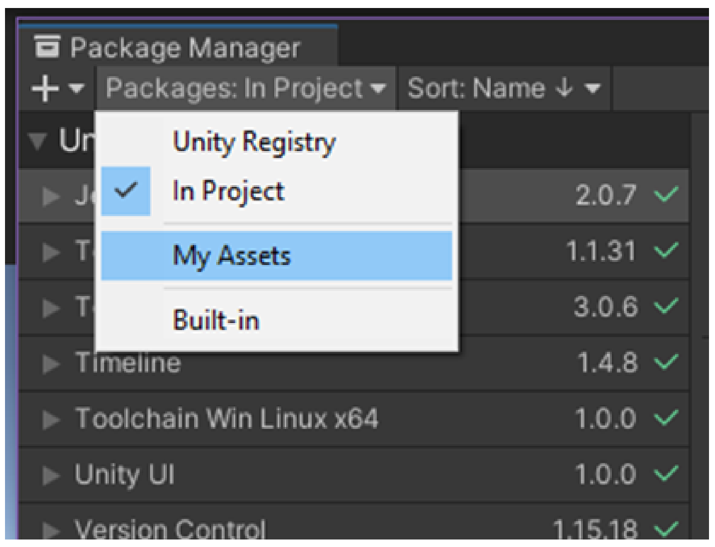
## ➕ Installation

### Unity Asset Store





To get the asset you'll have to press the `Add to My Assets` button. Doing so will add the asset your account so you can access it in whichever version of Unity you want that the asset is supported in. From the asset store you can then press the `Open in Unity` button. Doing so will open Unity on your system with the package to use.



When in Unity you can import the asset via the package manager, found under `Window → Package Manager`

The window will have to option to filter to the packages, from here you can select My Assets. You may need to login to your UnityID to see your assets. Once the packages appear, the Audio Manager asset will appear on the list. Here you can download the latest version of the asset and then import it via the button in the bottom right of the Package Manager panel.

Once you press the import button, a package window may appear with information about what the package contains and options for you to choose what to import. You should import the full package as there are no optional elements other than the text mesh pro support package. When ready, just press import and the package will import into project, the editor will reload, and you will be good to go.

### Other Source

If installing from off the asset store, such as git, itch.io or the carter games website. You'll simply get a .unitypackage file. To import the .unitypackage into your project just double click it with your project open or use the import custom package option in Unity under `Right click in Project Tab → Import Package → Custom Package`

## 🔺 Updating to a newer version

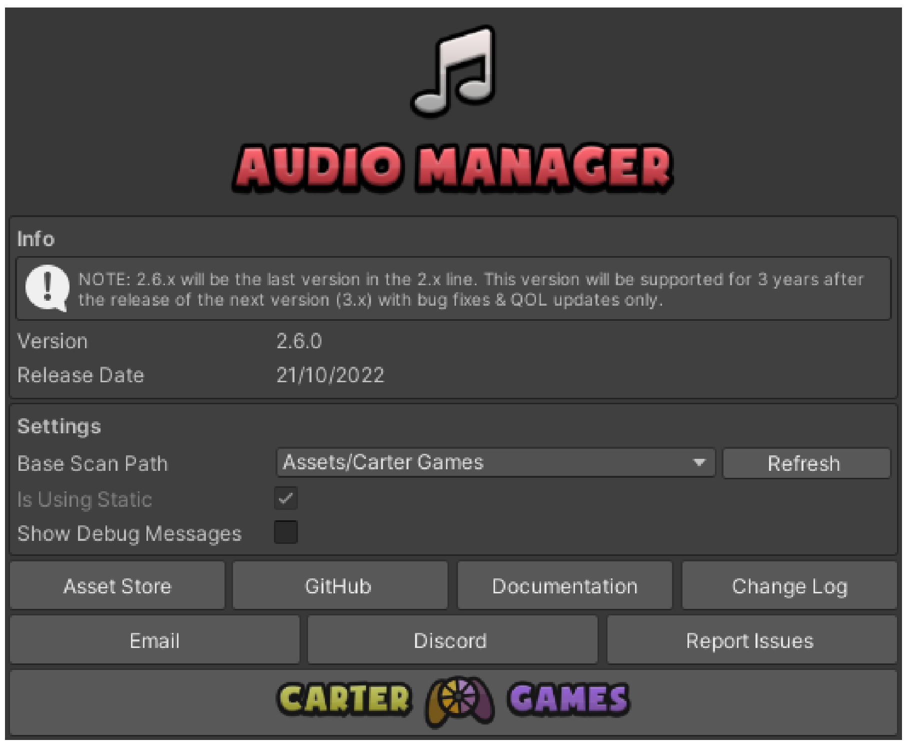
If you are updating the asset from an older version it is best to delete the asset folder & do a clean install for the least amount of friction with the new version. If the update is just a patch you should be good to just import on top of the old files, but consider a clean install if the asset doesn't work as intended or throws an unknown error.

