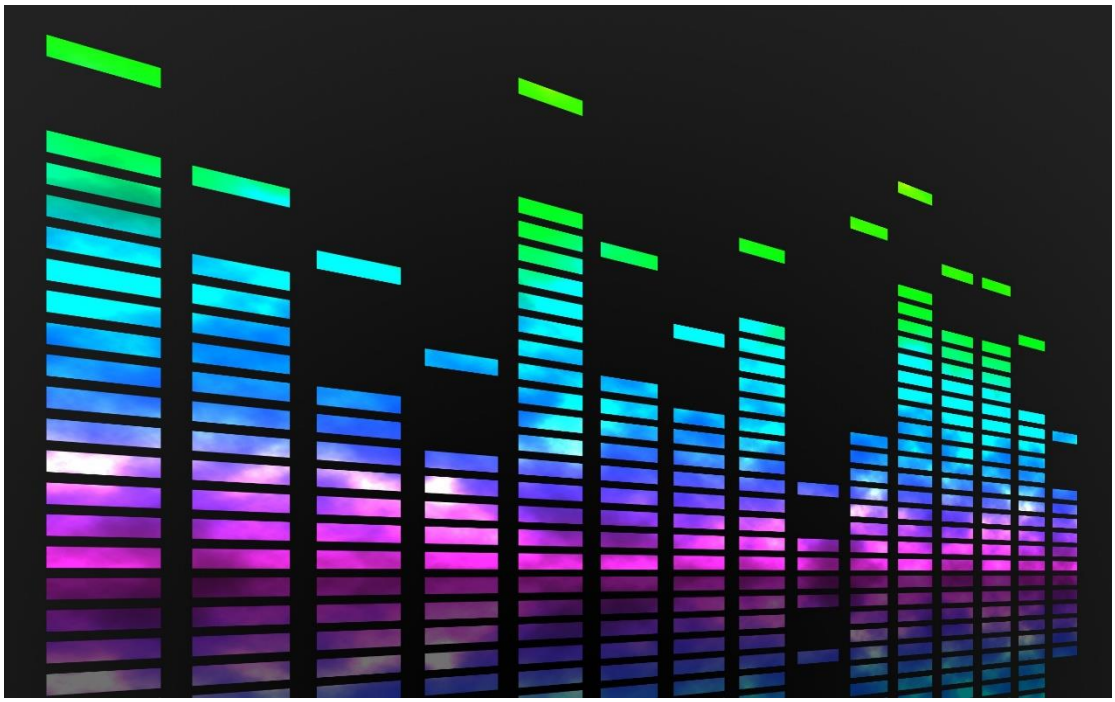
CAB401

Parallelism of Digital Music Analysis

Student number: n9300449

Student Name: Min-Pu Tsai



Contents

CAB401 1
Parallelism of Digital Music Analysis 1
Program Details 3
    Screenshots 4
Program Overview 5
    Class Diagram 5
Turning Digital Time Frequencies into Float Array (and Visualize) 5
Read in the Music Notation and Comparation 7
Paralyze Attempts 8
        Project 1 – Combine two fft() complexity array into one 8
        Project 2 – Adding threads at Possible Locations 9
Usages of Tools & Techniques 11
    – Visual Studio 2017 11
        Diagnostic Tools 11
        Debug -> Performance Profiler 12
        Performance Profiler -> Report -> Create detail report… 12
Reference 14

Program Details

Name: Digital Music Analysis
Language: C#
Development Framework: Microsoft Visual Studio 2017
Brief Description: Digital Music Analysis is a C# Form(Windows) program for new violin learners. It trans an input .wav file which approached by learner and transform it into image (according to its frequency signals that may include Harmonic). Also, it compares to another input file while is a .xml form music score and marks out the errors made by learners. Finally, it divided the Harmonics from main playing by analyze the frequency (Hz) of each sample. Each of these three functions is generated and placed under a correspond tab in the window form interface that allow to brows every particular range of first 2048 samples (for efficiency purpose it adopt only this few samples) of the input sounds.
Development Machine Specifications – Early Hardware: (Own Laptop)

Processer: Intel® Core™ i7-6700HQ CPU @ 2.60GHz 2.59GHz
Number of Cores: 1
Socket: LGA1155
Cache: 6 MB Smart Cache
Clock Speed: 2.60 GHz (maximum 3.50 GHz)
Hyper-Threading: No
Memory: 4GB (3.80GB usable) LPDDR3-1866

For the higher and favorable performant testing, the processer soon changed to QUT campus devices with more information details below:

Development Machine Specifications – Current Hardware: (QUT Machine)

Processer: Intel® Core™ i7-4770S CPU @ 3.10GHz 3.10GHz
Processor Graphics: Intel® HD Graphics 4600
Number of Cores: 4
Number of Threads: 8
Socket: FCLGA1150
Cache: 8 MB Smart Cache
Clock Speed: 3.10 GHz (maximum 3.90 GHz)
Hyper-Threading: No
Memory: 32GB (25.6GB/s maximum Bandwidth) DDR3-1333/1600

Screenshots

(Example sounds and notation pair: *Jupiter.wav* and *Jupiter.xml*)

