Sound Code

a place for me to talk about programming and .NET ... because my friends and family aren't interested

THURSDAY, 8 OCTOBER 2009

Playback of Sine Wave in NAudio

In this post I will demonstrate how to create and play a sine wave using NAudio. To do this, we need to create a derived WaveStream, or more simply, a class that implements IWaveProvider.
One awkwardness of implementing IWaveProvider or WaveStream, is the need to provide the data into a byte array, when it would be much easier to write to an array of floats for 32 bit audio (or shorts for 16 bit). To help with this, I have created the **WaveProvider32** class (likely to be committed into the source for NAudio 1.3), which uses the magic of the WaveBuffer class, to allow us to cast the target byte array into an array of floats.

```
public abstract class WaveProvider32 : IWaveProvider
{
    private WaveFormat waveFormat;

    public WaveProvider32()
        : this(44100, 1)
    {
    }

    public WaveProvider32(int sampleRate, int channels)
    {
        SetWaveFormat(sampleRate, channels);
    }

    public void SetWaveFormat(int sampleRate, int channels)
    {
        this.waveFormat = WaveFormat.CreateIeeeFloatWaveFormat(sample
    }

    public int Read(byte[] buffer, int offset, int count)
    {
        WaveBuffer waveBuffer = new WaveBuffer(buffer);
        int samplesRequired = count / 4;
        int samplesRead = Read(waveBuffer.FloatBuffer, offset / 4, sa
        return samplesRead * 4;
    }

    public abstract int Read(float[] buffer, int offset, int sampleCo

    public WaveFormat WaveFormat
    {
        get { return waveFormat; }
    }
}
```

Keep me updated

email address

Subscribe

ABOUT ME

Mark Heath
I'm a software developer, based in Southampton, UK specialising in .NET, trying to keep up with best practices and latest technologies. In my spare time, I maintain a variety of open source .NET applications, and this blog is for sharing things I discover along the way. Read more....

MY PLURALSIGHT COURSES

- More Effective LINQ
- Building Serverless Applications in Azure
- Azure CLI: Getting Started
- Azure Functions Fundamentals
- Creating Modern WPF Applications with MahApps.Metro
- Windows Forms Best Practices
- Understanding and Eliminating Technical Debt
- Digital Audio Fundamentals
- Audio Programming with NAudio
- Understanding Distributed Version Control Systems
- ClickOnce Deployment Fundamentals

MY SOFTWARE

- NAudio .NET Audio Library
- Skype Voice Changer
- .NET Voice Recorder
- TypeScript Tetris
- Silverlight Audio Player
- NLayer Managed MP3 Decoder

Now we can derive from WaveProvider32 to supply our actual sine wave data:

```
public class SineWaveProvider32 : WaveProvider32
{
    int sample;

    public SineWaveProvider32()
    {
        Frequency = 1000;
        Amplitude = 0.25f; // let's not hurt our ears
    }

    public float Frequency { get; set; }
    public float Amplitude { get; set; }

    public override int Read(float[] buffer, int offset, int sampleCo
    {
        int sampleRate = WaveFormat.SampleRate;
        for (int n = 0; n < sampleCount; n++)
        {
            buffer[n+offset] = (float)(Amplitude * Math.Sin((2 * Math
            sample++;
            if (sample >= sampleRate) sample = 0;
        }
        return sampleCount;
    }
}
```

Using it is straightforward. We choose our output sample rate, the frequency of the sine wave and amplitude. You can even adjust them in real-time.

```
private WaveOut waveOut;

private void button1_Click(object sender, EventArgs e)
{
    StartStopSineWave();
}

private void StartStopSineWave()
{
    if (waveOut == null)
    {
        var sineWaveProvider = new SineWaveProvider32();
        sineWaveProvider.SetWaveFormat(16000, 1); // 16kHz mono
        sineWaveProvider.Frequency = 1000;
        sineWaveProvider.Amplitude = 0.25f;
        waveOut = new WaveOut();
        waveOut.Init(sineWaveProvider);
        waveOut.Play();
    }
    else
    {
        waveOut.Stop();
        waveOut.Dispose();
        waveOut = null;
    }
}
```