# ⚙️ Asset Settings

## 🗺️ Location

The settings for the asset can be access via the following methods and has the icon shown on the right. This asset holds all settings, if deleted the asset will regenerate a new one with default settings when you next use the asset.

### Project Settings Window



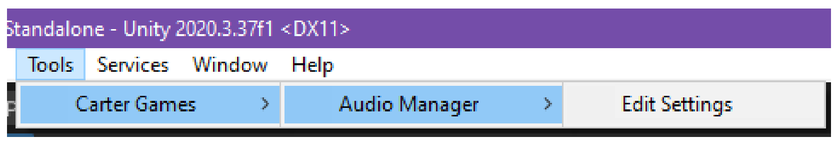
NOTE: 2.6.x will be the last version in the 2.x line. This version will be supported for 3 years after the release of the next version (3.x) with bug fixes & QOL updates only.

Version 2.6.0
Release Date 21/10/2022
Base Scan Path Assets/Carter Games
Is Using Static ✓
Show Debug Messages

You can edit the settings via this window. It can be found under `Project Settings/Carter Games/Audio Manager`

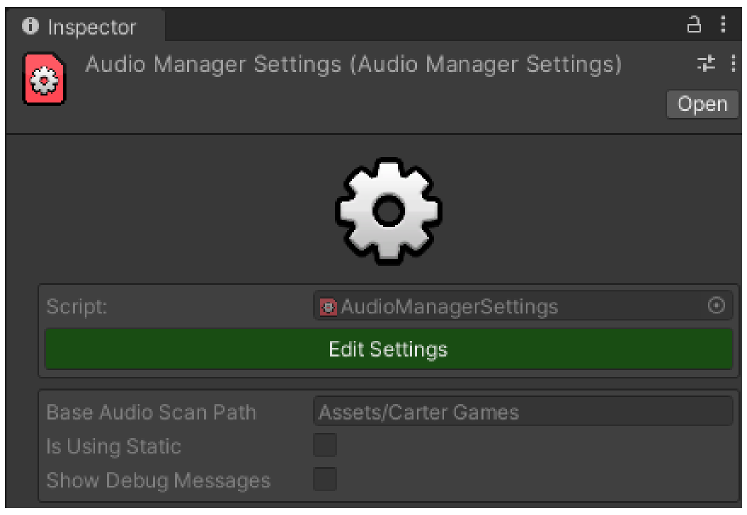
This window also shows the version number of the asset, release date & some helpful links as well as the settings for the asset. See what each option does here: ⚙️ Options.

### Menu Item



You can access the settings via a menu item on the top bar navigation menu which can be found under `Tools/Audio Manager/Edit Settings` This will open the project settings window on the build versions settings.

### Settings Asset Button



Script: AudioManagerSettings
Base Audio Scan Path: Assets/Carter Games
Is Using Static
Show Debug Messages

The settings asset can be found under `Assets/Resources/Carter Games/Audio Manager/Audio Manager Settings` This is the scriptable object that controls the settings of the asset. You can only view the settings from there. To actually edit the settings, press the `Edit Settings` button in the inspector of the asset.

## ⚙️ Options

### Base Audio Scan Path

This is the location in your project where the system looks for audio from. You can add additional directories in the inspector of the audio manager file or the audio manager using a file. You can set this field with a drop down which is categorised by folder like so:

| - Assets

| - Assets >

    | - Audio

    | —- Audio >

        | - Clicks

        | - Sfx

        | - Music

In the settings section of the asset you also have a refresh button next to the directories select. Use this to update the directories list if you have added a new directory that the system hasn't registered.

