Gregor Bowie's Blog

Random thoughts

Audio Capture and WAV File Writing in .NET

gregorbowie .NET June 3, 2010 5 Minutes

Firstly, a lot of what I'm going to write here is available from a number of sources on the internet. However, I've spent the past week developing this solution, and none of what was out there quite solved it for me – a lot of it came close though. I'll give full references at the end, and I thoroughly recommend heading over to those sites.

Secondly, its been a week of playing around with various things. I'm hoping I can remember everything here, but there is a high chance that I'll miss something. If something doesn't hang together please let me know and I'll see if I can solve it.

Audio Capture

Download and install DirectX SDK from Microsoft. This provides a number of DLLs that we'll make use of it.

Now add references to Microsoft.DirectX.dll, Microsoft.DirectX.InputSound.dll and Microsoft.DirectX.AudioVideoPlayback. I'll explain where these are used as we walk through the implementation.

Add imports to these libraries:

```
1  using Microsoft.DirectX;
2  using Microsoft.DirectX.AudioVideoPlayback;
3  using DirectX = Microsoft.DirectX.DirectSound;
```

Set-up the WAV format parameters that will be used to record the audio:

```
1  WaveFormat = new DirectX.WaveFormat();
2  WaveFormat.AverageBytesPerSecond = 44100;
3  WaveFormat.BitsPerSample = 16;
4  WaveFormat.BlockAlign = 2;
5  WaveFormat.Channels = 1;
6  WaveFormat.FormatTag = DirectX.WaveFormatTag.Pcm;
7  WaveFormat.SamplesPerSecond = 22050;
```

Now implement a quick routine to see what devices are available to record from:

```
1  var cap = default(DirectX.Capture);
2  var cdc = new DirectX.CaptureDevicesCollection();
3  for (int i = 0; i < cdc.Count; i++)
4  {
5     if (cdc[i].Description.ToLower().Contains("mic"))
6     {
7        cap = new DirectX.Capture(cdc[i].DriverGuid);
8        break;
9     }
10 }
```

This code here is basically looping through all the capture devices. Along side this, I implemented a debug routine to record the description to a StringBuilder to output to a MessageBox. As you can see, we detect the 1st capture device that has a description containing 'mic'.

Next we need to set-up some boundaries for recordings, and then set up the description of the Capture Buffer, which for the main part sets up the Capture Buffer Size:

```
NotifySize = (1024 > WaveFormat.AverageBytesPerSecond / 8) ? 1024 : (Wavel
NotifySize -= NotifySize % WaveFormat.BlockAlign;
//CaptureBuffSize = 1000000;

var capDesc = new DirectX.CaptureBufferDescription()
{
   BufferBytes = CaptureBuffSize,
   WaveMapped = false,
   ControlEffects = false,
   Format = WaveFormat
};

TheCaptureBuffer = new DirectX.CaptureBuffer(capDesc, cap);
```

The Buffer Size is effectively how much audio can be stored in the buffer. The buffer loops, and therefore if you make it too small you will overwrite the start of the buffer. There are nicer solutions using Threads to write out the buffer contents, so the buffer can remain small. However it couldn't get these going. The references point to examples – I was very close to getting it working, so it must have been something minor that was missing. In this implementation I just have a huge buffer. I have found setting the buffer to 15,000,000 should give about 5 minutes recording time.

Now that the capture buffer is set up, we're able to record to the buffer. I started this going, and then copied to a second output buffer to playback the recording. However I've since implement saving to a file and don't have this code to hand. I should be easy enough to work out from the references at the end, but if someone specifically wants this functionality, please ask and I'll look to implement an example.

So my StartRecording() method currently does 2 things – writes out the file header as a holding place, and then starts recording into the buffer:

```
public void StartRecording()
{
   WriteFileHeader();
   TheCaptureBuffer.Start(true);
}
```

The WriteFileHeader() takes the form: (this is just a standard header for a WAV file – please see references for more detailed explanations)

```
1  private void WriteFileHeader()
2  {
3      bw = new BinaryWriter(new FileStream(WavFilename, FileMode.Create));
4
5      char[] Riff = { 'R', 'I', 'F', 'F' };
6      char[] Wave = { 'W', 'A', 'V', 'E' };
7      char[] Fmt = { 'f', 'm', 't', ' ' };
8      char[] Data = { 'd', 'a', 't', 'a' };
9      short padding = 1;
10     int formatLength = 0x10;
11     int length = 0; // fill this in later!
12     short shBytesPerSample = 2; // changing the WaveFormat recording param
13     // see referenced blog posts for more details
14
15     bw.Write(Riff);
16     bw.Write(length);
17     bw.Write(Wave);
18     bw.Write(Fmt);
19     bw.Write(formatLength);
20     bw.Write(padding);
21     bw.Write(WaveFormat.Channels);
22     bw.Write(WaveFormat.SamplesPerSecond);
23     bw.Write(WaveFormat.AverageBytesPerSecond);
24     bw.Write(shBytesPerSample);
25     bw.Write(WaveFormat.BitsPerSample);
26     bw.Write(Data);
27     bw.Write((int)0); // update sample later
28 }
```

