Record Audio in .NET Core

Asked 4 years, 7 months ago Modified 3 months ago Viewed 8k times

15

As the title suggests, I'm looking to capture audio from a microphone in .NET Core. It's important for me that it's cross platform. I'm doing a lot of video processing in OpenCV on Windows, Linux and OSX already. Audio is a missing piece of the puzzle for me.

.net audio .net-core cross-platform

Share Follow

asked Sep 21, 2018 at 13:24

Wojtek Turowicz
4,086 ● 3 ● 27 ● 38

OpenCV is a library that eventually calls OS-specific APIs. To use it from .NET you'd have to use interop or a managed wrapper. You should probably look for managed wrappers for the audio library you want to use. – Panagiotis Kanavos Sep 25, 2018 at 8:58

@PanagiotisKanavos I did find OpenTK.NETCore that does cross platform audio. – Wojtek Turowicz Jan 17, 2019 at 15:40

OpenTK.NetCore has been deprecated in favour of OpenTK.NetStandard – bre_dev Sep 20, 2020 at 13:44

4 Answers

Sorted by: Highest score (default)

4

OpenTK.NETCore is a really good cross platform port for OpenTK.

Using it's `AudioCapture` class I have been able to capture microphone data across different OS platforms.

```
using (var audioCapture = new AudioCapture(_recorders.First(), _sampling_rate,
ALFormat.Mono16, buffer_length_samples))
{
    audioCapture.Start();

    int available_samples = audioCapture.AvailableSamples;

    _buffer = _pool.Rent(MathHelper.NextPowerOfTwo((int)((available_samples *
_sampleToByte / (double)BlittableValueType.StrideOf(_buffer)) + 0.5)));

    if (available_samples > 0)
    {
        audioCapture.ReadSamples(_buffer, available_samples);
    }
    else
    {
        _pool.Return(_buffer);
    }
}
```

Edit: Based on `Parrot` project in OpenTK samples.

Share Follow

answered Oct 2, 2018 at 13:13

Wojtek Turowicz
4,086 ● 3 ● 27 ● 38

please send Parrot project in OpenTK samples url! – Ali Rasouli Aug 2, 2020 at 8:34

I think he is referring to these examples: github.com/mono/opentk/tree/master/Source/Examples/OpenAL/1.1 – Kappacake Aug 28, 2020 at 10:08

After spending some time reading and running the Parrot sample, I realized that it just plays back the audio as it captures it. How would you save that audio to a file? Have you implemented this functionality? – Kappacake Aug 28, 2020 at 10:44

How to save the captured audio in a wav file? Any suggestion about how to save it? – bre_dev Sep 17, 2020 at 20:23

To save it as WAV, check out this answer: stackoverflow.com/a/63962284/1204153 – Andy Sep 18, 2020 at 20:22

0

I don't know an API or library that does that for you, unfortunately.

If others also don't know, then you have two options:

1. Write a cross-platform api yourself (take a look at the CoreFx or CoreFxLab repos), on Windows you might use P/Invoke to use system dlls, on the other platforms you'll need something else
2. Take a cross-platform recorder executable and interface with that from your .net core functionality

Share Follow

answered Sep 25, 2018 at 6:35

hansmbakker
1,089 • 14 • 29

1 Thanks I've already managed with OpenTK.NETCore library and it's OpenAL support. – Wojtek Turowicz Oct 2, 2018 at 13:08

0

I've written a recorder app before on .NET Core, but its not my property anymore... However, here's the library that I used while working on it.