### Is Using Static

This setting is controlled by the asset and is used to update the scripting defines if you change platform with the static instance enabled so you don't have to update it yourself.

### Show Debug Messages

Toggles if the asset shows any messages in the debug console. Handy to turn off if you want to reduce the amount of messages the console receives.

## Basic Scripting API

For the full API rundown please see the online documentation, this is just a short summary of the most essential stuff to get to using the asset.

### ✅ Assemblies

All the code in this asset is under assembly definitions. This helps with compile times for your code & keeps the asset code all neat & out of the way. If your code is also in an assembly definition you will need reference the assembly/assemblies to access the code for this asset. The assemblies are:

`CarterGames.AudioManager.Runtime`

`CarterGames.AudioManager.Editor`

`CarterGames.AudioManager.Demo`

### 🚩 Namespaces

All code for this asset is under the following namespace(s):

```
// Editor logic for the asset.
CarterGames.Assets.AudioManager.Editor

// Runtime logic for the asset.
CarterGames.Assets.AudioManager

// Demo logic, is not needed for the asset to work.
CarterGames.Assets.AudioManager.Demo
```

### Audio Manager

**`CanPlayAudio`**

`Property`

`Returns` `bool`

Toggles whether or not the manager can play audio. This doesn't effect other Audio Manager instances or the other player scripts.

**`Play`**

`Method`

**`Declarations`**

```
public void Play(string clip, float volume = 1f, float pitch = 1f);
public void Play(string clip, AudioMixerGroup mixerGroup, float volume = 1f, float pitch = 1f);
public void Play(string clip, int mixerId, float volume = 1f, float pitch = 1f);
public void Play(string clip, AudioArgs args);
```

`Parameters`

- `string` → The name of the clip to play.
- `AudioMixerGroup` → The mixer group to play in.
- `int` → The id of the mixer to play in from the Audio Manager inspector.
- `AudioArgs/Hashtable` → The audio args or hash-table to setup the player.
- `float` `optional` → The volume for the clip (between 0, 1).
- `float` `optional` → The pitch for the clip (between -3, 3).

Plays the requested clip with optional values for volume and pitch. There are several overload methods are also available for audio mixers, either by manual assignment or via the audio manager mixer ID number.

```
AudioManager.instance.Play("MyClip");
AudioManager.instance.Play("MyClip", mixer);
AudioManager.instance.Play("MyClip", mixerID);
AudioManager.instance.Play("MyClip", args);
```

## `PlayFromTime`

`Method`

`Declarations`

```
public void PlayFromTime(string clip, float time, float volume = 1f, float pitch = 1f);
public void PlayFromTime(string clip, float time, AudioMixerGroup mixerGroup, float volume = 1f, float pitch = 1f);
public void PlayFromTime(string clip, float time, int mixerId, float volume = 1f, float pitch = 1f);
public void PlayFromTime(string clip, float time, AudioArgs args);
```

`Parameters`

- `string` → The name of the clip to play.
- `float` → The time to play the clip from.
- `AudioMixerGroup` → The mixer group to play in.
- `int` → The id of the mixer to play in from the Audio Manager inspector.
- `AudioArgs/Hashtable` → The audio args or hash-table to setup the player.
- `float` `optional` → The volume for the clip (between 0, 1).
- `float` `optional` → The pitch for the clip (between -3, 3).

Plays the requested clip with optional values for volume and pitch from the set time, handy if the clip doesn't start right away. There are several overload methods are also available for audio mixers, either by manual assignment or via the audio manager mixer ID number.

```
AudioManager.instance.PlayFromTime("MyClip", 0.1f);
AudioManager.instance.PlayFromTime("MyClip", 0.1f, mixer);
AudioManager.instance.PlayFromTime("MyClip", 0.1f, mixerID);
AudioManager.instance.PlayFromTime("MyClip", 0.1f, args);
```

## `PlayWithDelay`

`Method`

`Declarations`

```
public void PlayWithDelay(string clip, float delay, float volume = 1f, float pitch = 1f);
public void PlayWithDelay(string clip, float delay, AudioMixerGroup mixerGroup, float volume = 1f, float pitch = 1f);
public void PlayWithDelay(string clip, float delay, int mixerId, float volume = 1f, float pitch = 1f);
public void PlayWithDelay(string clip, float delay, AudioArgs args);
```

`Parameters`

- `string` → The name of the clip to play.
- `float` → The delay before playing the clip.
- `AudioMixerGroup` → The mixer group to play in.
- `int` → The id of the mixer to play in from the Audio Manager inspector.
- `AudioArgs/Hashtable` → The audio args or hash-table to setup the player.
- `float` `optional` → The volume for the clip (between 0, 1).
- `float` `optional` → The pitch for the clip (between -3, 3).