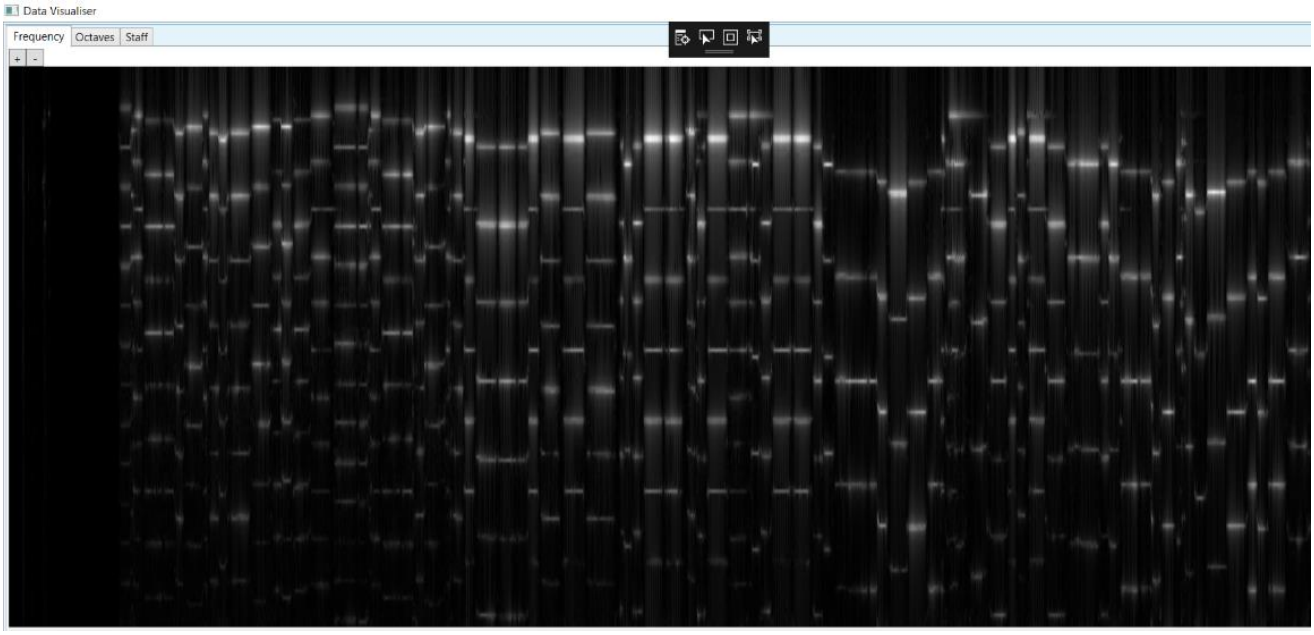


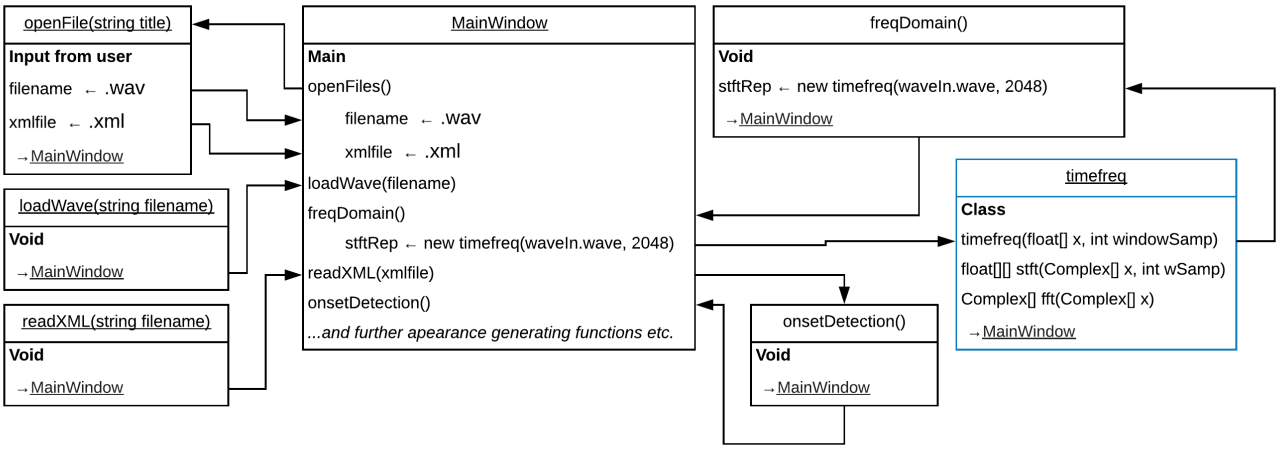
Figure1

Figure2 & Figure3

Program Overview

Class Diagram

This is a class diagram that illustrate the series of classes that read and analyse the digital music and music score form. Those other classes of appearance like images and GUI generating function or classes of music notes details and so on will not be discuss in the graph. In other words, the diagram only contains the main functions such as analyse of the input sounds and comparation with sheet music etc.



As the visualizations, the file open and loading functions are not directly relate to algorithms. Furthermore, the ways to shorten the time users spend on choosing files will not be discuss. Therefore, these parts will also been skip.
Besides, the loading starts by parsing that input sounds file of the name into an array.

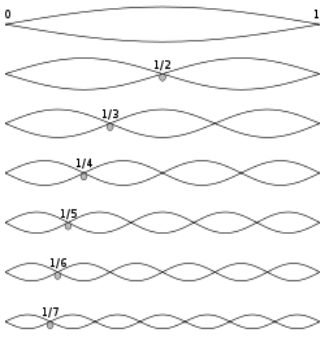
Turning Digital Time Frequencies into Float Array (and Visualize)

After the giving wav file has been read in, freqDomain() would analyse it and transform the digital music signal into necessary float (frequency) array of the shorten 2048 samples. Transform the sounds into frequency domain.

