Audio Manager Documentation

Summary

Contributors

Jonathan Carter

Valid for Version

2.5.0 & Higher

Last Updated

03/05/2021

Contents

Package Information

The package has 5 main folders & 15 files, all listed with an asterisk and coloured **green** are required files for the asset to work. Those without are in red are not needed for the functionality but are required for some cosmetic features.

- **Editor/Carter Games/Audio Manager**
  - ***AudioManagerEditor.cs**
  - ***MusicPlayerEditor.cs**
  - ***AudioPlayerEditor.cs**
- **Gizmos/Carter Games/Assets/Audio Manager**
  - AudioManagerFile Icon.png
- **Prefabs/Carter Games/Audio Manager**
  - ***AudioPrefab.prefab**
- **Recourses/Carter Games/Audio Manager**
  - LogoAM.png
  - Play.png
  - Stop.png
- **Scripts/Carter Games/Audio Manager**
  - ***AudioManagerScript.cs**
  - ***AudioManagerFile.cs**
  - ***AudioRemoval.cs**
  - ***MusicPlayer.cs**
  - ***AudioPlayer.cs**

Change Log: Shows the changes from previous versions of the asset.

Docs: Text file that links to here and provides an offline copy of this page.

First Use Setup

Initial Setup

Once the package has been imported into the project, create a empty gameobject for the manager and add the Audio Manager Script to it or add it to an existing gameobject if you wish. When using the script for the first time it will create a directory “/audio” if it doesn’t already exist, you will need to store the audio in this directory for script to work. The script will also create the directory “/audio/files” which is used as the default location for the script to make an audio manager file if it needs to. Next you will need to assign a prefab to the script, this is used to play the audio when you call a function to play a clip. There is a prefab in the package, which is setup correctly, however you are welcome to make your own and customize it.

Directories

The asset supports multiple directories. If the audio manager file inputted has no directories you will be prompted to add a directory and continue. If there is already a directory you can add additional ones by using the green plus button and remove directories by pressing the red minus button next to the directory you wish to remove. to scan the /audio directory, just have a blank directory field in your list and it will scan the directory.

Scanning

By default the script scans the “/audio” directory if you have left a directory field blank, you can change the path 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 hve 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.

The Interface

Once it scans the set directory it will show each clip in a list but a preview button and the clip name for each clip. Pressing the preview button (the green arrow) plays the clip associated with it in the inspector, please note that the clip will play at the default volume and pitch. You can stop the preview by pressing the stop button next to the clip, which will appear when a clip is been played (the red dash). The names of all the clips can be selected and copied directly from the inspector of the script so you can avoid typo’s when calling the sound to be played.

Extension Scripts

In 2.4.0 we have added additional scripts to the asset. These scripts can only read and do not edit the Audio Manager File when used. Below are the extra scripts and what they do:

Music Player

The music player script allows the playing of background music in games, the script has a fade effect to transition the track in and out and can be edited on the fly.

Audio Player

The Audio Player makes it easier to play audio from an Audio Manager File using Unity Events that are commonly found on UI Buttons. The script has a custom inspector to assign sounds to play on call.

Static Audio Manager

In the 2.5.0 update we have added the option to have the Audio Manager script be static. In the inspector of any Audio Manager script you'll have a button to enable or disable the static instancing. When you press this button you will have to wait for a bit as the Unity Editor updates. Once ready the instance will be enabled or disabled, note that your IDE may not update until you restart Unity.

Using the Manager to play sounds

The first step to this is getting the manager, referencing the script and assigning it is the easiest way, you can also use **FindObjectOfType** or tags to get the **GameObject** the script is on and get the audio manager component. Once you have a reference setup, there are a variety of methods you can use to play the clips.

How To Call Methods

All the following functions have the optional passthrough for changing the volume the clip is played at as well as the pitch the clip is played at. You don't have to enter a value into a method call to use them for instance (Note: both of these example assume your variable reference to the audio manager script is "audioManager"):

```
// This will play MySound at the default volume of 1 and default pitch of 1.
audioManager.PlayClip("MySound");

// This will play MySound at the volume of 0.5 with the default pitch of 1. You get the idea…..
audioManager.PlayClip("MySound", .5f);
```

