MainWindow.xaml.cs Line ~313

// Onset Detection function - Determines Start and Finish times of a note and the frequency of the note over each duration.

public void onsetDetection()

{

float[] HFC;

int starts = 0;

int stops = 0;

Complex[] Y;

double[] absY;

List<int> lengths;

List<int> noteStarts;

List<int> noteStops;

List<double> pitches;

int ll;

double pi = 3.14159265;

Complex i = Complex.ImaginaryOne;

noteStarts = new List<int>(100);

noteStops = new List<int>(100);

lengths = new List<int>(100);

pitches = new List<double>(100);

SolidColorBrush sheetBrush = new SolidColorBrush(Colors.Black);

SolidColorBrush ErrorBrush = new SolidColorBrush(Colors.Red);

SolidColorBrush whiteBrush = new SolidColorBrush(Colors.White);

HFC = new float[stftRep.timeFreqData[0].Length];

for (int jj = 0; jj < stftRep.timeFreqData[0].Length; jj++)

{

for (int ii = 0; ii < stftRep.wSamp / 2; ii++)

{

HFC[jj] = HFC[jj] + (float)Math.Pow((double)stftRep.timeFreqData[ii][jj] \* ii, 2);

}

}

float maxi = HFC.Max();

for (int jj = 0; jj < stftRep.timeFreqData[0].Length; jj++)

{

HFC[jj] = (float)Math.Pow((HFC[jj] / maxi), 2);

}

for (int jj = 0; jj < stftRep.timeFreqData[0].Length; jj++)

{

if (starts > stops)

{

if (HFC[jj] < 0.001)

{

noteStops.Add(jj \* ((stftRep.wSamp - 1) / 2));

stops = stops + 1;

}

}

else if (starts - stops == 0)

{

if (HFC[jj] > 0.001)

{

noteStarts.Add(jj \* ((stftRep.wSamp - 1) / 2));

starts = starts + 1;

}

}

}

if (starts > stops)

{

noteStops.Add(waveIn.data.Length);

}

// DETERMINES START AND FINISH TIME OF NOTES BASED ON ONSET DETECTION

///\*

for (int ii = 0; ii < noteStops.Count; ii++)

{

lengths.Add(noteStops[ii] - noteStarts[ii]);

}

for (int mm = 0; mm < lengths.Count; mm++)

{

int nearest = (int)Math.Pow(2, Math.Ceiling(Math.Log(lengths[mm], 2)));

twiddles = new Complex[nearest];

for (ll = 0; ll < nearest; ll++)

{

double a = 2 \* pi \* ll / (double)nearest;

twiddles[ll] = Complex.Pow(Complex.Exp(-i), (float)a);

}

compX = new Complex[nearest];

for (int kk = 0; kk < nearest; kk++)

{

if (kk < lengths[mm] && (noteStarts[mm] + kk) < waveIn.wave.Length)

{

compX[kk] = waveIn.wave[noteStarts[mm] + kk];

}

else

{

compX[kk] = Complex.Zero;

}

}

Y = new Complex[nearest];

Y = fft(compX, nearest);

absY = new double[nearest];

double maximum = 0;

int maxInd = 0;

for (int jj = 0; jj < Y.Length; jj++)

{

absY[jj] = Y[jj].Magnitude;

if (absY[jj] > maximum)

{

maximum = absY[jj];

maxInd = jj;

}

}

for (int div = 6; div > 1; div--)

{

if (maxInd > nearest / 2)

{

if (absY[(int)Math.Floor((double)(nearest - maxInd) / div)] / absY[(maxInd)] > 0.10)

{

maxInd = (nearest - maxInd) / div;

}

}

else

{

if (absY[(int)Math.Floor((double)maxInd / div)] / absY[(maxInd)] > 0.10)

{

maxInd = maxInd / div;

}

}

}

if (maxInd > nearest / 2)

{

pitches.Add((nearest - maxInd) \* waveIn.SampleRate / nearest);