1. First of all, there are 48 samples of each second. The specific 2048 is enough for the purpose of shorten the process and make it clear. In other words, its 2048 samples per second in the case.
2. In an audio, we can hear, or on the other words, recognize the harmonic from whole playing if we listen really closely. Harmonics are those further sounds beside of the octave on operon. It's kind of extra frequency of an audio. The sounds we hear are usually combinations of fundamental frequency and harmonics. Especially we check out by time domain, we will see the graph it

would seems chaos.

3. As the results, if we try to parse the frequencies into frequency domain, the main fundamental frequency and harmonics would be divided more obviously.
4. This float array of frequencies will be helpful while we compare the playing with another input file which is a musical notation in xml form.
5. In conclusion, the timefreq() basically doing most of the word in this part. It creates a complex array and fill it in with input float array, which was transform from input sounds.



Harmonic. Author: Qef.

Illustration of harmonic overtones on the wave set up along a string when it is held steady in certain places, as when a guitar string is plucked while lightly held exactly half way along its length.

6. The stft(), which means and doing the short-term frequency transform on the complex numbers array base on certain sample array, would return a two-dimensional float array. One dimension is for time access, another is for frequency. It was design for knowing the intensity of each frequency in every 2048 samples.
7. fft() divide the conker roll, distinguish the input array into two by the odd and even elements of it.
8. Visualisation. Including amplitude contain time domain in *Figure1*, and frequency domain in *Figure 2*.

```
using System;
using System.Numerics;

namespace DigitalMusicAnalysis{
    public class timefreq{
        public float[][] timeFreqData;
        public int wSamp;
        public Complex[] twiddles;

        public timefreq(float[] x, int windowSamp){
            int ii;
            double pi = 3.14159265;
            Complex i = Complex.ImaginaryOne;
            this.wSamp = windowSamp;
            twiddles = new Complex[wSamp];
            for (ii = 0; ii < wSamp; ii++){
                double a = 2 * pi * ii / (double)wSamp;
                twiddles[ii] = Complex.Pow(Complex.Exp(-i), (float)a);
            }

            timeFreqData = new float[wSamp/2][];

            int nearest = (int)Math.Ceiling((double)x.Length / (double)wSamp);
            nearest = nearest * wSamp;

            Complex[] compX = new Complex[nearest];
            for (int kk = 0; kk < nearest; kk++){
                if (kk < x.Length){
                    compX[kk] = x[kk];
                }else{
                    compX[kk] = Complex.Zero;
                }
            }

            int cols = 2 * nearest /wSamp;

            for (int jj = 0; jj < wSamp / 2; jj++){
                timeFreqData[jj] = new float[cols];
            }
            timeFreqData = stft(compX, wSamp);
        }

        float[][] stft(Complex[] x, int wSamp){
            int ii = 0;
            int jj = 0;
            int kk = 0;
```

```
            int kk = 0;
            int ii = 0;
            int N = x.Length;
            float fftMax = 0;

            float[][] Y = new float[wSamp / 2][];

            for (ii = 0; ii < wSamp / 2; ii++){
                Y[ii] = new float[2 * (int)Math.Floor((double)N / (double)wSamp)];
            }

            Complex[] temp = new Complex[wSamp];
            Complex[] tempFFT = new Complex[wSamp];

            for (ii = 0; ii < 2 * Math.Floor((double)N / (double)wSamp) - 1; ii++){

                for (jj = 0; jj < wSamp; jj++){
                    temp[jj] = x[ii * (wSamp / 2) + jj];
                }

                tempFFT = fft(temp);

                for (kk = 0; kk < wSamp / 2; kk++){
                    Y[kk][ii] = (float)Complex.Abs(tempFFT[kk]);

                    if (Y[kk][ii] > fftMax){
                        fftMax = Y[kk][ii];
                    }
                }
            }

            for (ii = 0; ii < 2 * Math.Floor((double)N / (double)wSamp) - 1; ii++){
                for (kk = 0; kk < wSamp / 2; kk++){
                    Y[kk][ii] /= fftMax;
                }
            }
            return Y;
        }

        Complex[] fft(Complex[] x)

        int ii = 0;
        int kk = 0;
        int N = x.Length;

        Complex[] Y = new Complex[N];
```

```
        Complex[] Y = new Complex[N];

        // NEED TO MEMSET TO ZERO?

        if (N == 1){
            Y[0] = x[0];
        }else{
            Complex[] E = new Complex[N/2];
            Complex[] O = new Complex[N/2];
            Complex[] even = new Complex[N/2];
            Complex[] odd = new Complex[N/2];

            for (ii = 0; ii < N; ii++){

                if (ii % 2 == 0){
                    even[ii / 2] = x[ii];
                }
                if (ii % 2 == 1){
                    odd[(ii - 1) / 2] = x[ii];
                }
            }

            E = fft(even);
            O = fft(odd);

            for (kk = 0; kk < N; kk++){
                Y[kk] = E[(kk % (N / 2))] + O[(kk % (N / 2))] * twiddles[kk * wSamp / N];
            }
        }
        return Y;
    }
}
```