Please note, that you will need to be using the latest NAudio code from source

- SilverNibbles Silverlight Snake Game
- MIDI File Mapper
- MIDI File Splitter
- Asterisk.NET retro game
- Learning Games

SUBSCRIBE TO

Posts

Comments

SEARCH THIS BLOG

Search

BLOG ARCHIVE

▶ 2016 (2)
▶ 2015 (15)
▶ 2014 (50)
▶ 2013 (19)
▶ 2012 (30)
▶ 2011 (35)
▶ 2010 (29)
▼ 2009 (26)
▶ November (1)
▼ October (6)
The “Unlegacy my Code” Kata
Merge-Friendly Code
NAudio 1.3 Release Notes
Recording the Soundcard Output to WAV in NAudio
Playback of Sine Wave in NAudio
Looped Playback in .NET with NAudio
▶ September (10)
▶ August (3)
▶ July (2)
▶ May (1)
▶ March (1)
▶ February (1)
▶ January (1)
▶ 2008 (64)
▶ 2007 (27)

TAGS

NAudio (64) XAML (37) WPF (35) Silverlight (33) audio (33) Mercurial (14) IronPython (13) C# (11) DVCS (11) HOWTO (11) ASP.NET (10) MVVM (10) unit testing (10) Python (9) Software Development (9) TDD (9) LINQ

control to use this (until 1.3 is released).

Posted by Mark Heath at 12:30

Labels: audio, NAudio

59 comments:

Andri said...

Great post Mark.

do you happen to know if this would be possible for mpeg2 audio?

22 October 2009 at 17:46

Mark H said...

hi Andri, I'm not sure what you mean. Do you want to convert the sine wave to mpeg2?

22 October 2009 at 22:44

Andri said...

no to display a cine wave of mpeg2 audio file (actually a mplayer audiodump of a mpeg2-ts file)

23 October 2009 at 14:15

Mark H said...

this code doesn't display anything. It creates an audio sine wave.

23 October 2009 at 14:28

tom said...

Hi Mark,

Thanks for the run down, it's really helpful.

My problem is that I need to be able to create complex noises comprised of many waves.

Any idea how I could achieve this?

Thanks,

Tom

3 November 2009 at 15:31

Mark H said...

Hi Tom, you would make one WaveProvider for each simple waveform, and then another WaveProvider that read out of each one and summed them together.

Have a look at the WaveMixerStream in NAudio for an example

3 November 2009 at 16:47

Laserbeak43 said...

Hello Mark,
I'm really excited about this source, but I haven't used a third party library in a very long time and am not sure how to set your library up in Visual Stuido, without just altering the provided solution. Do you think you could show me how to do this?

(8) git (8) Refactoring (7) Single Responsibility Principle (7) TFS (7) Technical Debt (7) Windows 8 (7) Windows Forms (7) IoC (6) Pluralsight (6) Developer Principles (5) MP3 (5) MSBuild (5) NUnit (5) Visual Studio (5) ASP.NET MVC (4) Data Binding (4) Dependency Inversion Principle (4) NuGet (4) Open Closed Principle (4) Unity (4) clean code (4) screencast (4) Code Reviews (3) F# (3) HTML (3) JavaScript (3) Live Mesh (3) Media Foundation (3) Threading (3) VistaDB (3) Windows Vista (3) architecture (3) AIFF (2) Azure (2) ClickOnce (2) Liskov Substitution Principle (2) Model View Presenter (2) PowerShell (2) Reactive Extensions (2) SQL (2) Version Control (2) event aggregator (2) interop (2) Excel (1) Interface Segregation Principle (1) JSON (1) LINQPad (1) MEF (1) Observer Pattern (1) Razor (1) SOLID (1) Skype Voice Changer (1) SubSonic (1) Subversion (1) WASAPI (1) WCF (1) WebMatrix (1) book review (1) branching (1) code signing (1) coding dojo (1) complexity (1) css (1) debugging (1) headphones (1) kata (1) knockout (1) merging (1) microservices (1) mono (1) nupack (1) templating engine (1) typescript (1)

Thanks
Malik

11 November 2009 at 04:41

Mark H said...