}

else

{

pitches.Add(maxInd \* waveIn.SampleRate / nearest);

}

}

musicNote[] noteArray;

noteArray = new musicNote[noteStarts.Count()];

for (int ii = 0; ii < noteStarts.Count(); ii++)

{

noteArray[ii] = new musicNote(pitches[ii], lengths[ii]);

}

int[] sheetPitchArray = new int[sheetmusic.Length];

int[] notePitchArray = new int[noteArray.Length];

for (int ii = 0; ii < sheetmusic.Length; ii++)

{

sheetPitchArray[ii] = sheetmusic[ii].pitch % 12;

}

for (int jj = 0; jj < noteArray.Length; jj++)

{

notePitchArray[jj] = noteArray[jj].pitch % 12;

}

string[] alignedStrings = new string[2];

alignedStrings = stringMatch(sheetPitchArray, notePitchArray);

musicNote[] alignedStaffArray = new musicNote[alignedStrings[0].Length / 2];

musicNote[] alignedNoteArray = new musicNote[alignedStrings[1].Length / 2];

int staffCount = 0;

int noteCount = 0;

for (int ii = 0; ii < alignedStrings[0].Length / 2; ii++)

{

if (alignedStrings[0][2 \* ii] == ' ')

{

alignedStaffArray[ii] = new musicNote(0, 0);

}

else

{

alignedStaffArray[ii] = sheetmusic[staffCount];

staffCount++;

}

if (alignedStrings[1][2 \* ii] == ' ')

{

alignedNoteArray[ii] = new musicNote(0, 0);

}

else

{

alignedNoteArray[ii] = noteArray[noteCount];

noteCount++;

}

}

// STAFF TAB DISPLAY

Ellipse[] notes;

Line[] stems;

notes = new Ellipse[alignedNoteArray.Length];

stems = new Line[alignedNoteArray.Length];

SolidColorBrush myBrush = new SolidColorBrush(Colors.Green);

RotateTransform rotate = new RotateTransform(45);

for (int ii = 0; ii < alignedNoteArray.Length; ii++)

{

//noteArray[ii] = new musicNote(pitches[ii], lengths[ii]);

//System.Console.Out.Write("Note " + (ii + 1) + ": \nDuration: " + noteArray[ii].duration / waveIn.SampleRate + " seconds \nPitch: " + Enum.GetName(typeof(musicNote.notePitch), (noteArray[ii].pitch) % 12) + " / " + pitches[ii] + "\nError: " + noteArray[ii].error \* 100 + "%\n");

notes[ii] = new Ellipse();

notes[ii].Tag = alignedNoteArray[ii];

notes[ii].Height = 20;

notes[ii].Width = 15;

notes[ii].Margin = new Thickness(ii \* 30, 0, 0, 0);

notes[ii].LayoutTransform = rotate;

notes[ii].MouseEnter += DisplayStats;

notes[ii].MouseLeave += ClearStats;

stems[ii] = new Line();

stems[ii].StrokeThickness = 1;

stems[ii].X1 = ii \* 30 + 20;

stems[ii].X2 = ii \* 30 + 20;

stems[ii].Y1 = 250 - 10 \* alignedNoteArray[ii].staffPos;

stems[ii].Y2 = 250 - 10 \* alignedNoteArray[ii].staffPos - 40;

notes[ii].Fill = ErrorBrush;

notes[ii].StrokeThickness = 1;

stems[ii].Stroke = ErrorBrush;

Canvas.SetTop(notes[ii], (240 - 10 \* alignedNoteArray[ii].staffPos));

if (alignedNoteArray[ii].flat)

{

System.Windows.Controls.Label flat = new System.Windows.Controls.Label();

flat.Content = "b";

flat.FontFamily = new FontFamily("Mistral");

flat.Margin = new Thickness(ii \* 30 + 15, 0, 0, 0);

Canvas.SetTop(flat, (240 - 10 \* alignedNoteArray[ii].staffPos));

noteStaff.Children.Insert(ii, flat);