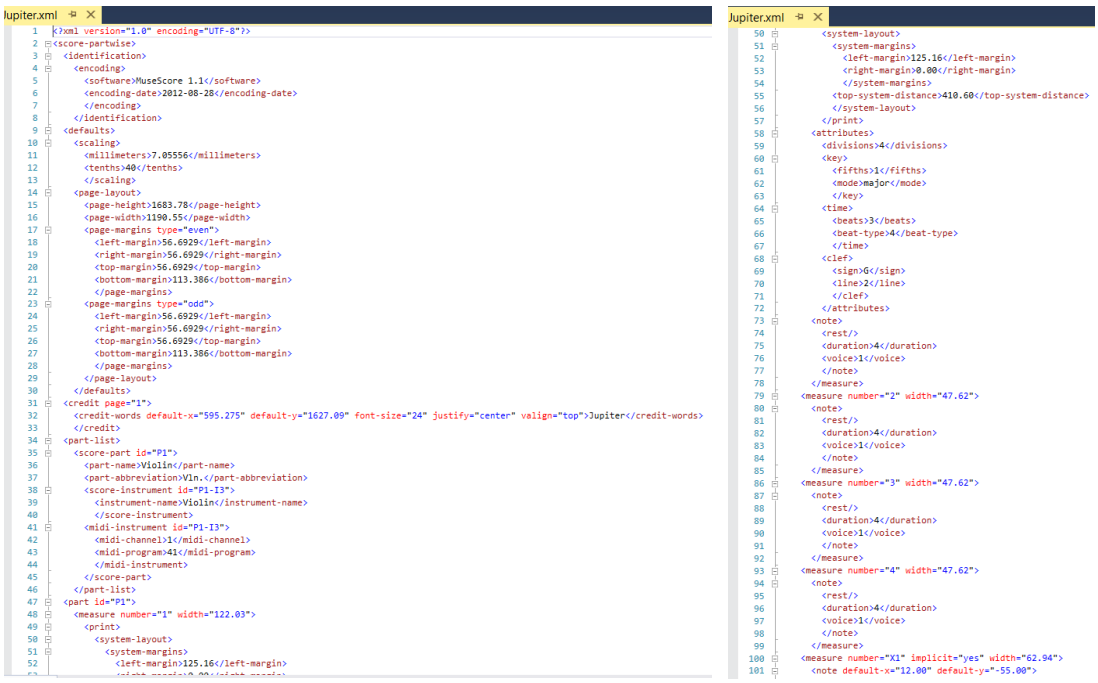
timefreq.cs

Read in the Music Notation and Comparation

The music notation has been translated into xml form already. Containing the frequencies of consider range of octave. Moreover, each note is having their own type, octave and duration comments.

On the other hand, onSetDetection() will detect when a note starts and when the next finish in each period of time. Once its revealed, we can do the full transform base on all the notes in longer term.

After fully transform, the process compares each frequency from wav with xml which both being transform into a series of float values to enable the comparation. And then, visualize as *Figure3*.



Example from Jupiter.xml

The difference between the sounds and the music notation would be mark as a red music notes under the Staff tab of the results Window Form. So does the Actual Frequency and Pitch Error(%) of each single note in the certain range.

Paralyze Attempts

Origin Processing times: 21.38s - 32s *(effect by the time user spend on file selection)*
Average Analyzing times: 6.567s (*details in Project 2*)

Project 1 – Combine two fft() complexity array into one

Statue: Failure

Reason: The requirement of two fft is quite different. Especially the input variable.

Target Process: MainWindow.fft(Complex[] x, int L), timefreq.fft(Complex[] x)

```
Y = new Complex[nearest];

Y = fft(compX, nearest);
//Y = this.//.fft(compX, nearest)
```

```
// FFT function for Pitch Detection
/*private*/ public Complex[] fft(Complex[] x/*, int L*/){
    int ii = 0;
    int kk = 0;
    int N = x.Length;

    Complex[] Y = new Complex[N];

    if (N == 1){
        Y[0] = x[0];
    }else{
        Complex[] E = new Complex[N / 2];
        Complex[] O = new Complex[N / 2];
        Complex[] even = new Complex[N / 2];
        Complex[] odd = new Complex[N / 2];

        for (ii = 0; ii < N; ii++){
            if (ii % 2 == 0){
                even[ii / 2] = x[ii];
            }
            if (ii % 2 == 1){
                odd[(ii - 1) / 2] = x[ii];
            }
```

```
public Complex[] fft(Complex[] x){
    int ii = 0;
    int kk = 0;
    int N = x.Length;

    Complex[] Y = new Complex[N];

    // NEED TO MEMSET TO ZERO?
    if (N == 1){
        Y[0] = x[0];
    }else{
        Complex[] E = new Complex[N/2];
        Complex[] O = new Complex[N/2];
        Complex[] even = new Complex[N/2];
        Complex[] odd = new Complex[N/2];

        for (ii = 0; ii < N; ii++){
```

Extra: Divided the variable into three methods. And let two classes share two of it.

Effect: No improvement.

