Runtime AudioClip Loader

Documentation

v3.0

Overview

Runtime AudioClip Loader allows you to load **MP3, WAV, Ogg, AIFF** into Unity's AudioClip from file or internet on many platforms.

Audio files are loaded into Unity's AudioClip, thus audio mixers, spatial audio and AudioSource are fully supported.

Demo: WebGL | Windows IL2CPP | Windows Mono

Support | Documentation

Includes documented example scenes and scripts.

All platforms are theoretically supported (fully managed libraries as fallback)

Tested with

- Windows
- WebGL
- IL2CPP
- Compatible with **Unity 2020.3 – 2024**

Possible uses

- Play any song from user's computer
- Download and play all 4 formats from internet
- Add support for modding into your game, as your players can easily replace sounds
- Put your sounds outside Unity project to get theoretically faster build times

This is a scripts only asset, you are expected to know how to code in C#.

This asset uses NAudio, NVorbis and NLayer all of which are under MIT License. See Third-Party Notices.txt file in package for details.

Getting started

Project contains Examples folder which contains various demonstrations where you can see and learn intended usage.

Example: Simplest load from disk

Demonstrates how to load and play audio file from drive. RuntimeAudioclipLoader tries all available decoders it knows to decode your file, so you don't have specify worry about format. On Windows all 4 format are fully supported other platforms have some limitations (see Format and platform support)

```
    // Demonstrates how to load and play audio file from drive
    [RequireComponent(typeof(AudioSource))]
    public class SimplestLoadFromDisk : MonoBehaviour
    {
        // File path to load and play
        public string path = @"RuntimeAudioClipLoader/explanation_mp3.mp3";
        // UI debug message
        string uiMessage;
        // AudioSource that will play loaded AudioClip
        AudioSource audioSource;

        // Start is called on the frame when a script is enabled just before any of the Update methods are
called the first time
        void Start()
        {
            // Cache our AudioSource component
            audioSource = GetComponent<AudioSource>();
            StartCoroutine(Load());
        }

        // OnGUI is called for rendering and handling GUI events
        void OnGUI()
        {
            // Update debug message if currently playing AudioClip
            if (audioSource.isPlaying)
                uiMessage = "Playing " + (int)audioSource.time + "/" + (int)audioSource.clip.length + "s";
            // Show debug message
            GUILayout.Label(uiMessage);
        }

        // Coroutine used to start and wait for AudioClipLoader load process
        IEnumerator Load()
        {
```

```
            // Make config instance
            var config = new AudioLoaderConfig();
            // Specify we want to load from file stream
            config.DataStream = File.OpenRead(Path.Combine(Application.streamingAssetsPath, path));
            // New loader instance and pass in config
            var loader = new AudioLoader(config);
            // Initiate loading process
            loader.StartLoading();
            // Wait until audio clip finishes loading
            while (loader.LoadState == AudioDataLoadState.Loading)
            {
                uiMessage = "Loading " + (int)(loader.LoadProgress * 100) + "%";
                yield return new WaitForEndOfFrame();
            }
            // If succesfully loaded, play AudioClip
            if (loader.LoadState == AudioDataLoadState.Loaded)
            {
                audioSource.clip = loader.AudioClip;
                audioSource.Play();
            }
            // Otherwise show error
            else if (loader.LoadState == AudioDataLoadState.Failed)
            {
                uiMessage = "Failed";
            }
        }
    }
```

Example: Simplest load from internet

Demonstrates how to download and play file from internet. RuntimeAudioclipLoader only cares about the bytes/data in System.IO.Stream, it doesn't matter where you got it from.