}

noteStaff.Children.Insert(ii, notes[ii]);

noteStaff.Children.Insert(ii, stems[ii]);

}

Ellipse[] sheetNotes;

Rectangle[] timeRect;

Line[] sheetStems;

sheetNotes = new Ellipse[alignedStaffArray.Length];

sheetStems = new Line[alignedStaffArray.Length];

timeRect = new Rectangle[2 \* alignedStaffArray.Length];

Fline.Width = alignedStaffArray.Length \* 30;

Dline.Width = alignedStaffArray.Length \* 30;

Bline.Width = alignedStaffArray.Length \* 30;

Gline.Width = alignedStaffArray.Length \* 30;

Eline.Width = alignedStaffArray.Length \* 30;

noteStaff.Width = alignedStaffArray.Length \* 30;

for (int ii = 0; ii < alignedStaffArray.Length; ii++)

{

sheetNotes[ii] = new Ellipse();

sheetNotes[ii].Tag = alignedStaffArray[ii];

sheetNotes[ii].Height = 20;

sheetNotes[ii].Width = 15;

sheetNotes[ii].Margin = new Thickness(ii \* 30, 0, 0, 0);

sheetNotes[ii].LayoutTransform = rotate;

sheetNotes[ii].MouseEnter += DisplayStats;

sheetNotes[ii].MouseLeave += ClearStats;

sheetStems[ii] = new Line();

sheetStems[ii].StrokeThickness = 1;

sheetStems[ii].X1 = ii \* 30 + 20;

sheetStems[ii].X2 = ii \* 30 + 20;

sheetStems[ii].Y1 = 250 - 10 \* alignedStaffArray[ii].staffPos;

sheetStems[ii].Y2 = 250 - 10 \* alignedStaffArray[ii].staffPos - 40;

sheetNotes[ii].Fill = sheetBrush;

sheetNotes[ii].StrokeThickness = 1;

sheetStems[ii].Stroke = sheetBrush;

Canvas.SetTop(sheetNotes[ii], (240 - 10 \* alignedStaffArray[ii].staffPos));

if (alignedStaffArray[ii].flat)

{

System.Windows.Controls.Label flat = new System.Windows.Controls.Label();

flat.Content = "b";

flat.FontFamily = new FontFamily("Mistral");

flat.Margin = new Thickness(ii \* 30 + 15, 0, 0, 0);

Canvas.SetTop(flat, (240 - 10 \* alignedStaffArray[ii].staffPos));

noteStaff.Children.Insert(ii, flat);

}

noteStaff.Children.Insert(ii, sheetNotes[ii]);

noteStaff.Children.Insert(ii, sheetStems[ii]);

}

// FOR TIMING ERROR RECTANGLES

for (int ii = 0; ii < alignedStaffArray.Length; ii++)

{

timeRect[ii] = new Rectangle();

timeRect[ii].Fill = sheetBrush;

timeRect[ii].Height = 10 \* alignedStaffArray[ii].duration \* 4 \* bpm / (60 \* waveIn.SampleRate);

timeRect[ii].Width = 15;

timeRect[ii].Margin = new Thickness(ii \* 30 + 5, 0, 0, 0);

Canvas.SetTop(timeRect[ii], 200);

noteStaff.Children.Insert(ii, timeRect[ii]);

}

for (int ii = alignedStaffArray.Length; ii < alignedStaffArray.Length + alignedNoteArray.Length; ii++)

{

timeRect[ii] = new Rectangle();

timeRect[ii].Fill = ErrorBrush;

timeRect[ii].Height = 10 \* alignedNoteArray[ii - alignedStaffArray.Length].duration \* 4 \* bpm / (60 \* waveIn.SampleRate);

timeRect[ii].Width = 10;

timeRect[ii].Margin = new Thickness((ii - alignedStaffArray.Length) \* 30 + 5, 0, 0, 0);

Canvas.SetTop(timeRect[ii], 200);

noteStaff.Children.Insert(ii, timeRect[ii]);

}

}