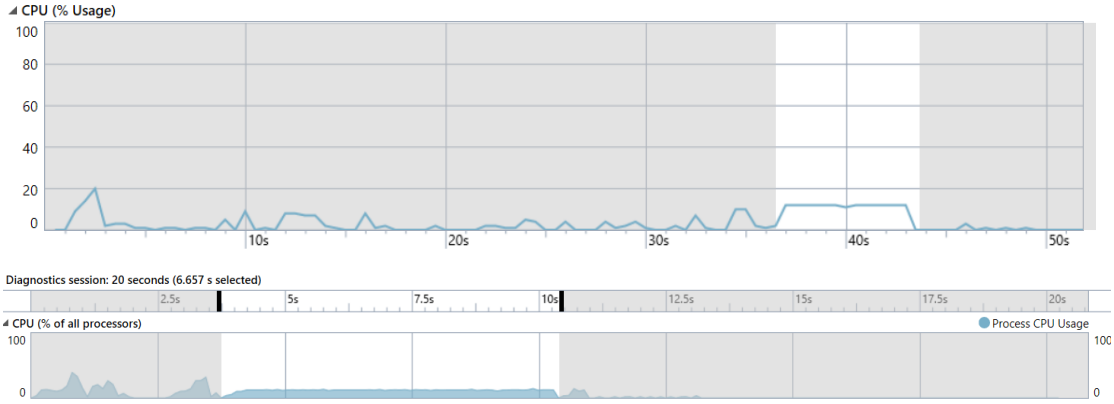
```
public void fft01(Complex[] x){
    int ii = 0;
    N = x.Length;
    Y = new Complex[N];
    if (N == 1){
        Y[0] = x[0];
    }else{
        E = new Complex[N/2];
        O = new Complex[N/2];
        even = new Complex[N/2];
        odd = new Complex[N/2];
        for (ii = 0; ii < N; ii++){
            if (ii % 2 == 0){
                even[ii / 2] = x[ii];
            }
            if (ii % 2 == 1){
                odd[(ii - 1) / 2] = x[ii];
            }
        }
    }
}
```

```
private void kk(Complex[] E, Complex[] O){
    int kk = 0;
    for (kk = 0; kk < N; kk++)
    {
        Y[kk] = E[(kk % (N / 2))] + O[(kk % (N / 2))] * twiddles[kk * wSamp / N];
    }
}

private Complex[] fft(Complex[] x){
    fft01(x);
    E = fft(even);
    O = fft(odd);
    kk(E, O);
    return Y;
}
```

Project 2 – Adding threads at Possible Locations

Statue: Failure

Process: Although there's no way to increase the speed of selecting files (by user), there is always a plateau like line in the graph of CPU usage through time after the selection/file opening is done. It's the analyzation part of whole process. Also, the targets might be able to improve. They are roughly around 6.5 to 7 seconds.

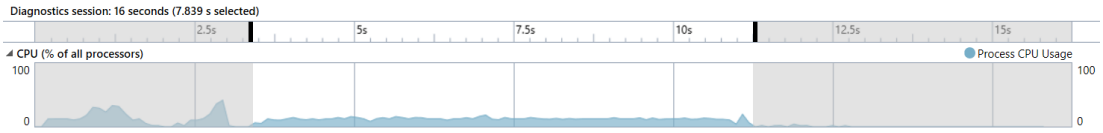


By using the performance profiler functions tools, I locate to some line that may require more resources. (details at Usages of Tools & Techniques)

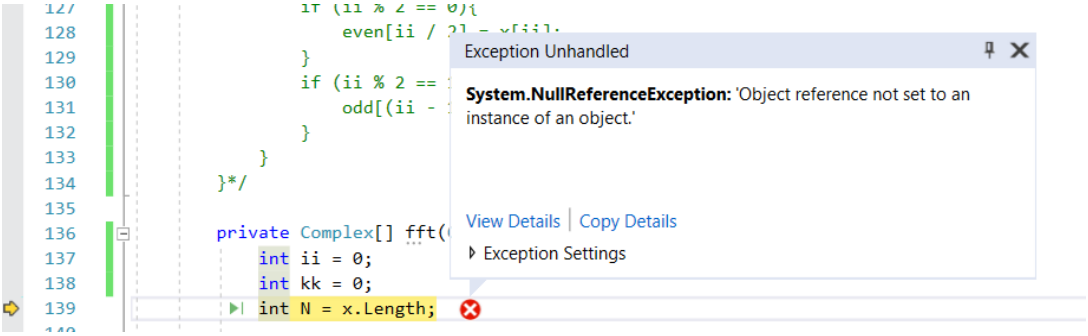
```
//tempFFT = fft(temp);
Thread oThreadA = new Thread(new ThreadStart(tempFFT_eq_fft));
oThreadA.Start();
oThreadA.Join();
```

```
private void tempFFT_eq_fft(){ //threadA
    tempFFT = fft(temp);
}
```

There are several locations I place the threads but mostly cause error. And this thread A cause even more time to process: **7.757s**. I suppose it was because of the .Join(). The waiting makes the thread meaningless if there were no other threads paralyse calculating.



Here are some more failures. Besides of Thread A, there was thread B, C, D and E at timefreq() but all cause problems that makes whole program fail to process.



Exception while trying to add Thread B & C.

```
/*private void threadB(){
    E = fft(even);
}*/
/*private void threadC(){
    O = fft(odd);
}*/
private void threadD(object length){
    int kk = 0;
    int N = Convert.ToInt32(length);
    for (kk = 0; kk < N; kk++)
    {
        Y[kk] = E[(kk % (N / 2))] + O[(kk % (N / 2))] * twiddles[kk * wSamp / N];
    }
}
/*
private void threadE(object length)
{
    int ii = 0;
    int N = Convert.ToInt32(length);
    for (ii = 0; ii < N; ii++){
        if (ii % 2 == 0){
            even[ii / 2] = x[ii];
        }
        if (ii % 2 == 1){
```

Exception Unhandled

System.NullReferenceException: 'Object reference not set to an instance of an object.'