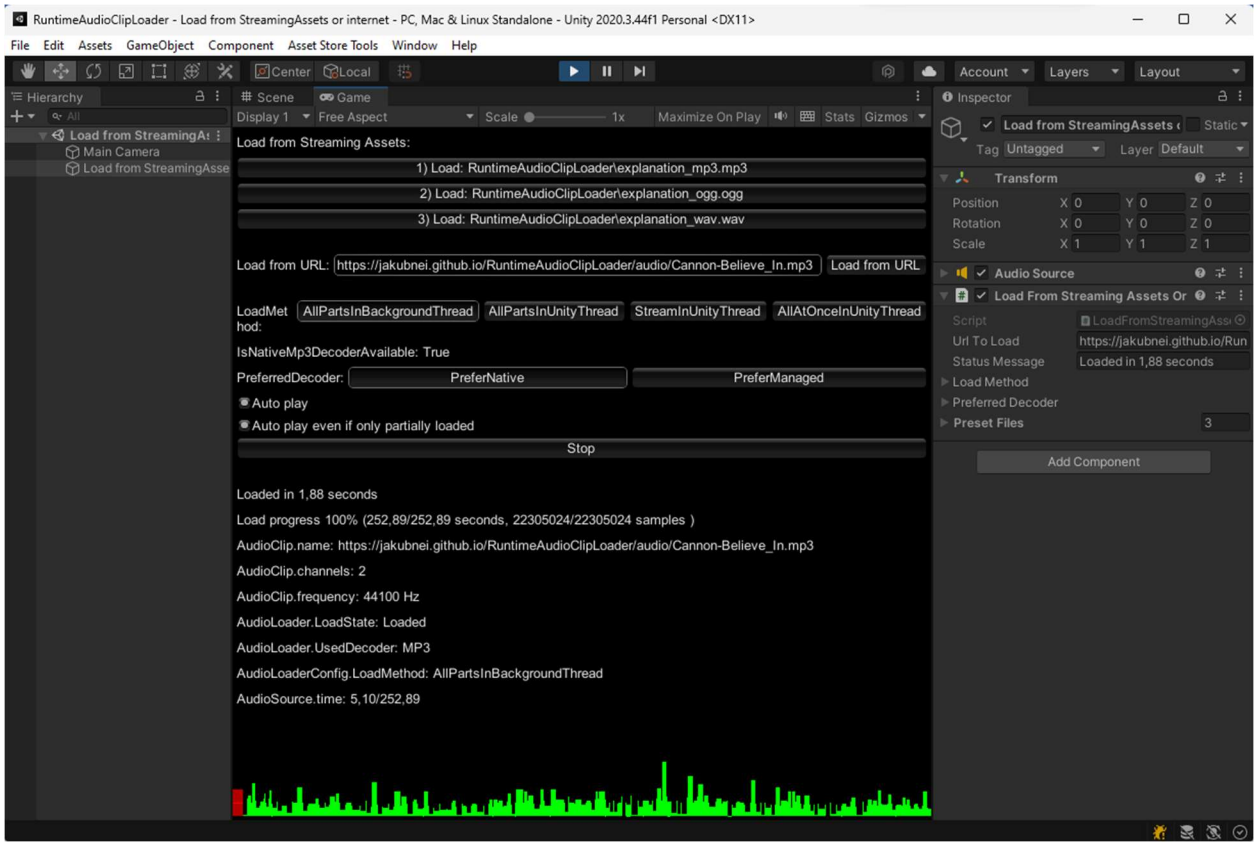
```
using System.Collections;
using System.IO;
using RuntimeAudioClipLoader;
using UnityEngine;

namespace RuntimeAudioClipLoaderDemo
{
    [RequireComponent(typeof(AudioSource))]
    public class SimplestLoadFromInternet : MonoBehaviour
    {
        public string urlToLoad =
@"https://jakubnei.github.io/RuntimeAudioClipLoader/audio/Cannon-Believe_In.mp3";
```

```
        public string uiMessage;

        void Start()
        {
            StartCoroutine(Load());
        }

        void OnGUI()
        {
            GUILayout.Label(uiMessage);

            var audioSource = GetComponent<AudioSource>();
            if (audioSource.isPlaying)
                uiMessage = "Playing " + (int)audioSource.time + "/" +
(int)audioSource.clip.length + " s";
        }

        IEnumerator Load()
        {
            byte[] bytes = null;

            using (var webRequest =
UnityEngine.Networking.UnityWebRequest.Get(urlToLoad))
            {
                webRequest.SendWebRequest();
                while (!webRequest.isDone)
                {
                    uiMessage = "Downloading " +
(int)(webRequest.downloadProgress * 100) + "%";
                    yield return new WaitForEndOfFrame();
                }

                if (webRequest.isNetworkError)
                {
                    uiMessage = "Error: " + webRequest.error;
                    yield break;
                }
                else
                {
                    bytes = webRequest.downloadHandler.data;
                }
            }

            var config = new AudioLoaderConfig();
```

```
            config.DataStream = new MemoryStream(bytes);

            var loader = new AudioLoader(config);
            loader.StartLoading();
            while (!loader.IsLoadingDone)
            {
                uiMessage = "Decoding " + (int)(loader.LoadProgress * 100) + "%";
                yield return new WaitForEndOfFrame();
            }

            var audioSource = GetComponent<AudioSource>();
            audioSource.clip = loader.AudioClip;
            uiMessage = "Playing";
            audioSource.Play();
        }

    }
}
```

