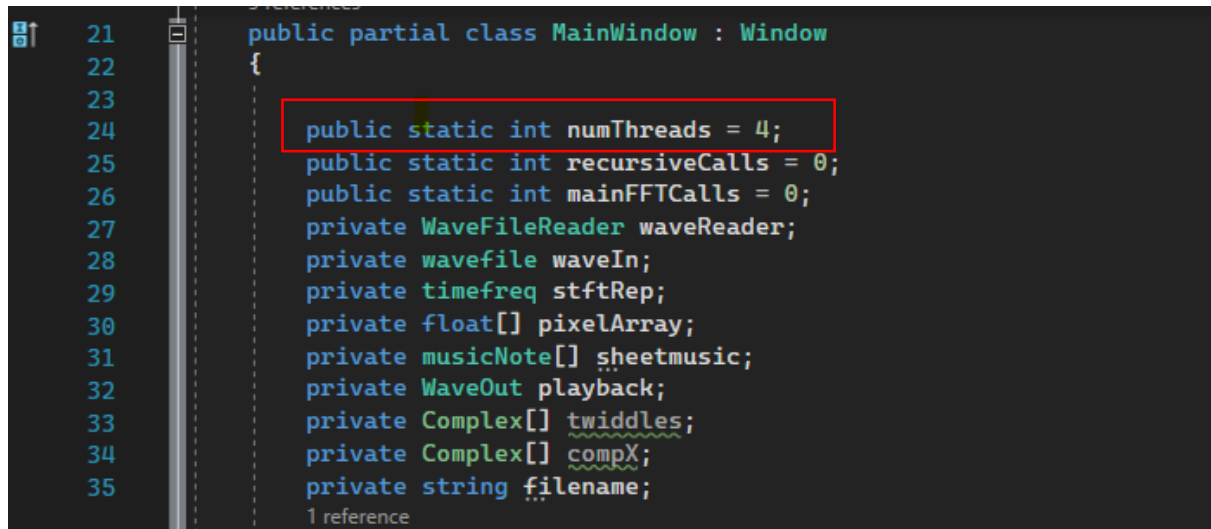


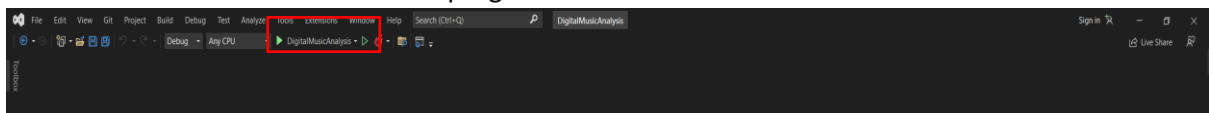
Instructions

1. There are two different folders that will be submitted. “DigitalMusicAnalysis” contains the sequential version of the code (with a few stopwatches added) and “DigitalMusciAnalysisParallel” contains the parallel code. However, both will be ran in a similar manner.
2. The parallel version of the code base has a global variable that will specify the number of threads used for the program.



```
21 public partial class MainWindow : Window
22 {
23
24     public static int numThreads = 4;
25     public static int recursiveCalls = 0;
26     public static int mainFFTCalls = 0;
27     private WaveFileReader waveReader;
28     private wavefile waveIn;
29     private timefreq stftRep;
30     private float[] pixelArray;
31     private musicNote[] sheetmusic;
32     private WaveOut playback;
33     private Complex[] twiddles;
34     private Complex[] compX;
35     private string filename;
    1 reference
```

3. Select the “run” to build and run the program



4. A file explorer prompt will appear and request that you put in your .wav file, then another file explorer prompt will appear asking for your .xml file.
5. The program should now begin the computation and will shortly playing the music as well as displaying the various graphs.

Hardware Requirements

This program has only been run on an x86 Windows10 environment. It is recommended to run with 4 or more cores to see the performance increase mentioned in the report.

Input Data sets

Throughout testing “Jupiter.wav” and “Jupiter.xml” files were used however, any of the other .wav files will work provided they have a corresponding xml files.