View Details | Copy Details

Exception Settings

Exception while trying to add Thread D.

```
private void tempFFT_eq_fft(){ //threadA
    tempFFT = fft(temp);
}
private void threadB(){
    E = fft(even);
}
private void threadC(){
    O = fft(odd);
}
private void threadD(object length){
    int kk = 0;
    int N = Convert.ToInt32(length);
    for (kk = 0; kk < N; kk++)
    {
        Y[kk] = E[(kk % (N / 2))] + O[(kk % (N / 2))] * twiddles[kk * wSamp / N];
    }
}
private void threadE(object length)
{
    int ii = 0;
    int N = Convert.ToInt32(length);
    for (ii = 0; ii < N; ii++){
        if (ii % 2 == 0){
            even[ii / 2] = x[ii];
        }
        if (ii % 2 == 1){
            odd[(ii - 1) / 2] = x[ii];
        }
    }
}
```

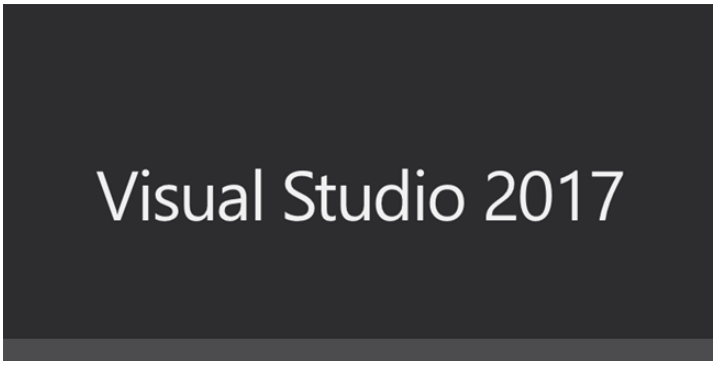
```
Thread oThreadA = new Thread(new ThreadStart(tempFFT_eq_fft));
oThreadA.Start();
oThreadA.Join();
Thread oThreadB = new Thread(new ThreadStart(threadB));
oThreadB.Name = "B Thread";
oThreadB.Start();

Thread oThreadC = new Thread(new ThreadStart(threadC));
oThreadC.Name = "C Thread";
oThreadC.Start();

//E = fft(even);
//O = fft(odd);
oThreadB.Join();
oThreadC.Join();

Thread oThreadD = new Thread(new ParameterizedThreadStart(threadD));
oThreadD.Name = "D Thread";
oThreadD.Start(N);
oThreadD.Join();
```

Thread Functions from A to E.

Conclusion: It was more likely coding errors. If the threading were using in MainWindow(), it might also be different. Although timefreq() do occupied a range of system resources, there suppose also more in main window function.

Usages of Tools & Techniques

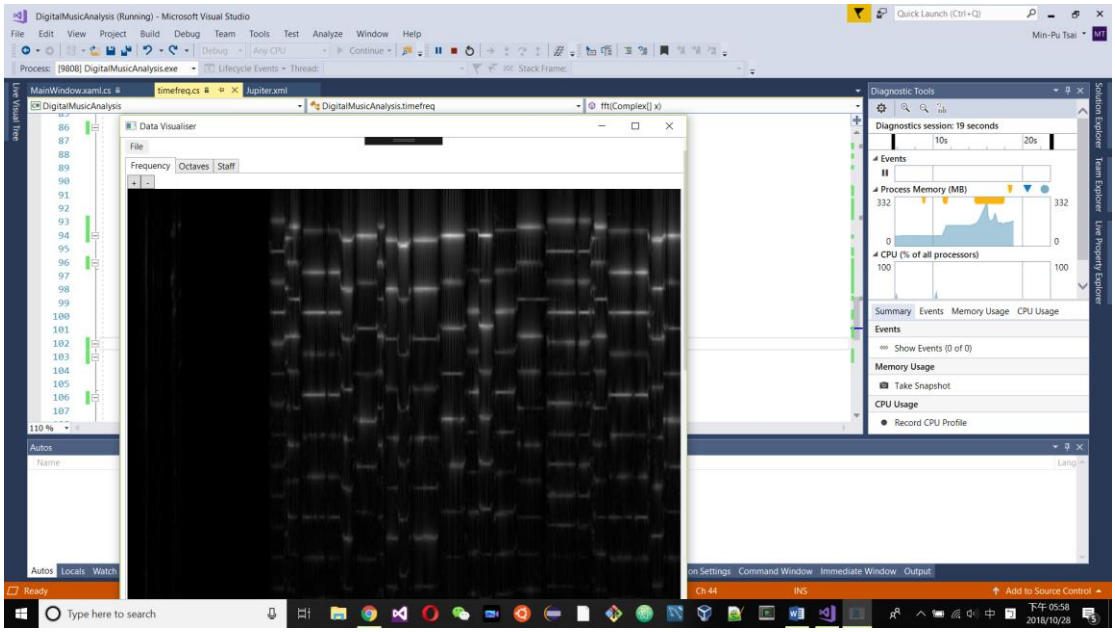
– Visual Studio 2017



To work with C# codes, I chose the **Visual Studio** as common usual but **2017** version in this project. A Full-featured integrated development environment (IDE) for Android, iOS, Windows, web, and cloud. There are several tools are using in the application/work studio which contain multiple high functional profiling, debugging and paralysing tools.

