Saving Microphone stream to wave format in Windows Phone

This article explains how to get audio input from the microphone, convert it to .wav format, and save it to IsolatedStorage.

Introduction

As the microphone's stream buffer provide raw data we need to convert it into a well known file format before saving in order to allow users to play the recorded file everywhere. The easiest solution is to use the .wav file format.

Microphone Sample demonstrates how to get audio input from the microphone in a Silverlight for Windows Phone application. This project is used in this guide and we will be focused on saving process.

Note: This article was created using un-attributed code samples from the Internet. Dmehers asserts the origin of this information is his blog post: Storing WP7 recorded audio as WAV format streams.

Saving into WAV format

As we use isolated storage we need to use the following namespaces:

```
using System.Windows;
using System.IO.IsolatedStorage;
using System.Windows.Resources;
```

Then add the following functions to your code:

```
private void SaveToIsolatedStorage()
{
            // first, we grab the current apps isolated storage handle
            IsolatedStorageFile isf = IsolatedStorageFile.GetUserStoreForApplication();

            // we give our file a filename
            string strSaveName = "myFile.wav";

            // if that file exists...
            if (isf.FileExists(strSaveName))
            {
                // then delete it
                isf.DeleteFile(strSaveName);
            }

            // now we set up an isolated storage stream to point to store our data
            IsolatedStorageFileStream isfStream =
                     new IsolatedStorageFileStream(strSaveName,
                     FileMode.Create, IsolatedStorageFile.GetUserStoreForApplication());

            isfStream.Write(stream.ToArray(), 0, stream.ToArray().Length);

            // ok, done with isolated storage... so close it
            isfStream.Close();
}

public void UpdateWavHeader(Stream stream)
{
            if (!stream.CanSeek) throw new Exception("Can't seek stream to update wav header");

            var oldPos = stream.Position;

            // ChunkSize 36 + SubChunk2Size
            stream.Seek(4, SeekOrigin.Begin);
            stream.Write(BitConverter.GetBytes((int)stream.Length - 8), 0, 4);

            // Subchunk2Size == NumSamples * NumChannels * BitsPerSample/8 This is the number of bytes in the data.
            stream.Seek(40, SeekOrigin.Begin);
            stream.Write(BitConverter.GetBytes((int)stream.Length - 44), 0, 4);

            stream.Seek(oldPos, SeekOrigin.Begin);
}

 public void WriteWavHeader(Stream stream, int sampleRate)
{
            const int bitsPerSample = 16;
            const int bytesPerSample = bitsPerSample / 8;
            var encoding = System.Text.Encoding.UTF8;

            // ChunkID Contains the letters "RIFF" in ASCII form (0x52494646 big-endian form).
            stream.Write(encoding.GetBytes("RIFF"), 0, 4);

            // NOTE this will be filled in later
            stream.Write(BitConverter.GetBytes(0), 0, 4);

            // Format Contains the letters "WAVE"(0x57415645 big-endian form).
            stream.Write(encoding.GetBytes("WAVE"), 0, 4);

            // Subchunk1ID Contains the letters "fmt " (0x666d7420 big-endian form).
            stream.Write(encoding.GetBytes("fmt "), 0, 4);

            // Subchunk1Size 16 for PCM. This is the size of therest of the Subchunk which follows this number.
            stream.Write(BitConverter.GetBytes(16), 0, 4);
```


(C) Copyright Nokia 2013. All rights reserved.

```
            // AudioFormat PCM = 1 (i.e. Linear quantization) Values other than 1 indicate some form of compression.
            stream.Write(BitConverter.GetBytes((short)1), 0, 2);

            // NumChannels Mono = 1, Stereo = 2, etc.
            stream.Write(BitConverter.GetBytes((short)1), 0, 2);

            // SampleRate 8000, 44100, etc.
            stream.Write(BitConverter.GetBytes(sampleRate), 0, 4);

            // ByteRate = SampleRate * NumChannels * BitsPerSample/8
            stream.Write(BitConverter.GetBytes(sampleRate * bytesPerSample), 0, 4);

            // BlockAlign NumChannels * BitsPerSample/8 The number of bytes for one sample including all channels.
            stream.Write(BitConverter.GetBytes((short)(bytesPerSample)), 0, 2);

            // BitsPerSample 8 bits = 8, 16 bits = 16, etc.
            stream.Write(BitConverter.GetBytes((short)(bitsPerSample)), 0, 2);

            // Subchunk2ID Contains the letters "data" (0x64617461 big-endian form).
            stream.Write(encoding.GetBytes("data"), 0, 4);

            // NOTE to be filled in later
            stream.Write(BitConverter.GetBytes(0), 0, 4);
}
```

On function delegated to begin the record process include the call to `WriteWavHeader()` before the `microphone.Start()` function

```
private void recordButton_Click(object sender, EventArgs e)
{
            ...
            stream.SetLength(0);

            WriteWavHeader(stream, microphone.SampleRate);

            // Start recording
            microphone.Start();
            ...
}
```

After `microphone.Stop()` add the call to `UpdateWavHeader` and to `SaveToIsolatedStorage`

```
microphone.Stop();
UpdateWavHeader(stream);
SaveToIsolatedStorage();
```

Related links

- Isolated Storage Explorer Tool
- FFT project
- SkyDrive - How to upload content on Windows Phone

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.