LibsoundIO Sharp is able to extend configurations to things such as the sampling rate of the Input or output and has sweet examples like that:

```
class Record
    {
        static SoundIORingBuffer ring_buffer = null;

        static SoundIOFormat [] prioritized_formats = {
            SoundIODevice.Float32NE,
            SoundIODevice.Float32FE,
            SoundIODevice.S32NE,
            SoundIODevice.S32FE,
            SoundIODevice.S24NE,
            SoundIODevice.S24FE,
            SoundIODevice.S16NE,
            SoundIODevice.S16FE,
            SoundIODevice.Float64NE,
            SoundIODevice.Float64FE,
            SoundIODevice.U32NE,
            SoundIODevice.U32FE,
            SoundIODevice.U24NE,
            SoundIODevice.U24FE,
            SoundIODevice.U16NE,
            SoundIODevice.U16FE,
            SoundIOFormat.S8,
            SoundIOFormat.U8,
            SoundIOFormat.Invalid,
        };

        static readonly int [] prioritized_sample_rates = {
            48000,
            44100,
            96000,
            24000,
            0,
        };

        public static int Main (string [] args)
```

```
        {
            string device_id = null;
            string backend_name = null;
            bool raw = false;
            string outfile = null;

            foreach (var arg in args) {
                switch (arg) {
                case "--raw":
                    raw = true;
                    continue;
                default:
                    if (arg.StartsWith ("--backend:"))
                        backend_name = arg.Substring (arg.IndexOf (':') + 1);
                    else if (arg.StartsWith ("--device:"))
                        device_id = arg.Substring (arg.IndexOf (':') + 1);
                    else
                        outfile = arg;
                    continue;
                }
            }

            var api = new SoundIO ();

            var backend = backend_name == null ? SoundIOBackend.None :
(SoundIOBackend)Enum.Parse (typeof (SoundIOBackend), backend_name);
            if (backend == SoundIOBackend.None)
                api.Connect ();
            else
                api.ConnectBackend (backend);
            Console.WriteLine ("backend: " + api.CurrentBackend);

            api.FlushEvents ();

            var device = device_id == null ? api.GetInputDevice
(api.DefaultInputDeviceIndex) :
                Enumerable.Range (0, api.InputDeviceCount)
                .Select (i => api.GetInputDevice (i))
                .FirstOrDefault (d => d.Id == device_id && d.IsRaw == raw);
            if (device == null) {
                Console.Error.WriteLine ("device " + device_id + " not found.");
                return 1;
            }
            Console.WriteLine ("device: " + device.Name);
            if (device.ProbeError != 0) {
                Console.Error.WriteLine ("Cannot probe device " + device_id + ".");
                return 1;
            }

            var sample_rate = prioritized_sample_rates.First (sr =>
device.SupportsSampleRate (sr));

            var fmt = prioritized_formats.First (f => device.SupportsFormat (f));

            var instream = device.CreateInStream ();
            instream.Format = fmt;
            instream.SampleRate = sample_rate;
            instream.ReadCallback = (fmin, fmax) => read_callback (instream, fmin,
fmax);
            instream.OverflowCallback = () => overflow_callback (instream);

            instream.Open ();

            const int ring_buffer_duration_seconds = 30;
            int capacity = (int)(ring_buffer_duration_seconds * instream.SampleRate *
instream.BytesPerFrame);
            ring_buffer = api.CreateRingBuffer (capacity);
            var buf = ring_buffer.WritePointer;

            instream.Start ();

            Console.WriteLine ("Type CTRL+C to quit by killing process...");
            using (var fs = File.OpenWrite (outfile)) {
                var arr = new byte [capacity];
                unsafe {
                    fixed (void* arrptr = arr) {
                        for (; ; ) {
                            api.FlushEvents ();
```

```
                            Thread.Sleep (1000);
                            int fill_bytes = ring_buffer.FillCount;
                            var read_buf = ring_buffer.ReadPointer;

                            Buffer.MemoryCopy ((void*)read_buf, arrptr, fill_bytes,
fill_bytes);
                            fs.Write (arr, 0, fill_bytes);
                            ring_buffer.AdvanceReadPointer (fill_bytes);
                        }
                    }
                }
            }
            instream.Dispose ();
            device.RemoveReference ();
            api.Dispose ();
            return 0;
        }

        static void read_callback (SoundIOInStream instream, int frame_count_min, int
frame_count_max)
        {
            var write_ptr = ring_buffer.WritePointer;
            int free_bytes = ring_buffer.FreeCount;
            int free_count = free_bytes / instream.BytesPerFrame;

            if (frame_count_min > free_count)
                throw new InvalidOperationException ("ring buffer overflow"); //
panic()

            int write_frames = Math.Min (free_count, frame_count_max);
            int frames_left = write_frames;

            for (; ; ) {
                int frame_count = frames_left;

                var areas = instream.BeginRead (ref frame_count);

                if (frame_count == 0)
                    break;

                if (areas.IsEmpty) {
                    // Due to an overflow there is a hole. Fill the ring buffer with
                    // silence for the size of the hole.
                    for (int i = 0; i < frame_count * instream.BytesPerFrame; i++)
                        Marshal.WriteByte (write_ptr + i, 0);
                    Console.Error.WriteLine ("Dropped {0} frames due to internal
overflow", frame_count);
                } else {
                    for (int frame = 0; frame < frame_count; frame += 1) {
                        int chCount = instream.Layout.ChannelCount;
                        int copySize = instream.BytesPerSample;
                        unsafe {
                            for (int ch = 0; ch < chCount; ch += 1) {
                                var area = areas.GetArea (ch);
                                Buffer.MemoryCopy ((void*)area.Pointer,
(void*)write_ptr, copySize, copySize);
                                area.Pointer += area.Step;
                                write_ptr += copySize;
                            }
                        }
                    }
                }

                instream.EndRead ();

                frames_left -= frame_count;
                if (frames_left <= 0)
                    break;
            }

            int advance_bytes = write_frames * instream.BytesPerFrame;
            ring_buffer.AdvanceWritePointer (advance_bytes);
        }

        static int overflow_callback_count = 0;
        static void overflow_callback (SoundIOInStream instream)
        {
            Console.Error.WriteLine ("overflow {0}", overflow_callback_count++);
```

```
        }
    }
```