Audio Manager Methods

As of 2.5.0 we have changes how the methods in the main audio manager script work. You can now use the AudioArgs static class or a Hashtable to define all the params for the method. With this change we have also added a few additional params such as priority and reverb & spatial blend.

```
// Plays the requested clip with optional values for volume and pitch.
// Overload methods are also available for audio mixers, either by
// manual assignment or via the audio manager mixer ID number.
Play(string Request, float Volume = 1f, float Pitch = 1f);
Play(string Request, AudioMixerGroup mixer ,float Volume = 1f, float Pitch = 1f);
Play(string Request, int mixerID, float Volume = 1f, float Pitch = 1f);
Play(string Request, AudioArgs args);


// Play a sound that is scanned into the audio manager and returns the
// audio source for you to check / use as needed.
// Returns: AudioSource
PlayAndGetSource(string request, float volume = 1, float pitch = 1);
PlayAndGetSource(string request, AudioMixerGroup mixer, float volume = 1, float pitch = 1);
PlayAndGetSource(string request, int mixerID, float volume = 1, float pitch = 1);
PlayAndGetSource(string request, AudioArgs args);


// Plays the requested clip with optional values for volume and pitch
// from the set time, handy if the clip doesn't start right away.
// Overload methods are also available for audio mixers like before,
// either by manual assignment or via the audio manager mixer ID number.
PlayFromTime(string Request, float Time, float Volume = 1f, float Pitch = 1f);
PlayFromTime(string Request, float Time, AudioMixerGroup mixer, float Volume = 1f, float Pitch = 1f);
PlayFromTime(string Request, float Time, int mixerID, float Volume = 1f, float Pitch = 1f);
PlayFromTime(string Request, float Time, AudioArgs args);
```

```
// Play a sound from a particular time code on the audio clip audioManagerFile
// and returns the audio source for you to check / use as needed
// Returns: AudioSource
PlayFromTimeAndGetSource(string request, float time, float volume = 1, float pitch = 1);
PlayFromTimeAndGetSource(string request, AudioMixerGroup mixer, float time, float volume = 1, float pi
tch = 1);
PlayFromTimeAndGetSource(string request, int mixerID, float time, float volume = 1, float pitch = 1);
PlayFromTimeAndGetSource(string request, float Time, AudioArgs args);


// Plays the requested clip with optional values for volume and pitch after the entered delay.
// Overload methods are also available for audio mixers like before,
// either by manual assignment or via the audio manager mixer ID number.
PlayWithDelay(string Request, float Delay, float Volume = 1f, float Pitch = 1f);
PlayWithDelay(string Request, float Delay, AudioMixerGroup mixer, float Volume = 1f, float Pitch = 1
f);
PlayWithDelay(string Request, float Delay, int mixerID, float Volume = 1f, float Pitch = 1f);
PlayWithDelay(string Request, float Delay, AudioArgs args);


// Play a sound after a defined amount of time and returns the audio source for you to check / use as
 needed.
// Returns: AudioSource
PlayWithDelayAndGetSource(string request, float delay, float volume = 1f, float pitch = 1f);
PlayWithDelayAndGetSource(string request, AudioMixerGroup mixer,float delay, float volume = 1f, float
 pitch = 1f);
PlayWithDelayAndGetSource(string request, int mixerID, float delay, float volume = 1f, float pitch = 1
f);
PlayWithDelayAndGetSource(string request, float delay, AudioArgs args);


// Plays a random clip with optional values for volume and pitch.
// Overload methods are also available for audio mixers like before,
// either by manual assignment or via the audio manager mixer ID number.
PlayRandom(float Volume = 1f, float Pitch = 1f);
PlayRandom(AudioMixerGroup mixer, float Volume = 1f, float Pitch = 1f);
PlayRandom(int mixerID, float Volume = 1f, float Pitch = 1f);
PlayRandom(int mixerID, AudioArgs args);


// Plays a random clip with optional values for volume and pitch.
// Returns: AudioSource
PlayRandomAndGetSource(float volume = 1f, float pitch = 1f);
PlayRandomAndGetSource(AudioMixerGroup mixer, float volume = 1f, float pitch = 1f);
PlayRandomAndGetSource(int mixerID, float volume = 1f, float pitch = 1f);
PlayRandomAndGetSource(AudioArgs args);


// Plays a random clip with optional values for volume and pitch from the set time.
// Overload methods are also available for audio mixers like before,
// either by manual assignment or via the audio manager mixer ID number.
PlayRandomFromTime(float Time, float Volume = 1f, float Pitch = 1f);
PlayRandomFromTime(float Time, AudioMixerGroup mixer, float Volume = 1f, float Pitch = 1f);
PlayRandomFromTime(float Time, int mixerID, float Volume = 1f, float Pitch = 1f);
PlayRandomFromTime(float Time, AudioArgs args);


// Plays a random clip with optional values for volume and pitch.
// Returns: AudioSource
PlayRandomFromTimeAndGetSource(float time, float volume = 1f, float pitch = 1f);
PlayRandomFromTimeAndGetSource(float time, AudioMixerGroup mixer, float volume = 1f, float pitch = 1
f);
PlayRandomFromTimeAndGetSource(float time, int mixerID, float volume = 1f, float pitch = 1f);
PlayRandomFromTimeAndGetSource(float time, AudioArgs args);
```

```
// Plays a random clip with optional values for volume and pitch after the entered delay.
// Overload methods are also available for audio mixers like before,
// either by manual assignment or via the audio manager mixer ID number.
PlayRandomWithDelay(float Delay, float Volume = 1f, float Pitch = 1f);
PlayRandomWithDelay(float Delay, AudioMixerGroup mixer, float Volume = 1f, float Pitch = 1f);
PlayRandomWithDelay(float Delay, int mixerID, float Volume = 1f, float Pitch = 1f);
PlayRandomWithDelay(float Delay, AudioArgs args);


// Plays a random clip with optional values for volume and pitch.
// Returns: AudioSource
PlayRandomWithDelayAndGetSource(float delay, float volume = 1f, float pitch = 1f);
PlayRandomWithDelayAndGetSource(float delay, AudioMixerGroup mixer, float volume = 1f, float pitch = 1
f);
PlayRandomWithDelayAndGetSource(float delay, int mixerID, float volume = 1f, float pitch = 1f);
PlayRandomWithDelayAndGetSource(float delay, AudioArgs args);
```