Hi Malik,
Simply do an add reference to your project in Visual Studio and browse to the NAudio dll (downloaded from Codeplex).

11 November 2009 at 10:00

Laserbeak43 said...

This comment has been removed by the author.

11 November 2009 at 12:04

Laserbeak43 said...

Hello Mark,
That was great, thanks! I've added a slider to the form in an attempt to update the frequency in realtime, realizing that the sine wave's settings are set when the startStop button is toggled on.
Is there a way to update things in realtime?
----------added----------
private int Frequency;
private void trackBar1_Scroll(object sender, EventArgs e)
{
Frequency = trackBar1.Value;
}
----

11 November 2009 at 14:10

Mark H said...

you just need to keep a reference to the sine wave provider and set the Frequency property on that when your slider moves

11 November 2009 at 14:16

Laserbeak43 said...

VERY COOL! i just made the SineWaveProvider object viewable to whole class and updated it in the slider's scroll method. It works great, but is there a way to update it faster? seems to change values kind of choppy? would that be the fact that i'm no using Soundcard settings made for low latency response?

Thanks!!
Malik

11 November 2009 at 20:05

Mark H said...

the chopiness is most likely because you are not transitioning smoothly between frequencies, as well as due to latency settings (the Read method will run quickly and fill an entire buffer of say 100ms), so your changing Frequency will not kick in until the next buffer is processed.

12 November 2009 at 10:58

Frank said...

I have the example working in a form. Clicking the button starts the tone, and clicking it again stops it. However, when I try to get it to run from Main() itself, the sound only lasts a fraction of a second, then stops. If I put a breakpoint in the Read() routine, I see it only gets called 3 times.

Here is my code in Main:

```
Worker worker = new Worker();
worker.StartStopSineWave();
System.Threading.Thread.Sleep(5000);
worker.StartStopSineWave();
worker = null;
```

And here is the class:

```
public class Worker
{
WaveOut waveOut;

public void StartStopSineWave()
{

if (waveOut == null)
{
SineWaveProvider32 sineWaveProvider = new SineWaveProvider32();
sineWaveProvider.SetWaveFormat(16000, 1); // 16kHz mono
sineWaveProvider.Frequency = 1000;
sineWaveProvider.Amplitude = 0.10f;
waveOut = new WaveOut();
waveOut.Init(sineWaveProvider);
waveOut.DesiredLatency = 1000;
waveOut.Play();
}
else
{
waveOut.Stop();
waveOut.Dispose();
waveOut = null;
}
}

}
```

I suspect this problem has something to do with threading. Any ideas on how to solve it?

18 January 2010 at 19:46

Mark H said...

hi Frank,
you don't need to run WaveOut on a background thread. Try it on the GUI thread. Don't worry about blocking the GUI - things will still be responsive.

18 January 2010 at 20:08

Frank said...

Thanks Mark for the reply.

Actually I need it in a batch/console process. The override I am doing (in place of the sine generator) will continually get input from an external device, process it, and pass the resulting audio to the sound card for the user to hear. I guess I could create a dummy form but would prefer this to work in the background instead.

18 January 2010 at 20:56

Mark H said...

ok then, you need to create WaveOut with the Function callback (see the WaveOut constructor). Only be warned that certain laptop drivers seem to have problems with hanging in calls to WaveOutReset

18 January 2010 at 21:10

Frankl said...

That works. Thanks!

19 January 2010 at 00:40

Coach said...

Hi Mark!
I have implemented the sine wave generator. I would like it to play for a certain time so I added a timer. During that time I would like all other activites to stop, how do I manage that?
What I am after is:
Call StartSine
"While playing do nothing"
Call next action

20 January 2010 at 20:52

Frank said...

Coach, can't you just set some type of flag when the timer starts, clear it when the timer is done, and then check for the flag in the other activities and don't proceed (say sleep and then check again) as long as its set?

Frank

21 January 2010 at 14:59

Scott said...

Hi Mark!

Fantastic API - I've been struggling with the old Managed DirectX and this is much better.

