4 januari 2016

Dwight Van der Velpen 2EO2

Odisee

Wetenschappelijke toepassingen 1

Workshop 2: Fourier

Inhoudsopgave

[1. Research: 2](#_Toc439712581)

[1.1. Onderzoek wat een WAV File in elkaar zit. 2](#_Toc439712582)

[1.2. Wat krijgen we, wat willen we. 3](#_Toc439712583)

[1.3. CSV bestanden 3](#_Toc439712584)

[1.4. Fourier 3](#_Toc439712585)

[2. Opbouw Project en schema 5](#_Toc439712586)

[2.1. GUI 5](#_Toc439712587)

[2.2 Logica 5](#_Toc439712588)

[2.2.1 Basis Logica 5](#_Toc439712589)

[2.2.2 Fourier 6](#_Toc439712590)

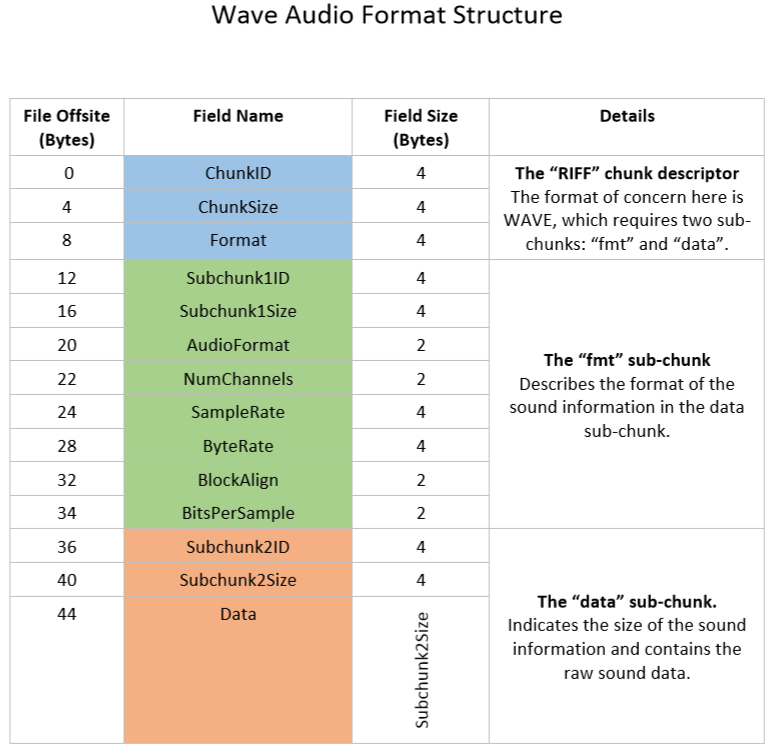
[2.3 Data 6](#_Toc439712591)

[2.4 schema 6](#_Toc439712592)

[Verwijzingen 7](#_Toc439712593)

# Research:

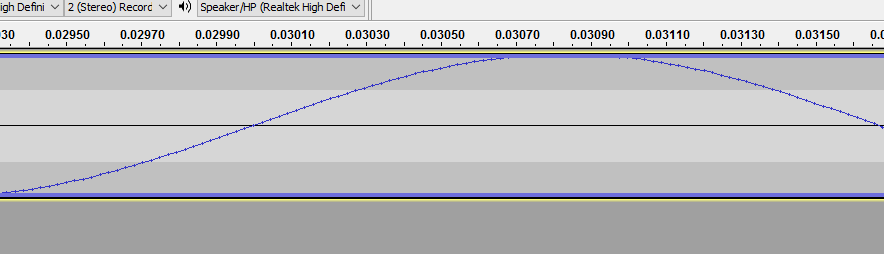
## Onderzoek wat een WAV File in elkaar zit.

Voor ik begon met het kijken naar hoe de code zou werken heb ik besloten me te verdiepen in hoe deze opgeslagen zou moeten worden en hoe een “.WAV” file er in algemeen uitziet aangezien alles wat we in dit project genereren in een WAV file wordt opgeslagen. Hiervoor heb ik beroep gedaan op de MSDN tutorial deel 1 en 2 die zich verdiepen in hoe audio en WAV files werken.  
  
In een noten dop zou je kunnen zeggen dat het bestaat uit 3 grote onderdelen (chunks) die elk starten met een ID;

* Header, ·waar dat de extensie (format) en grote van het volledige bestand is.
* Format,  
  waar we de instellingen geven (kanalen, sample rate…)
* En een data deel,  
  waar de waarden van je sinus zich in zullen bevinden.

De data word in een SHORT-array opgeslagen die per sample (kleine stap in tijd) de waarde (hoogte).