Example: Load from StreamingAssets or internet

More complicated comprehensive example, intended to demonstrate most features. GUI allows you to test different settings for Load Methods and Native/Managed decoders.


RuntimeAudioClipLoader - Load from StreamingAssets or internet - PC, Mac & Linux Standalone - Unity 2020.3.44f1 Personal <DX11>
File Edit Assets GameObject Component Asset Store Tools Window Help
Hierarchy
Load from StreamingAs
Main Camera
Load from StreamingAsse
Scene
Game
Display 1
Free Aspect
Scale
1x
Maximize On Play
Stats
Gizmos
Load from Streaming Assets:
1) Load: RuntimeAudioClipLoader\explanation_mp3.mp3
2) Load: RuntimeAudioClipLoader\explanation_ogg.ogg
3) Load: RuntimeAudioClipLoader\explanation_wav.wav
Load from URL:
https://jakubnei.github.io/RuntimeAudioClipLoader/audio/Cannon-Believe_In.mp3
Load from URL
LoadMethod:
AllPartsInBackgroundThread
AllPartsInUnityThread
StreamInUnityThread
AllAtOnceInUnityThread
IsNativeMp3DecoderAvailable: True
PreferredDecoder:
PreferNative
PreferManaged
Auto play
Auto play even if only partially loaded
Stop
Loaded in 1,88 seconds
Load progress 100% (252,89/252,89 seconds, 22305024/22305024 samples)
AudioClip.name: https://jakubnei.github.io/RuntimeAudioClipLoader/audio/Cannon-Believe_In.mp3
AudioClip.channels: 2
AudioClip.frequency: 44100 Hz
AudioLoader.LoadState: Loaded
AudioLoader.UsedDecoder: MP3
AudioLoaderConfig.LoadMethod: AllPartsInBackgroundThread
AudioSource.time: 5,10/252,89
Inspector
Load from StreamingAssets
Static
Tag Untagged
Layer Default
Transform
Position
Rotation
Scale
Audio Source
Load From Streaming Assets Or
Script
Url To Load
https://jakubnei.github.io/Run
Status Message
Loaded in 1,88 seconds
Load Method
Preferred Decoder
Preset Files
3
Add Component

More details

Before shipping your game It's better to make sure all your audio files are loadable. If some are not, you might be able to fix it by encoding them with different settings.

Remember to Destroy(audioClip); once you are done using it, otherwise you risk running out of memory.

WAV and AIFF are loadable on any platform if they are not compressed at all (only simple Pcm or IeeeFloat formats). If they are compressed Window's ACM is used.

Ogg is loadable only if it uses Xiph.org Ogg Skeleton container, to make sure that is the case you may use free audio software such as https://www.audacityteam.org/ or some online convert tool such as https://audio.online-convert.com/

Unity does not support Streaming AudioClips in WebGL, they explain it here https://docs.unity3d.com/Manual/webgl-audio.html In WebGL, only LoadMethod.AllPartsInUnityThread is supported, it automatically reverts to it if it detects it's running inside WebGL. In WebGL, the AudioClip is playable only once it's fully loaded, if you started playing it during loading you will not hear anything, you need to stop and start the AudioSource again.

If you have any issues, feel free to contact me at nei.jakub@gmail.com It would be nice to send a link to anything you have created with this asset.

You can use either recommended OOP approach with instances of RuntimeAudioClipLoader.AudioLoader or legacy static manager class RuntimeAudioClipLoader.Manager.