I created the SineWaveProvider example you have. I'm using Windows 7 (64-bit) and Visual Studio C# Express. When I try to run the app, I get a "BadImageFormatException" error trying to load NAudio. The exception message is:

Could not load file or assembly 'NAudio, Version=1.3.8.0, Culture=neutral, PublicKeyToken=null' or one of its dependencies. An attempt was made to load a program with an incorrect format.

Is this a Windows 7 thing? Or a 64-bit thing? Or something else? Just wondering - I'll give it a go on my XP machine and see how that works out...

-Scott

17 February 2010 at 03:55

Mark H said...

Hi Scott,
NAudio is marked as x86 only due to the fact that the interop signatures have not been updated to work on x64 in all cases. When I eventually get myself an x64 system then I will be able to fix NAudio up. I suspect this is the problem you are running into

22 February 2010 at 10:49

Richard Allen said...

I can't get this to work, i'm not familiar with C#. How do you structure this code, in separate classes? Is there an example I can look at?

Thanks

Richard

25 February 2010 at 02:15

Scott said...

Interestingly enough, when I built the project in SharpDevelop 3.2, with the build options set to "Build for 32-bit Intel", it worked fine on my 64-bit system. Go figure. :)

28 February 2010 at 02:19

sdecorme said...

Hi Mark
I try to use the naudio lib in c# with VS2005 .
Is there a way to select which channel (Left or right) with the Wavein Function.
I don't information about that.
Thanks

9 March 2010 at 12:00

Mark H said...

@sdecorme you should record in stereo and then just throw away every other sample (usually it is left right left right)

11 March 2010 at 14:26

d.j.z. said...

What would be the best way to do this in stereo? I'm currently using this for a soft-synth and have been struggling trying to get this to work with the mixer; i can't create a channel like this

WaveChannel32 = new WaveChannel32(new MyWaveProvider32) because MyWaveProvider32 cannot be cast to WaveStream and I can't figure out why this is the case...

9 June 2010 at 04:53

Mark H said...

hi djz. NAudio doens't have a mixer that works with IWaveProvider yet. I hope to add one very soon

18 June 2010 at 15:37

d.j.z. said...

Thanks for responding! I am so thankful for the time you've spent on this project. Kudos!!

26 June 2010 at 01:25

Laserbeak43 said...

Hello Mark,
I was just wondering, How fast could i use this example if I separate the start and stop code and put them in KeyDown and KeyUp methods?
If I press the keys too fast, the debugger says that waveOut.Stop(); is referencing an object that doesn't exist anymore. "Object reference not set to an instance of an object."

My code is below, do you have any Ideas why this might happen?

Thanks for your library :)

private void Button_KeyDown(object sender, KeyEventArgs e)

```
{
...
startSine(1000, amp);
...
}

private void startSine(int freq, float amp)
{
if (waveOut == null)
{
sineWaveProvider = new SineWaveProvider32();
waveOut = new WaveOut();
sineWaveProvider.SetWaveFormat(16000, 1);
sineWaveProvider.Frequency = freq;
sineWaveProvider.Amplitude = amp;
waveOut.Init(sineWaveProvider);
waveOut.Play();
}

#endregion

#region Key Up Event Handler
private void Button_KeyUp(object sender, KeyEventArgs e)
{
...
stopSine();
...
}
private void stopSine()
{
waveOut.Stop();
waveOut.Dispose();
waveOut = null;
}
#endregion
```

11 July 2010 at 08:17

Laserbeak43 said...

Oh, it seems that my problem is multiple key presses. I'll have to write code to fix this. Please delete my last post, if you see it fit to do so.
Thanks!!

11 July 2010 at 08:21

Anzar said...

Thats really wonderful lib but I want to know is there any option available to know the frequency and amplitude of the laoded wave file ultimately to modify it???

21 March 2011 at 19:38

Anonymous said...

Hi Guys,How i can read the stereo channel using NAudio(Left Speaker and Right Speaker) values of a video while it's playing.Please reply for this.I need to use this for a project.

22 July 2011 at 03:24

Mark H said...

@Anonymous, I'm afraid NAudio can't access the audio channels of video files.

26 July 2011 at 11:00

Matt said...

Hi Mark,

I was wondering, will the above code still work with the latest beta version of NAudio that is on codeplex, or have you included some of the functionality that you suggested you would in the update?