Audio Manager Utility Properties/Methods

```
// Method - checks to see if the clip exists in the library.
// Returns: Bool
HasClip(string request);


// Property - Returns the number of clips in the library.
// Returns: int
GetNumberOfClips;


// Property - Gets a random sound from the manager it is called from and returns the result.
// Returns: AudioClip
GetRandomSound;


// Property - changes the audiomanagerfile used on the audiomanager..
// Usage: am.AudioManagerFile = myAMF;
AudioManagerFile;


// Property, Gets the current audiomanagerfile being used and returns it.
// Returns: AudioManagerFile
GetAudioManagerFile();


// Mathod - Checks to see if the request in question is playing
// Only check if it has been played by the audio manager this is called from.
// Returns: Bool
IsClipPlaying(string request);


// STATIC - Audio Arguments for use in the play methods...
// Use is like a paring
// e.g "volume", 1f
AudioArgs(params object[] values)
```

Music Player Properties/Methods

```
// Property - Gets the track currently being played...
GetActiveTrack;


// Property - Gets the position where the track is at...
GetTrackPosition;


// Property - Gets the audio source for the music player...
GetAudioSource;


// Property - Both Get or Set...
// Usage: if (MusicPlayer.Instance.ShouldLoop) // Gets...
// MusicPlayer.Instance.ShouldLoop = false // Sets...
ShouldLoop;


// Plays the music track, Fades In/Out the music based on the bool value passed in.
PlayMusic(bool play);


// Sets the volume of the source to the entered value (should be between 0-1)
SetVolume(float value);


// Changes the track instantly without and fading effects.
// Optional start and end time pass throughs.
ChangeTrackNoFading(AudioClip track, float startTime = 0f, float endTime = 0f)


// Changes the track to the new track entered with a fade effect.
ChangeTrack(AudioClip track)


// Changes the track to the new track entered with a fade effect.
// Added option to change the start time for the new track.
ChangeTrack(AudioClip track, float startTime);


// Changes the track to the new track entered with a fade effect.
// Added option to change the start & end time for the new track.
ChangeTrack(AudioClip track, float startTime, float endTime);
```

Audio Player Method

```
// Plays all of the tracks selected in the inspector with the settings set in the inspector.
Play();
```

Error Messages & Common Problems

The scripts have a selection of error messages in the form of console warnings, all errors from this script come with the prefix “Audio Manager |” so you will know it is from this package. Most errors shouldn’t come up, but they should explain what you’ve done wrong and how to fix it.

- **Warning Code 1:** Make sure you have a sound prefab assigned to the AMF you are using, this is caused when there is not prefab found.
- **Warning Code 2:** Make sure you have spelt the clip you want correctly, this warning shows up if the audio could not be found in the audio manager when called.
- **Warning Code 3:** Could not find audio mixer, Please ensure the mixer is in the inspector and you have the right ID.
- **Warning Code 4:** No AudioSource Component found on the Sound Prefab. Please ensure a AudioSouce Component is attached to your prefab.

However, if you run into a problem or get an error and are unsure, feel free to drop us an email at (hello@carter.games) and we’ll do my best to help you out.

Common Problems

The manager can find my audio

Please make sure all audio you want to use with this manager is in the /audio directory or in the defined sub-directories you are trying to scan. If this is a feature you require, please do let us know!

I called a function to play audio and nothing happened

Please make sure you spelt the clip name correctly, note it is CaSe SeNsItIvE, also make sure the code is running with a debug log and the script is references correctly.

I called a function and the clip plays a million times!!!

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

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