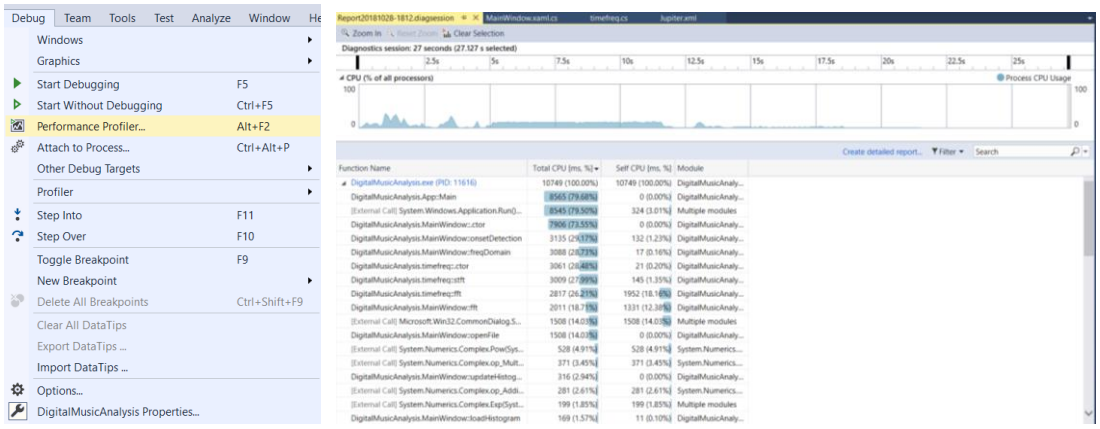
Diagnostic Tools

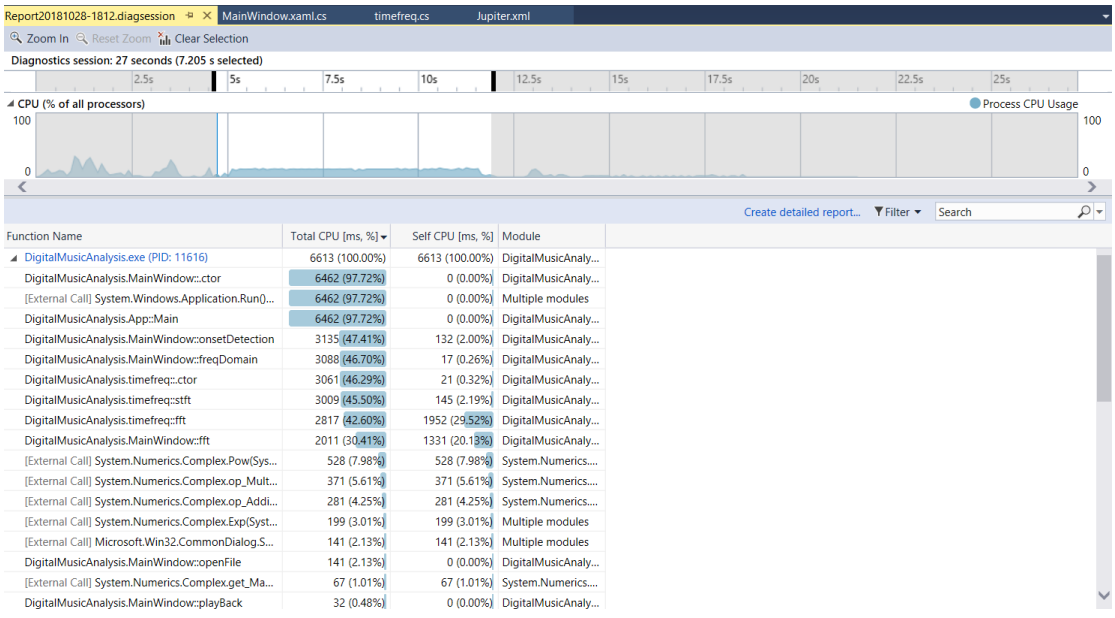
As part of test running function of it, **Diagnostic Tools** provided some satisfice information for common debugging such as Process Memory(MB) usage through times.



Debug-> Performance Profiler

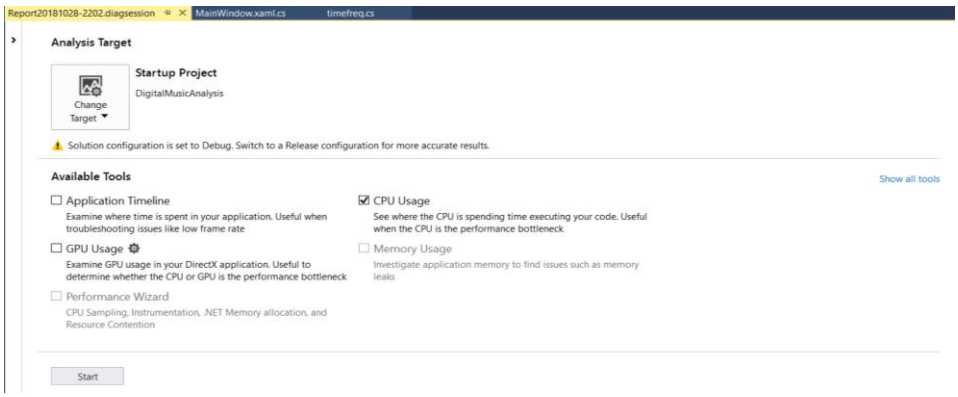
The **Performance Profiler** shows the CPU usages through times and the occupation percentage of those methods that listed from large to small. It also allow user to choose the certain rage of time to shows the statue of methods and CPU usage in the period.