5 August 2011 at 15:21

Mark H said...

@Matt, I can't see any reason why this wouldn't work with the latest NAudio code

9 August 2011 at 15:47

Steve said...

Hey Mark-

I have been playing with the NAudio library for a couple days. I got the sine wave to play. Very cool. But now I want to play back my own arbitrary float data. What's the best way to stuff in float data to a WaveOut object? My guess is to use BufferedWaveProvider. I'm a little uncertain how to get that class to handle float data though.

I tried instantiating BufferedWaveProvider a couple of ways. My first shot was using WaveForm(fs,1). I also tried WaveFormat.CreateIeeeFloatWaveFormat(fs,1).

But then I was shaky on how to pack my float data into a byte array to be used by the AddSamples() method of BufferedWaveProvider.

What do you recommend?

14 October 2011 at 22:57

Kyle said...

HI Mark

I was wondering maybe if there was a way to save the data directly to a .wav file ?

15 October 2011 at 17:24

Mark H said...

@Kyle, you can use the WaveFileWriter class from NAudio to do this

17 October 2011 at 10:28

Mark H said...

@Steve,
inherit from WaveProvider32 and provide your own float samples in the Read method

18 October 2011 at 16:02

Murat said...

thanks a lot Mark
great work

5 January 2012 at 13:45

John Dixon said...

Hi there Mark,

I've converted this into a derived WaveStream to allow mixing using WaveMixerStream.

I've added:

public override long Length
{
get { return long.MaxValue; }
}

However, I'm not sure how to get/set position.

Could you help me out?

Thanks.

5 March 2012 at 17:39

Mark H said...

@John, For position, just return the number of bytes read. For a sine generator there is no point allowing position to be set, so either throw an exception or just ignore the position value.

5 March 2012 at 18:59

Anonymous said...

Hello,

How could I use this to create other waveform types such as a triangle or sawtooth?

Also is there any way to avoid a click when changing frequencies during playback?

9 July 2012 at 23:39

Mark H said...

creating other waveforms is easy, especially sawtooth/triangle/square, as they can be easily calcualted. You can also use a wavetable.

To eliminate clicks, either fade out and back in over a short period (a few ms), or implement portamento. I hope to blog about that at smoe point.

10 July 2012 at 09:46

Saeed Shariati said...

If I want to Stop the Play automatically, after finishing the buffer, what can I do?

It seems that the read method will be executed by offset=0 each time!

22 August 2012 at 10:02

Mark H said...

offset is just the offset into the buffer that you should write to so this will normally be zero. You maintain state outside the Read method to know where you were up to.

5 September 2012 at 17:27

Alain Bressers said...

Hey Mark,

I'm a student and have the assignment to create a recording application which then records microphone input and has to determine the frequency and then display which note it is related to said frequency.

Will this article be the good way to follow in order to achieve this (retrieval of

frequency)?

Thanks!

12 March 2013 at 14:25

Mark H said...

hi Alain, this article is about how to generate sine waves, rather than how to detect frequency. To do that, I'd recommend researching FFT or autocorrelation algorithms.

12 March 2013 at 14:54

Rudi said...

Thanks for the Post. I have noticed that it work wrong with Stereo....
To fix it i add a line in the SineWaveProviderClass to proof if byte is even Number.

```
for (int n = 0; n < sampleCount; n++)
{
buffer[n + offset] = (float)(Amplitude * Math.Sin((2 * Math.PI * sample *
Frequency) / sampleRate));
if (this.WaveFormat.Channels == 1 || (n + offset) % 2 == 0)
sample++;
if (sample >= sampleRate) sample = 0;
}
```

I'm sorry for my bad english and hope it ist helpflull.

25 July 2013 at 18:41

Rudi said...

Tahnk you for this great Post. I have noticed that it is working wrong in stereo-mode. To fix it i have added a line to proof if floatarray is even number:

```
for (int n = 0; n < sampleCount; n++)
{
buffer[n + offset] = (float)(Amplitude * Math.Sin((2 * Math.PI * sample *
Frequency) / sampleRate));
if (this.WaveFormat.Channels == 1 || (n + offset) % 2 == 0)
sample++;
if (sample >= sampleRate) sample = 0;
}
```

I'm sorry for my bad English and hope my Coment ist helpfully.