Next step is to stop recording and then complete the write out of the WAV file:

```
public void StopRecording()
{
   TheCaptureBuffer.Stop();
   WriteFileChunks();
   WriteFileTrailer();
}

private void WriteFileChunks()
{
   int readPos, capturePos;

   TheCaptureBuffer.GetCurrentPosition(out capturePos, out readPos);
   int lockSize = readPos - CaptureOffset;
   if (lockSize < 0)
      lockSize += CaptureBuffSize;

   lockSize -= (lockSize % NotifySize);

   if (0 == lockSize)
      return;

   byte[] capturedData = (byte[])TheCaptureBuffer.Read(CaptureOffset, typ
   bw.Write(capturedData, 0, capturedData.Length);

   SampleCount += capturedData.Length;

   // CaptureOffset probably not required - hangover from threaded implem
   CaptureOffset += capturedData.Length;
   CaptureOffset %= CaptureBuffSize;
}

private void WriteFileTrailer()
{
   bw.Seek(4, SeekOrigin.Begin);
   bw.Write((int)(SampleCount + 36));
   bw.Seek(40, SeekOrigin.Begin);
   bw.Write(SampleCount);
   bw.Close();
}
```

Hopefully most of that fairly self explanatory. Just whizzing through it quickly:

1. The Buffer stops recording.
2. We then determine how many bytes have been recording, and move this along to the next boundary.
3. We then read these bytes into a byte array, which are written to the file.
4. Finally we got back and update the file size information in the file header.

WAV to MP3 Conversion

I then moved on to convert the WAV file to MP3 – but this is documented here so I won't repeat: http://www.codeproject.com/KB/audio-video/MP3Compressor.aspx (http://www.codeproject.com/KB/audio-video/MP3Compressor.aspx)

I then moved on to get integration going with a WebCam. This was more straight-forward, but I'll document again tomorrow about how I did that.

Lock Loader Issue

As a final thought, I've just remembered a Lock Loader issue. Reading around its something to do with the conflict between .NET 1.1 libraries and .NET 2.0. To overcome this, head to the Debug menu in Visual Studio. Then click Exceptions, then expand Managed Code, and then find 'Lock Loader' and untick the throw checkbox.

There is another solution documented in the codeproject blog post linked to from this site (see a few lines up for MP3 conversion).

References

I'd like to make it clear I can't thank these references enough. They really got me started, and without them I wouldn't have solved my problem. Like I start at the start, I don't know why I could get the file writing to work from many of the examples, but got there in the end with a mish-mash of all the work these people had done before me.

The blog post got me started. It outlines the code required to initialise the capture device, and then also a secondary output buffer to playback the recording. It plays back the recording with a little echoing effect.
http://khason.net/blog/capturing-and-streaming-sound-by-using-directsound-with-c/
(http://khason.net/blog/capturing-and-streaming-sound-by-using-directsound-with-c/)

A WAV recording example in VB.NET:
http://www.freevbcode.com/ShowCode.asp?ID=8643&NoBox=True
(http://www.freevbcode.com/ShowCode.asp?ID=8643&NoBox=True)

This blog post gives a quick example of how to playback a file. Something I forgot to include in my post.
http://www.riemers.net/eng/Tutorials/DirectX/Csharp/Series2/tut17.php
(http://www.riemers.net/eng/Tutorials/DirectX/Csharp/Series2/tut17.php)

Tagged:
.NET,
audio recording,
C#,
directx,
microphone,
wav

Published by gregorbowie

View all posts by gregorbowie

3 thoughts on "Audio Capture and WAV File Writing in .NET"

Omar says:
December 29, 2012 at 12:15

Hi ,
Actually I am facing a problem , I capture the sound from the mic and it saved to file but when i try to play the file , it notifies me that the file is corrupt … any idea ?

↳ Reply
gregorbowie says:
January 2, 2013 at 08:42
Hi Omar,

Afraid its been a while since I looked at this code. This 1st thought I had when I saw your comment was that you might not be flushing the entire buffer to file?

↳ Reply
Abhilash says:
January 11, 2013 at 10:14
private void WriteFileChunks()
09 {
10 int readPos, capturePos;
11
12 TheCaptureBuffer.GetCurrentPosition(out capturePos, out readPos);
13 int lockSize = readPos – CaptureOffset;
14 if (lockSize < 0)

What u mean by this capture offset?? what kind of data is holding in than???

↳ Reply