[Adapted from sio-record/Program.cs]

As far as .NET Core is concerned, you can leave it in a singleton class as a standalone project or if you're working on a project that has a DI container, simply toss it in as a singleton.

That being said, you need [libsoundio] to make things work because afterall, its like a "java" for audio and libsoundio-sharp is the wrapper for C#. Cheers!

Share Follow

edited Sep 26, 2018 at 10:15

answered Sep 26, 2018 at 2:58

Nicholas
1,873 ● 21 ● 39

Thanks I've already managed with OpenTK.NETCore library and it's OpenAL support. – Wojtek Turowicz Oct 2, 2018 at 13:08

0

I wrote a simple program that prints the current volume using the OpenTK.OpenAL nuGet-package.

```
using OpenTK.Audio.OpenAL;
using System;

Console.WriteLine("Default Mic:\n"+ ALC.GetString(ALDevice.Null,
AlcGetString.CaptureDefaultDeviceSpecifier));
Console.WriteLine("Mic List:\n"+ string.Join("\n", ALC.GetString(ALDevice.Null,
AlcGetStringList.CaptureDeviceSpecifier)));

int bufferLength = 10 * 16000;//10 sec
ALCaptureDevice mic= ALC.CaptureOpenDevice(null,16000,ALFormat.Mono8,
bufferLength);//opens default mic //null specifies default
Console.WriteLine("Using:");
Console.WriteLine(ALC.GetString(new ALDevice(mic.Handle),
AlcGetString.DeviceSpecifier));

ALC.CaptureStart(mic);
byte[] buffer = new byte[bufferLength];
for (int i = 0; i < 1000; ++i)
{
    Thread.Sleep(100);
    int samplesAvailable = ALC.GetAvailableSamples(mic);
    ALC.CaptureSamples(mic, buffer, samplesAvailable);

    if(samplesAvailable>0)
        //Console.WriteLine(new string('|', (buffer[..samplesAvailable].Max())*80/256
));
        Console.WriteLine(new string('|', (buffer[..samplesAvailable].Select(x =>
Math.Abs((int)(x)-128)).Max()) * 80/ 128));
}
ALC.CaptureStop(mic);

ALC.CaptureCloseDevice(mic);
```

Share Follow

edited Feb 2 at 22:07

answered Feb 2 at 21:17

K1521
11 ● 3