Multimedia-Lecture-Four

Audio

Eng. Noor Alhakim

Audio Signal

An audio signal is a representation of sound.

It is typically using a level of electrical voltage for analog signals, and a series of binary numbers for digital signals.

Audio signals have frequencies in the audio frequency range of roughly 20 to 20,000 Hz, which corresponds to the lower and upper limits of human hearing.

Dealing With Audio in C#

- Reading Audio File
- Writing Audio File
- Playing Audio File
- Find More Information about Audio
- Audio Scaling
- Drawing the Audio Signal
- Merge Two Audio Files
- Recording Audio File

Reading Audio File:

```
string audioFile = "input.wav";

var audioFileReader = new AudioFileReader(audioFile);
```

Writing Audio File:

```
WaveFileWriter.CreateWaveFile("output.wav", audioFileReader);
```

Playing an audio:

```
using (var outputDevice = new WaveOutEvent())
  {
   outputDevice.Init(audioFileReader);
   outputDevice.Play();
   while (outputDevice.PlaybackState == PlaybackState.Playing)
     {
      System.Threading.Thread.Sleep(1000);
     }

  }
```

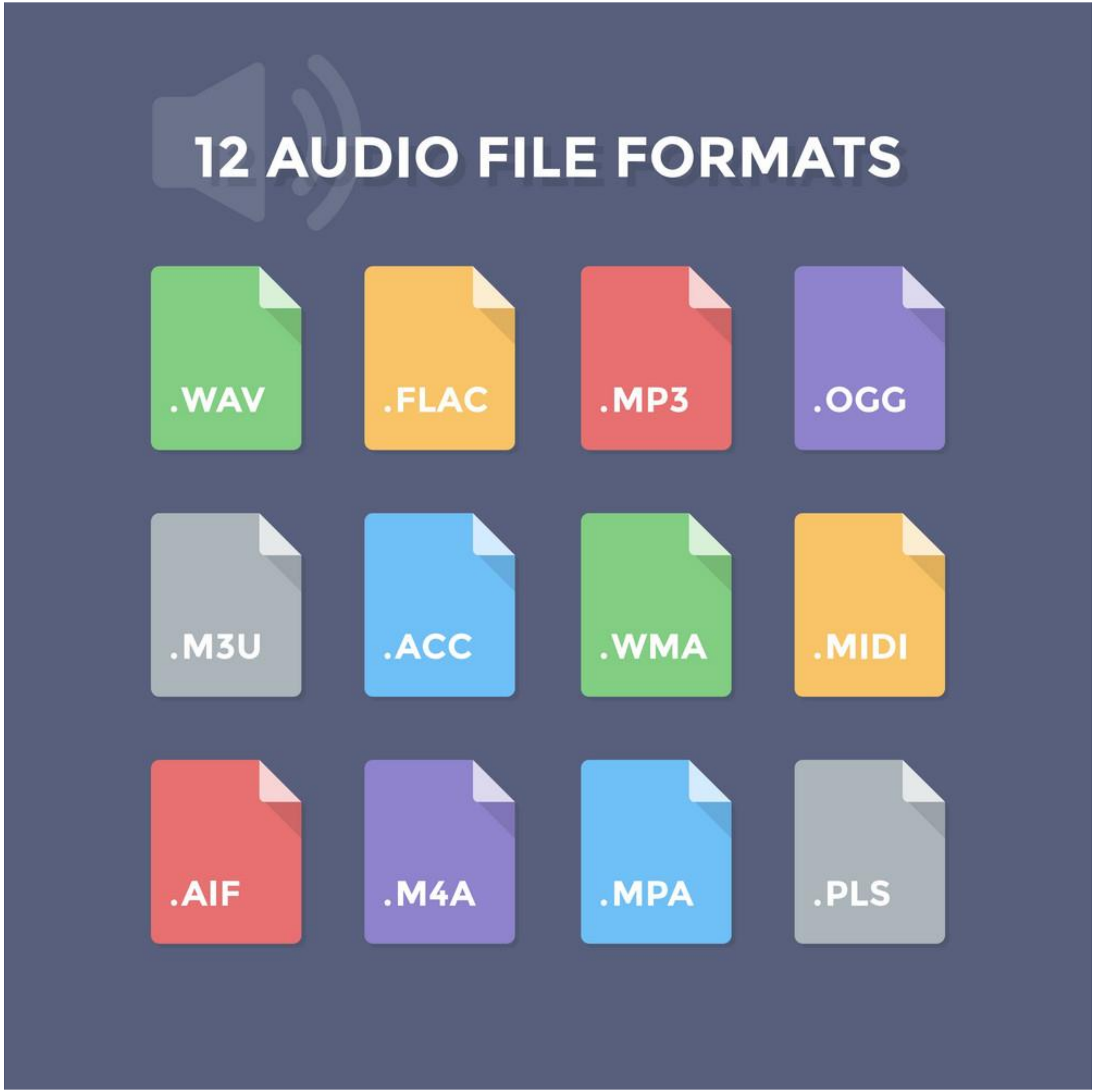
Find More Information about Audio:

```
using (var audioFileReader = new AudioFileReader(audioFile))
    {
        // Get information about the audio file
        TimeSpan duration = audioFileReader.TotalTime;
        int sampleRate = audioFileReader.WaveFormat.SampleRate;
        int channels = audioFileReader.WaveFormat.Channels;

        // Display the information
        Console.WriteLine($"Duration: {duration}");
        Console.WriteLine($"Sample Rate: {sampleRate}");
        Console.WriteLine($"Channels: {channels}");
    }
```

Find out the BitPerSample property

Audio Format



Audio Scaling

Audio Scaling allows for modifying an audio signal amplitude or frequency.

