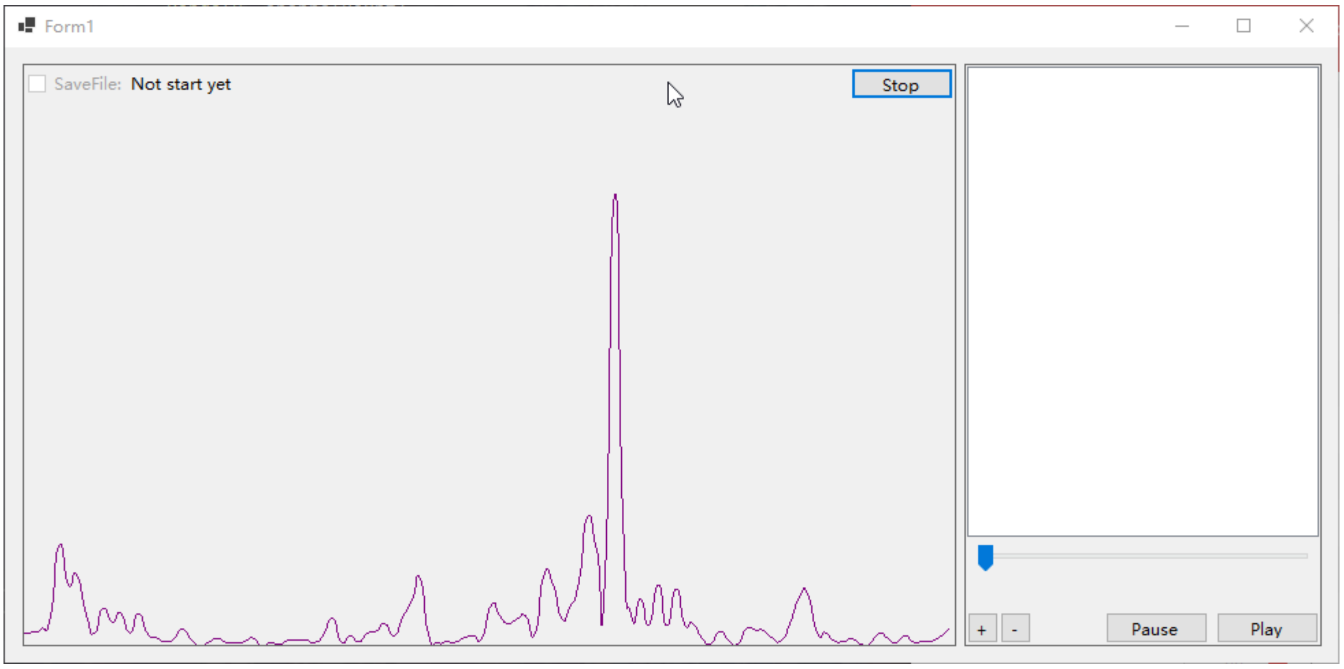
(https://developpaper.com) Navigator

Position: Home (https://developpaper.com) > Blogs (https://developpaper.com/category/blogs/) > .NET (https://developpaper.com/category/blogs/net/) > C# (https://developpaper.com/category/blogs/net/c-net/) > Content

C # using naudio to realize audio visualization

Time： 2022-2-17

Preview:



Capture sound c (https://developpaper.com/tag/c/)ard output:

To realize audio visualization, the first step is to obtain audio sampling. Here, we choose to use the audio being played by the computer as the sampling source for processing:

In naudio, you can use wasapiloopbackcapture to capture:

```
WasapiLoopbackCapture cap = new WasapiLoopbackCapture();
cap. Dataavailable + = (sender, e) = > // this event is triggered when recording da
ta is available. The parameter contains audio data
{
    Float [] allsamples = enumerable // extract samples from data
        . range (0, e.bytesrecorded / 4) // divide by four because every four bytes
in the buffer constitute a floating point number, and a floating point number is a
 sample
        . select (I = > bitconverter. Tosingle (e.buffer, I * 4)) // convert to flo
at
        .ToArray();    //  Convert to array
    //After the sampling is obtained, it is processed in detail here
}
cap. StartRecording();   //  start recording
```

Separate left and right channels:

After obtaining the samples, we need to do a little processing on the samples, because the captured data is divided into channels, generally left and right channels:

```
//Set that we have saved the sample to the variable allsamples with the type of flo
at []
int channelCount = cap. WaveFormat. Channels;   //  The waveformat of wasapiloopbac
kcapture specifies the waveform format of the current sound, including the number o
f channels
float[][] channelSamples = Enumerable
    .Range(0, channelCount)
    .Select(channel => Enumerable
        .Range(0, AllSamples.Length / channelCount)
        .Select(i => AllSamples[channel + i * channelCount])
        .ToArray())
    .ToArray();
```

Take the average value of the channel

After the samples are divided into channel samples, we can combine them and take the average value for drawing:

```
//Set that we have saved the separated samples to the variable channelsamples with
 the type of float [] []
//For example, if the number of channels is 2, the samples of the left channel are
 channelsamples [0], and the samples of the right channel are channelsamples [1]
float[] averageSamples = Enumerable
    .Range(0, AllSamples.Length / channelCount)
    .Select(index => Enumerable
        .Range(0, channelCount)
        .Select(channel => ChannelSamples[channel][index])
        .Average())
    .ToArray();
```

Draw time domain image:

After processing the just sampled data, you can directly draw it as data into the window. This is the time domain image. Here, the simplest polyline drawing is used

```
//Set G as the graphics object of the window, and windowheight as the height of the
display area of the window
//Set the average value of channel sampling as averagesamples and the type as float
[]
Point[] points = AverageSamples
    .Select((v, i) => new Point(i, windowHeight - v))
    .ToArray();   //  Convert the data into coordinate points
g.DrawLines(Pens.Black, points);   //  Connect these points and draw lines
```

Fourier transform:

The method of fast Fourier transform is also provided in naudio. Through Fourier transform, the time domain data can be converted into frequency domain data, which is what we call spectrum

```
//We Fourier transform the complex number to get an array of samples

//Because for the fast Fourier transform algorithm, the data length needs to be the
nth power of 2, which is carried out here
float log = Math. Ceiling(Math.Log(AverageSamples.Length, 2));   //  Take logarithm
and round up
int newLen = (int)Math. Pow(2, log);                             //  Calculate new
 length
float[] filledSamples = new float[newLen];
Array. Copy(AverageSamples, filledSamples, AverageSamples.Length);   //  Copy to ne
w array
Complex[] complexSrc = filledSamples
    . select (v = > New complex() {x = V}) // convert samples to complex numbers
    .ToArray();
FastFourierTransform(false, log, complexSrc);   //  Fourier transform

//After the transformation, complexsrc has been processed, in which the frequency d
omain information is stored
```

Analyze frequency domain information:

The frequency domain information of Fourier transform needs a little processing before it can be used conveniently. The first is to extract useful information:

```
//In the Fourier transform result of naudio, there seems to be no DC component (whi
ch makes our processing more convenient), but it also has conjugation (that is, the
data is left-right symmetrical, and only half is useful)
//Still use the just complexsrc as the transformation result, and its type is compl
ex []

Complex[] halfData = complexSrc
    .Take(complexSrc.Length / 2)
    .ToArray();    //  Half the data
float[] dftData = halfData
    . select (v = > math. Sqrt (v.x * v.x + v.y * v.y)) // take the modulus of the
 complex number
    .ToArray();    //  Convert the complex result to the frequency amplitude we nee
d

//In fact, here you can draw these data on the window, which is already a frequency
domain image, but for music visualization, we don't need some frequency data at all
//For example, for a frequency of 10000hz, we don't need to draw it at all. It's en
ough to take the minimum frequency ~ 2500Hz
//For the transformation result, the frequency difference between each two data is
 calculated as the sampling rate / number of samples, so the number we want to take
can also be obtained from 2500 / (sampling rate / number of samples)
int count = 2500 / (cap.WaveFormat.SampleRate / filledSamples.Length);
float[] finalData = dftData.Take(count).ToArray();
```

Draw frequency domain image:

After obtaining the finaldata analyzed above, we can draw it directly. This time, we use soft curves

```
//Set G as the graphics object of the window and height as the height of the window
PointF[] points = finalData
    .Select((v, i) => new PointF(i, height - v))
    .ToArray();
g.DrawCurve(Pens.Purple, points);    //  Graphics can draw curves directly
```

Better drawing:

In the above time domain and frequency domain images, we all use the index of the data as the X coordinate and the window height minus the data value as the Y coordinate. There are two prominent problems:

- The data may not fill the width of the window or exceed the width of the window
- When the data is too large, it will also cause the drawn lines to exceed the window height

The first problem is easy to solve. Just make the percentage of the index in the data length exactly equal to the percentage of the X coordinate relative to the window width:

\[x = index \div dataLength * windowWidth\]

For the second problem, there are two solutions: one is to weight the data directly, for example, uniformly multiply by 0.5 to reduce the data by one section, and the other is to set a function, such as log function. After all, when the log function has higher independent variables, the change trend of dependent variables becomes smaller and smaller, so we only need to deal with this log function slightly, It can be directly applied to the data transformation data so that it does not exceed the drawing area of the window

In addition, we can also smooth the spectrum display (referring to animation transformation). Its principle is roughly as follows:

- For example, the result of Fourier transform this time is: `{0, 100, 50}` ,
- The result of the next Fourier transform is: `{100, 0, 0}` ,
- It can be concluded that the increment is: `{100, -100, -50}` ,
- When updating the transformation results, we no longer directly replace the new results with the old results, but add increment on the basis of the old results × weight
- For example, the weight is `0.5` The actual increment is: `{50, -50, -25}` ,
- Then the actual new value is: `{50, 50, 25}` ,
- If the result of the next transformation is still `{100, 0, 0}` , let's start again `{50, 50, 25}` Approaching the new value, the weight is still `0.5` , then the actual increment is: `{25, -25, -12.5}` ,

Did you notice? This increment is half of the last increment, which is exactly a deceleration movement. The greater the difference between the new value and the old value, the faster the change, and they will continue to coincide, so the speed will continue to slow down, forming a spectrum of deceleration movement

More:

For more information about the use of naudio, see this article: [c#] various common ways of using naudio, playing, recording, transcoding and audio visualization

The project is open source:

Most of the contents involved in this article are in GitHub COM / slimenull / null in audiotest repository Audiovisualizer project has write (notes are in place)

In fact, I've wanted to do audio visualization for a long time, but I'm not very good at math, but I finally stuck to it and got it out. I'm always excited

This is the end of this article about [c#] using naudio to realize audio visualization. For more relevant c# audio visualization content, please search the previous articles of developeppaer or continue to browse the relevant articles below. I hope you will support developeppaer in the future!

Tags: c (https://developpaper.com/tag/c/), C# audio visualization (https://developpaper.com/tag/c-audio-visualization/), Naudio audio visualization (https://developpaper.com/tag/naudio-audio-visualization/)

Recommended Today

Modify user information changeinfo (https://developpap…

When judging the persistence layer: Problem: there is such a problem when modifying user information. For example: the user's email is not required. It was not empty originally. At this time, the user deletes the mailbox information and submits it. At this time, if it is not empty to judge whether it needs to be […]

- Combination of quartz and topshelf to realize window…
- Solution to the dllnotfoundexception error reported by …
- Dynamic splicing expression (https://developpaper.c…
- Unity3d uses fairygui to customize font operation (htt…
- 2022-03-09 switching between two scenes of unity 3…
- C#tcp communication (https://developpaper.com/ctcp…
- Example: use C # Net to teach you how to develop w…
- C # based socket chat room (with source code) (http…
- Get and set methods in C # (https://developpaper.co…
- Three methods of c# loading word (https://developpa…

Pre: Delete folder with DOS command under Windows 2000 (https://developpaper.com/delete-folder-with-

Next: supporter5. Exe – what process is supporter5 (https://developpaper.com/supporter5-exe-what-

java (https://developpaper.com/question/tag/java/)
php (https://developpaper.com/question/tag/php/)
python (https://developpaper.com/question/tag/python/)
linux (https://developpaper.com/question/tag/linux/)
windows (https://developpaper.com/question/tag/windows/)
android (https://developpaper.com/question/tag/android/)
ios (https://developpaper.com/question/tag/ios/)
mysql (https://developpaper.com/question/tag/mysql/)
html (https://developpaper.com/question/tag/html/)
.net (https://developpaper.com/question/tag/net/)
github (https://developpaper.com/question/tag/github/)
node.js (https://developpaper.com/question/tag/node-js/)

Search