25 July 2013 at 18:44

Unknown said...

Hi!

Thanks a lot for your work on NAudio, it's much appreciated!

The last few days I have been working with additive synthesis based on this example. At first I used WaveOut (as in your code), but then I started moving my code to a Windows Store App. I know that NAudio isn't officially available for such apps, but I found some post stating how to use NuGet to get a pre-release. I changed WaveOut to WasapiOutRT like this:

```
_sampleMixer = new MixingSampleProvider(_voices[0].SampleProviders);
_sampleToWaveProvider = new SampleToWaveProvider(_sampleMixer);
_waveOut = new WasapiOutRT(AudioClientShareMode.Shared, 100);
await _waveOut.Init(_sampleToWaveProvider);
_waveOut.Play();
```

It works, but there's a horrible latency. I'm not sure, but I don't think it lagged as

much when using WaveOut. I am a total beginner in programming audio like this, but when I stepped through the code I realized that the Read() method is called just once every second when using WasapiOutRT, with a buffer size of 44100, as opposed to every ~150ms or so on WaveOut (buffer size 6615), and to me that sounds like a source of latency. Or do I miss something?

Best regards,
Håkan Eriksson

31 December 2013 at 12:43

Mark H said...

I think in shared mode, sometimes your requested latency can be ignored. If you can debug the NAudio code, look in the Init method of WasapiOutRT

31 December 2013 at 14:16

Unknown said...

Hi!

Thanks for your swift reply!

I stepped through the NAudio code. The reported latency from the AudioClient is 11 ms.

However, when stepping through the code I noticed that in MediaFoundationTransform you explicitly read one second from the provider:

```
private IMFSample ReadFromSource()
{
// we always read a full second
int bytesRead = sourceProvider.Read(sourceBuffer, 0, sourceBuffer.Length);
```

Maybe there's something I don't understand well enough, but by reading once a second, doesn't that imply that changes I make in the audio generation can be delayed up to one second until they are heard?

Thanks a lot for your help, and a happy New Year as well! :-)

31 December 2013 at 19:42

Unknown said...

Hi!

Thanks for your swift reply!

I stepped through the NAudio code. The reported latency from the AudioClient is 11 ms.

However, when stepping through the code I noticed that in MediaFoundationTransform you explicitly read one second from the provider:

```
private IMFSample ReadFromSource()
{
// we always read a full second
int bytesRead = sourceProvider.Read(sourceBuffer, 0, sourceBuffer.Length);
```

Maybe there's something I don't understand well enough, but by reading once a second, doesn't that imply that changes I make in the audio generation can be delayed up to one second until they are heard?

Thanks a lot for your help, and I wish you a happy New Year! :-)

1 January 2014 at 22:26

Mark H said...

why is an MFT involved? Are you resampling? If possible work at the sample rate of the soundcard. But I probably should do something a bit cleverer with that MFT read size code.

1 January 2014 at 22:31

Unknown said...

I haven't given much thought to where the sampling rate comes from, so I might be resampling without intending to. You've seen my initialization code (in my first post), and the code producing the sine wave is pretty much as your example. If I understand your code correctly, that would imply that the sampling frequency is set by the default constructor of WaveProvider32, i.e. 44100 Hz in mono?

And I see that it is possible to use other constructors to set other sample rates, but how would I know what sample rate the soundcard has?

2 January 2014 at 00:32

Unknown said...

Actually, by stepping through the NAudio initialization, I learned that my sound card's sample rate indeed is different from the one I got from the default constructor. When setting the sample rate correctly there's no lag (an no glitches, which also occurred before).

Is there a way to get the sound card's sample rate from NAudio?

2 January 2014 at 21:50

Mark H said...

With WasapiOut at least I think you can get it just by asking for the WaveFormat before you call Init. (It's been a while, but I think that's how it works from memory). By the way, best to ask NAudio questions over at the CodePlex discussion site if possible (naudio.codeplex.com)

2 January 2014 at 22:09

Post a Comment

Links to this post

Create a Link

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)