Plays the requested clip with optional values for volume and pitch after the entered delay. There are several overload methods are also available for audio mixers, either by manual assignment or via the audio manager mixer ID number.

```
AudioManager.instance.PlayWithDelay("MyClip", 0.1f);
AudioManager.instance.PlayWithDelay("MyClip", 0.1f, mixer);
AudioManager.instance.PlayWithDelay("MyClip", 0.1f, mixerID);
AudioManager.instance.PlayWithDelay("MyClip", 0.1f, args);
```

### `PlayFromRange`

`Method`

`Declarations`

```
public void PlayFromRange(List<string> clips, float delay, float volume = 1f, float pitch = 1f);
public void PlayFromRange(List<string> clips, float delay, AudioMixerGroup mixerGroup, float volume = 1f, float pitch = 1f);
public void PlayFromRange(List<string> clips, float delay, int mixerId, float volume = 1f, float pitch = 1f);
public void PlayFromRange(List<string> clips, float delay, AudioArgs args);
public void PlayFromRange(string[] clips, float delay, float volume = 1f, float pitch = 1f);
public void PlayFromRange(string[] clips, float delay, AudioMixerGroup mixerGroup, float volume = 1f, float pitch = 1f);
public void PlayFromRange(string[] clips, float delay, int mixerId, float volume = 1f, float pitch = 1f);
public void PlayFromRange(string[] clips, float delay, AudioArgs args);
```

`Parameters`

- `List<` `string` `>` `string[]` → The names of the clips to play.
- `AudioMixerGroup` → The mixer group to play in.
- `int` → The id of the mixer to play in from the Audio Manager inspector.
- `AudioArgs/Hashtable` → The audio args or hash-table to setup the player.
- `float` `optional` → The volume for the clip (between 0, 1).
- `float` `optional` → The pitch for the clip (between -3, 3).

Plays a random clip from the entered strings with optional values for volume and pitch. There are several overload methods are also available for audio mixers, either by manual assignment or via the audio manager mixer ID number.

```
AudioManager.instance.PlayFromRange(new string[] { "MyClip", "MyOtherClip" });
```

## Audio Player

### `Play`

`Method`

`Declaration`

```
public void Play()
```

Plays all the clips assigned in the inspector. This uses the settings from the inspector for the clip to set the values for it. You can edit the volume, pitch, start time & delay from the custom inspector for each clip in the player.

```
AudioPlayer.Play();
```

## Music Player

### `IsTrackPlaying`

`Property`

`Returns` `bool`

Gets whether or not there is a track currently playing?

### `GetActiveTrack`

`Property`

`Returns` `AudioClip`

Gets the track that is currently being player by the music player script.

## `ShouldLoop`

`Property`

Gets/Sets whether or not the active track should loop.

## `SetVolume`

`Method`

**`Declaration`**

```
MusicPlayer.SetVolume(float value);
```

`Parameters`

- `float` → The volume to set between 0-1.

Sets the volume of the music player to the entered value. Range is between 0-1.

```
MusicPlayer.SetVolume(.75f);
```

## `StopTrack`

`Method`

**`Declaration`**

```
MusicPlayer.StopTrack();
```

Stops the active track from playing without any transition.

```
MusicPlayer.StopTrack();
```

## `PlayTrack`

`Method`

**`Declarations`**

```
MusicPlayer.instance.PlayTrack(AudioClip track)
MusicPlayer.instance.PlayTrack(AudioClip track, TransitionType transitionType)
MusicPlayer.instance.PlayTrack(AudioClip track, TransitionType transitionType, float transitionDuration)
MusicPlayer.instance.PlayTrack(AudioClip track, float startTime)
MusicPlayer.instance.PlayTrack(AudioClip track, float startTime, TransitionType transitionType)
MusicPlayer.instance.PlayTrack(AudioClip track, float startTime, TransitionType transitionType, float transitionDuration)
MusicPlayer.instance.PlayTrack(AudioClip track, float startTime, float endTime)
MusicPlayer.instance.PlayTrack(AudioClip track, float startTime, float endTime, TransitionType transitionType)
MusicPlayer.instance.PlayTrack(AudioClip track, float startTime, float endTime, TransitionType transitionType, float transitionDuration)
```