RuntimeAudioClipLoader.Manager is kept for backwards compatibility and internally uses instances of RuntimeAudioClipLoader.AudioLoader.

When Unity can't use native Windows decoder it fall backs to managed decoders which come with some limitations as described in table bellow.

Format and platform support

format	*this asset (Windows)*	*this asset (other platforms)*	*Unity (with WWW.GetAudioClip)*
MP3	full support	only 44.1kHz stereo	phones
WAV	full support	only if no compression is used	phones
Ogg	full support	only Xiph.org Ogg Skeleton container	web, standalones
AIFF	full support	only if no compression is used	no

LoadMethod enum

Based on your needs you can change loading method.

```
    /// <summary>
    /// Where and how to load audio data.
    /// By default, the best available is chosen for you.
    /// </summary>
    public enum LoadMethod
    {
        /// <summary>
        /// Splits loading into parts.
        /// AudioClip can be played after the first part is loaded.
        /// All parts are loaded in background thread.
        /// Does not slow main Unity thread.
        /// Slowest but does not cause any FPS hitch or drop at all.
        /// </summary>
        AllPartsInBackgroundThread,
        /// <summary>
        /// Splits loading into parts.
        /// AudioClip can be played after the first part is loaded.
        /// All parts are loaded in main Unity thread in coroutine.
        /// Maximum Time spent in main Unity thread is limited by <see
cref="AudioLoaderConfig.LoadInUnityThreadMaximumMilisecondsSpentPerFrame"/>.
        /// </summary>
        AllPartsInUnityThread,
        /// <summary>
        /// Decode on demand in main Unity thread.
        /// Ideal for long audio clips you will use only once.
        /// </summary>
        StreamInUnityThread,
        /// <summary>
        /// Worst option, supported because this is only possible method on WebGL
platform.
        /// Main Unity thread freezes until whole AudioClip is loaded.
        /// Everything is loaded at once, right after you call <see
cref="AudioLoader.StartLoading"/>, which in this case must be called from Unity
thread.
        /// </summary>
        AllAtOnceInUnityThread,
    }
```

Third party libraries

This asset includes third party compiled libraries NAudio, NVorbis and NLayer:

NAudio

“Open source .NET audio library written by Mark Heath.”

https://github.com/naudio/NAudio

- supports WAV, MP3, WAV and AIFF formats
- uses interop with Windows's Multimedia Extensions (MME) and Audio Compression Manager (ACM)
- uses ACM for WAV and AIFF if they are compressed.
- uses MME for MP3 decoding.

Which means same formats may only be loadable on Windows

NVorbis

“Net library for decoding Xiph.org Vorbis files”

https://github.com/NVorbis/NVorbis

- adds managed Ogg support (through fully managed C# code).

NLayer

“Fully managed MP3 to WAV decoder.”

https://github.com/naudio/NLayer

- adds managed MP3 support (through fully managed C# code).
- is able to load only stereo MP3s with 44.1kHz sample rate.

Summary

NVorbis and NLayer are decoders only, it's not possible to encode OGG and MP3 with them.

By default, based on UnityEngine.Application.platform it is automatically decided whether to use NAudio or NLayer for MP3 decoding.

NLayer fully managed C# MP3 decoder failed on 55 out of 1552 test cases and was on average 7.2 times slower than Windows interop MP3 decoder. That is why **it’s recommended to test all your audio files on target platform before shipping. If some audio files are not possible to load, you might be able to fix it by encoding them with different settings.**

Changelog

v3.0

- Updated NAudio and NLayer to latest versions
- NLayer changed license to MIT
- Added example: simplest load from disk
- Added LoadMethod.AllAtOnceInUnityThread
- Fixed issue where WebGL creates AudioClip with less samples than provided
- Fixed few issues in Mac OS X build
- Fixed LoadMethod config
- Dropped older Unity support

v2.0

- Added NLayer for multiplatform MP3 loading
- Added object oriented approach
- Added example: simplest load from internet
- Improved code
- Improved documentation
- Decreased memory consumption
- Fixed various bugs

v1.1

- Added example: load and play from internet
- Added compatibility for older Unity versions

v1.0

- First release