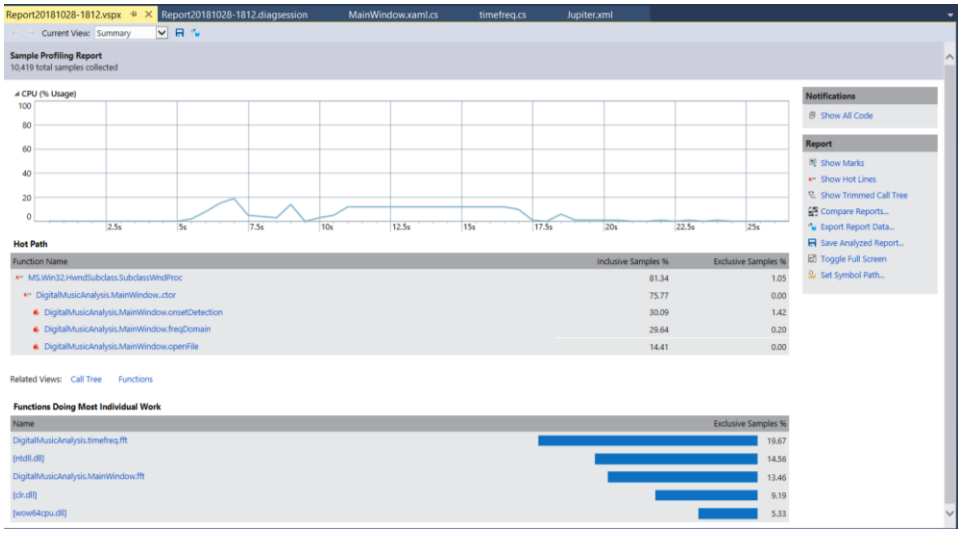




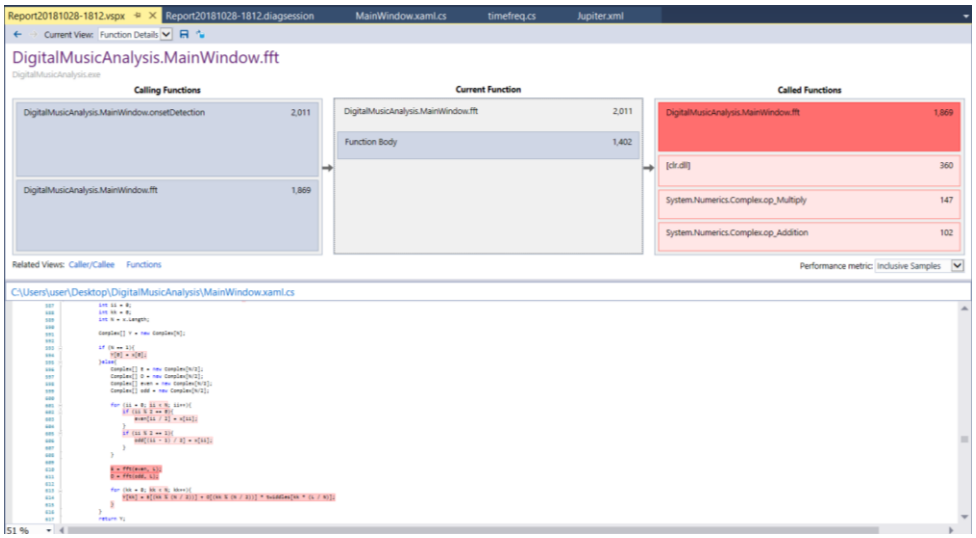
Performance Profiler-> Report-> Create detail report...

After the operation as this title, Visual Studio 2017 shows the **Sample Profiling Report** with the numbers of total samples collected. (Also Call Tree and Function selection related view but they did not used in this project so will not be mention below.) Not mention the graph shows CPU usage through times just as tools above. It shows up the list of functions that doing most individual works and the hot path link to the details of certain function and their samples percentage.





For example, if we take a look at DigitalMusicAnalysis.MainWindow, it shows up the DigitalMusicAnalysis.MainWindow.fft cause most system resource (samples inclusion/CPU usage/memory occupation). The details up to the affection of efficiency to single line. The more it affects the deeper red back colour highlight on it. By this tool, it would be clearly shows the modification we can edit to improve the efficiency of whole process.



Reference

Online Code Location of this assignment. github.com. CAB401-Assignment. https://github.com/a3698147852/CAB401-Assignment.git

See you in 2015! ”Equalizer”, Online Image, Quelab, 31 December 2014, http://quelab.net/blog/673/mays-theme-audio/

Visual Studio 2017. Microsoft. https://visualstudio.microsoft.com/vs/

Blackboard Example Report Form Reference: ‘Ossudo’ Parallelization Report. Mark Jones.https://blackboard.qut.edu.au/courses/1/CAB401_18se2/content/_7470201_1/Sudoku%20MicrosoftTPL.pdf

Intel® Core™ i7-6700HQ Processor. Intel. https://ark.intel.com/products/88967/Intel-Core-i7-6700HQ-Processor-6M-Cache-up-to-3-50-GHz-

Intel® Core™ i7-4770S Processor. Intel. https://ark.intel.com/products/75124/Intel-Core-i7-4770S-Processor-8M-Cache-up-to-3-90-GHz-

Harmonic partials on strings.svg. Author: Qef. https://commons.wikimedia.org/wiki/File:Harmonic_partials_on_strings.svg