Estrada is a drop-in replacement for Unity Microphone with WebGL support.

Check out the online documentation and demo.

For all questions write to bluezzzy.dev@gmail.com.

Features:

- Full compatibility with Unity Microphone API
- One line of code to switch from Unity Microphone to Estrada
- WebGL support (with the limitations)
- Uniform microphone permission handling for all platforms
- Automatic browser feature detection
- Uses AudioWorkletNode in HTTPS mode and falls back to ScriptProcessorNode for local development

Supported Unity editors:

- 2021.2 and above

Known issues

AudioClip cannot be read in WebGL

Symptom:

> (Error in browser console) Cannot get data on compressed samples for audio clip "*clipName*". Changing the load type to DecompressOnLoad on the audio clip will fix this.

Fix: Update your version of Unity to one of the following versions

Limitations

WebGL:

- Application using Estrada must run in an HTTPS context (HTTP is allowed for local development)
- Unity doesn't support *AudioClip.GetData()*. Use *Estrada.Microphone.GetCurrentData()* instead
- Unity doesn't support *AudioClip* streaming, simultaneous audio recording and playback is not supported
- Microphone permission is required to view available devices
- Unity AudioClip supports only 44100 Hz sample rate. Estrada detects this limitation and limits the sample rates of the recording devices to this value. If you want to have access to the full range of sample rates of the recording device, define ESTRADA_WEBGL_ALLOW_SAMPLE_RATE_MISMATCH symbol (Project Settings -> Player -> Other Settings -> Scripting Define Symbols). In this case, AudioClip.frequency will still equal 44100 Hz and AudioClip.samples may not equal the product of the specified sample rate and duration, but AudioClip will store samples for the sample rate specified when calling Estrada.Microphone.Start().

Getting started

How to replace Unity Microphone with Estrada

To use Estrada, add

```
using Microphone = Estrada.Microphone;
```

to your script file.

Example

```
using System.Collections;
using UnityEngine;

// Add this line to your code to replace Unity Microphone with Estrada Microphone
using Microphone = Estrada.Microphone;

public class EstradaExample : MonoBehaviour
{
   private IEnumerator Start()
   {
      // Get permission
      if (Microphone.RequiresPermission())
      {
         yield return Microphone.RequestPermission();

         if (!Microphone.HasPermission())
         {
            Debug.LogError("Permission to record a microphone is not granted");
            yield break;
         }
      }

      // Start microphone and connect it to AudioSource
      const string deviceName = null;
      const int length = 5;
      const int frequency = 16000;
      const bool loop = false;

      var audioSource = gameObject.AddComponent<AudioSource>();

      audioSource.clip = Microphone.Start(deviceName, loop, length, frequency);
      audioSource.loop = loop;

      // Wait for microphone to start
      while (Microphone.GetPosition(deviceName) == 0)
      {
         yield return null;
      }

      // Wait until recorded
      while (Microphone.IsRecording(deviceName))
      {
         yield return null;
      }

      // Play recorded audio
      audioSource.Play();
   }
}
```

Namespace Estrada

Classes

Microphone

Class Microphone

Inheritance

System.Object

Microphone

Namespace: Estrada

Assembly: Estrada.dll

Syntax

```
public static class Microphone
```

Properties

devices

Returns names of available devices.

Declaration

```
public static string[] devices { get; }
```

Property Value

TYPE	DESCRIPTION
System.String[]	

Methods

End(String)

Stops recording

Declaration

```
public static void End(string deviceName)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	deviceName	The name of the device.

GetCurrentData(Single[], Int32)

Fills an array with sample data from the clip. If not recording, samples from previous clip will be used.

Declaration

```
public static bool GetCurrentData(float[] data, int offsetSamples)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Single[]	data	Array to fill.

TYPE	NAME	DESCRIPTION
System.Int32	offsetSamples	Internal buffer offset.

Returns

TYPE	DESCRIPTION
System.Boolean	Whether the microphone data was accessed.

GetDeviceCaps(String, out Int32, out Int32)

Get the frequency capabilities of a device.

Declaration

```
public static void GetDeviceCaps(string deviceName, out int minFreq, out int maxFreq)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	deviceName	Device name. Null or empty string means default device.
System.Int32	minFreq	Minimal supported sample rate.
System.Int32	maxFreq	Maximal supported sample rate.

GetPosition(String)

Get current position of the recording.

Declaration

```
public static int GetPosition(string deviceName)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	deviceName	

Returns

TYPE	DESCRIPTION
System.Int32	Current recording position in samples.

HasPermission()

Whether a permission to record microphone has been granted.

Declaration

```
public static bool HasPermission()
```

Returns

TYPE	DESCRIPTION
System.Boolean	Microphone access state.

IsRecording(String)

Whether recording is active.

Declaration

```
public static bool IsRecording(string deviceName)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	deviceName	

Returns

TYPE	DESCRIPTION
System.Boolean	Recording state.

RequestPermission()

Request permission to record microphone.

Declaration

```
public static IEnumerator RequestPermission()
```

Returns

TYPE	DESCRIPTION
System.Collections.IEnumerator	Enumerator to run coroutine on.

RequiresPermission()

Whether a microphone permission is required

Declaration

```
public static bool RequiresPermission()
```

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Boolean	True if required.

Start(String, Boolean, Int32, Int32)

Start recording.

Declaration

```
public static AudioClip Start(string deviceName, bool loop, int lengthSec, int frequency)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	deviceName	Name of the device to be recorded. Null or empty string means default device.
System.Boolean	loop	Indicates whether the recording should continue recording if lengthSec is reached, and wrap around and record from the beginning of the AudioClip.
System.Int32	lengthSec	Length of the recording before overlap.
System.Int32	frequency	The sample rate of the AudioClip produced by the recording.

Returns

TYPE	DESCRIPTION
UnityEngine.AudioClip	Returns AudioClip or null if the recording fails to start.

Exceptions

TYPE	CONDITION
System.InvalidOperationException	If permission to record was not granted or no devices available.
System.ArgumentException	• If `deviceName` is not in the devices. • If `lengthSec` is negative or is greater than hour. • If `frequency` is negative.