To increase the volume of the audio track you can multiple the variable it is stored in by a scalar.

To slow down or speed up the track played you can adjust the sampling rate.

Modifying the Volume of Audio File

```
using (var audioFileReader = new AudioFileReader(audioFile))
     {
       // Create a new buffer to store scaled audio data
       float[] buffer = new float[audioFileReader.Length];

       // Read audio data into the buffer
       int samplesRead = audioFileReader.Read(buffer, 0, buffer.Length);

       // Scale each sample in the buffer
       for (int i = 0; i < samplesRead; i++)
           {
             buffer[i] *= scaleFactor;
           }

        // Create a new WaveFileWriter to write scaled audio data to a new file
        using (var waveFileWriter = new WaveFileWriter(outputAudioFile, audioFileReader.WaveFormat))
           {
             // Write the scaled audio data to the output file
             byte[] byteBuffer = new byte[samplesRead * sizeof(float)];
             Buffer.BlockCopy(buffer, 0, byteBuffer, 0, byteBuffer.Length);
             waveFileWriter.Write(byteBuffer, 0, byteBuffer.Length);
            }
      }
```

Drawing the Audio Signal

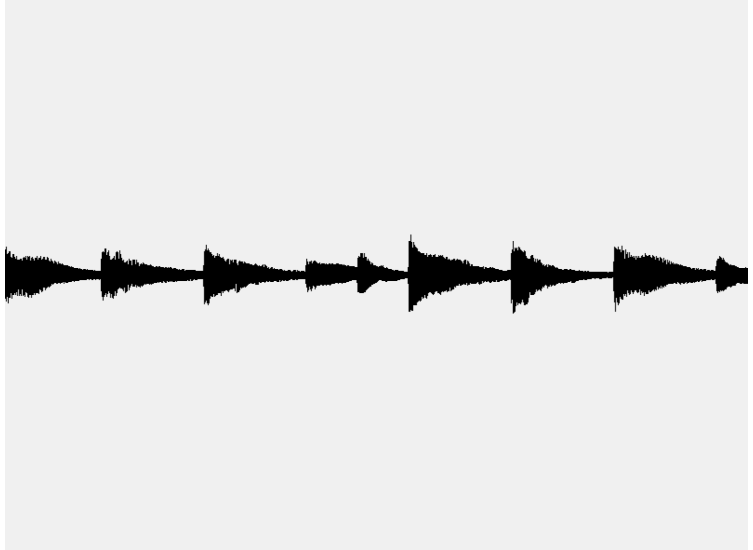
```
using NAudio.Gui;

WaveViewer waveViewer = new WaveViewer();
waveViewer.Dock = DockStyle.Fill;
waveViewer.SamplesPerPixel = 400;
waveViewer.StartPosition = 10000;

// Load the audio file and set it as the WaveStream for the WaveViewer
WaveFileReader waveFileReader = new WaveFileReader(audioFile1);
waveViewer.WaveStream = waveFileReader;

// Create and configure the form
Form form = new Form();
form.Text = "WaveViewer Example";
form.Controls.Add(waveViewer);

form.ClientSize = new System.Drawing.Size(800, 600); // Set the form size
form.ShowDialog();
```



Recording Audio File

```
using (var waveIn = new WaveInEvent())
      {
         waveIn.WaveFormat = new WaveFormat(44100, 1); // 44100 Hz, mono
         WaveFileWriter waveFileWriter = null;

         waveIn.DataAvailable += (sender, e) =>
           {
            // Initialize the WaveFileWriter on the first call to the DataAvailable event
             if (waveFileWriter == null)
              {
                waveFileWriter = new WaveFileWriter(outputAudioFile, waveIn.WaveFormat);
              }
               // Write the recorded audio data to the WAV file
               waveFileWriter.Write(e.Buffer, 0, e.BytesRecorded);
                };
```

Recording Audio File

```
        waveIn.StartRecording();

        Console.WriteLine("Recording. Press any key to stop...");
        Console.ReadKey();

        waveIn.StopRecording();

        // Close the WaveFileWriter after recording is stopped
        waveFileWriter?.Dispose();

        Console.WriteLine("Recording stopped. Audio saved to: " + outputAudioFile);
    }
```

Merge two audio files

Example:

```
using NAudio.Wave.SampleProviders;

using (var reader1 = new AudioFileReader(audioFile1))
 using (var reader2 = new AudioFileReader(audioFile2))
      {
        var mixer = new MixingSampleProvider(new[] { reader1, reader2 });
        WaveFileWriter.CreateWaveFile16(outputAudioFile, mixer);
      }
```

Now check newtest.wav file and note the duration property should be equal to duration of two files together

Exercise:

Write a C#-code to:

a. Reverse the audio signal.
b. Reverse the audio Channels.
c. Plot the audio signal after reversing it.
d. Save the new audio files on your disk

Tip:
Use Charting to plot the audio signal OR you can use OxyPlot

That’s All