`Parameters`

- `AudioClip` → The track that you wish to play. Use null if you want to just end the current track without changing to a new one.
- `TransitionType` → The type of transition the music player should use when changing the track.
- `float` → Start Time → The time when the track should start playing from. Time is in seconds.
- `float` → End Time → The time when the track should end/loop from. Time is in seconds.
- `float` → Transition Duration → The amount of time the transition should take to complete. Default is 1 second.

Plays the track requested on the music player script. There are several overloads for this method that allow for additional options. If just using the audioclip param the transition will stay as the last one used in the inspector, the start time will be set to 0 and the end time will be set to the track length.

```
AudioManager.instance.Play("MyClip");
AudioManager.instance.Play("MyClip", mixer);
AudioManager.instance.Play("MyClip", mixerID);
AudioManager.instance.Play("MyClip", args);
```

## Args Container

`AudioArgs`

`Method`

`Declarations`

```
public Hashtable AudioArgs(params object[] args);
```

`Returns`

`Hashtable` → The generated table for use with the asset.

Generates a hashtable with the parameters you enter to save a few lines of code and make the system easier to use.

```
var args = ArgsContainer.AudioArgs("position", Vector3.zero, "time", .1f, "pitch", .8f);
```

## 🤷‍♂️ F.A.Q

### ▼ How does scanning work?

By default the script scans the "/audio" directory if you have left a directory field left blank, you can change the path via the **Audio Manager Global Settings** which we recommend you do before scanning for the first time. to go to any sub-directory within the "/audio" folder, or add additional directories using the inspector display. This field is not case sensitive as shown below:

- **Example 1:** mygame – "/audio/mygame" will be scanned
- **Example 2:** MyGame – "/audio/mygame" will be scanned (capitals are ignored)
- **Example 3:** MyGame/SFX – "/audio/mygame/sfx" will be scanned

The script will automatically update with the new sounds once you have entered a valid directory. Once scanned the script will list all the audioclips it has found. This will update on the fly when you add new files into your scanned directory with the audio manager selected.

As of 2.6.0 you can select the directories to scan from a drop down. Each section of the dropdown is next in the directory hierarchy. You can change how the drop down is shown in the settings of the asset.

### ▼ Why is a file not scanning?

The audio manager will scan all valid audio file types that work with unity. There are a few reasons why a clip won't scan. Try some of the steps below if you are having this issue.

- Make sure the file type is accepted by Unity.
- Make sure the clip name is not the same as another in the project.
- If updating a clip with a new one with the same name, delete the old one and let the audio manager rescan, then add the new one.

### ▼ Why won't a clip won't play?

There can be a few reason why a clip won't play:

- Check that you have spelt the clip name correctly.
- Check that the clip is present in the audio manager file you are using.

### ▼ I called a function and the clip plays a million times! Why is this?

This is due to you having the call in an update() or similar, if you have the call in update you need to have either a Boolean or a coroutine to stop it been called more than once.

### ▼ The Inspector hasn't loaded correctly when I added the script, Why?

If this has happened, please sent me screenshots and ways to replicate the problem so I can fix it. email: support@carter.games

### ▼ Got an issue that isn't listed?

Please get in touch with us so we can do our best to help you out. Email: support@carter.games

## ❔ Support

### 📨 Email

You can email me any time through the support email: **hello@carter.games** and I aim to get back to you with 72 hours. Note I may be away for an extended period and may not be able to offer support instantly on some occasions.

### 💙 Discord

You can join the community discord server and react with the assets 🎨 role in the:

`✅ server-info-rules` channel to gain access to support channels for each asset. Like with emails I aim to get back to you within 72 hours, but it may not always be possible.

Join the Carter Games Discord Server!

Check out the Carter Games community on Discord - hang out with 65 other members and enjoy free voice and text chat.

https://carter.games/discord