Dit is wegens we geen perfecte sinus, (sample rate oneindig), kunnen gebruiken. We nemen kleine stappen die als je ze zou verbinden een gesimuleerde sinus geven. De “depth” in combinatie van het aantal samples geeft u dan een betere representatie van uw sinus.



In dit project is het nuttiger om niet verder te verdiepen op wat elk veld nu doet maar is het wel nuttig om te begrijpen welke velden er belangrijk zijn.

Sample rate, data, channels, block size en data size.

## Wat krijgen we, wat willen we.

Nadat we de WAV file hebben uitgelezen hebben we dus enkel een short List met elk sample.  
Maar we zouden ideaal willen:

* Dat elk karakter (toon) apart in een float ARRAY staat
* Dat elk float array een lengte heeft dat een macht is van 2 (zie stap FOURIER)

Daarom gaan we in enkele simpele stappen dit regelen.

1. Zet list<short> met alle samples om naar een list<list<short>> die per noot zijn samples heeft.
2. Zet het list<list<short>> om naar een <list<list<float>>
3. Zet de list<list<float>> om naar een list<float[ ]> die van juiste lengte is

Op deze float[]’s kunnen we nu een Fourier analyse doen en zo de samenstellingen van frequenties bepalen. Deze gaan we dan filteren en de dominante frequentie uitkiezen.

A.d.h.v deze frequentie kunnen we dan in ons CSV bestand kijken met welke letter deze geassocieerd was en zo de boodschap in de WAV achterhalen.

## CSV bestanden

Een CSV bestand is een simpel bestandje waarin je een tabel kan in opslaan. De kolommen worden voorgesteld als komma’s (Comma Seperated Values) en de rijen door New Lines. Voor dit project moest je je voorstelling van letters tegenover frequenties halen uit een CSV bestand. Dit heeft als voordelen dat je indien je elke letter kan specifiëren on the fly i.p.v. dat je deze in je code zelf moet aanpassen.

Een CSV bestand uitlezen en opsplitsen valt zeer goed mee, het is net zoals je een Tekst bestand zou uitlezen (Binary Reader) en lijn per lijn lezen. Deze lijnen kan je dan met String.Trim opkuisen (spaties verwijderen) en met splitsen op ‘,’ of ‘;’.

Ook heb ik er in toegevoegd dat je geen lijnen inleest die beginnen met //. Dit zorgt ervoor dat commentaar zou kunnen voorzien worden.

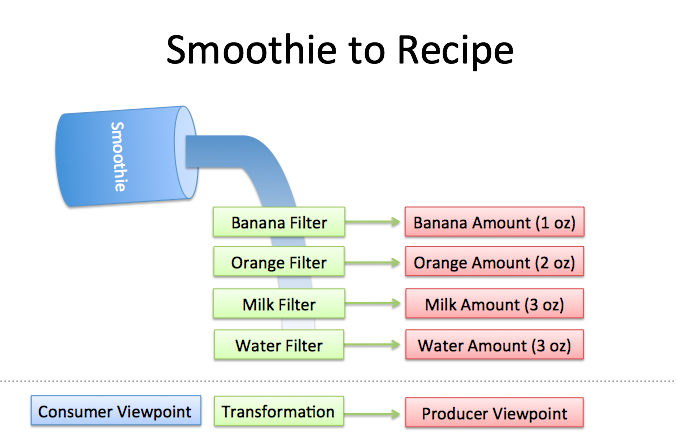
## Fourier

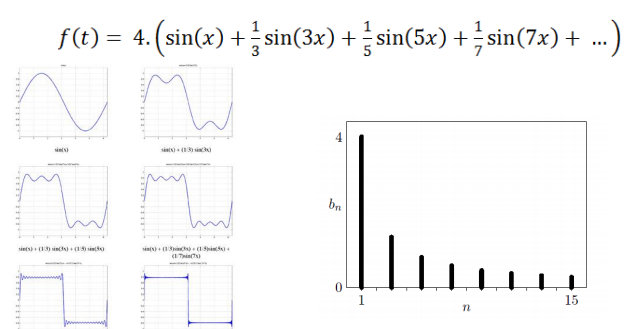
Waarvoor wordt Fourier gebruikt. Dat is de eerste vraag. Ik baseer deze uitleg vooral op “betterExplained.com”’s artikel over Fourier wegens deze alles veel eenvoudiger maakt voor niet wiskundigen het principe te begrijpen.

In het Nederlands:

* **Wat doet een Fourier analyse?** Als je een smoothie geeft, krijg je de ingrediënten.
* **Hoe?** Door de smoothie te filteren en zo een recept krijgen van ingrediënten.
* **Waarom?** Omdat als elke smoothie uit ingrediënten bestaat kunnen we welke smoothie ook simpel als een recept schijven. Ze lezen makkelijker en je kan ze via recept ook makkelijker vergelijken en veranderen.

In het wiskundige:

* Een Fourier analyse neemt een tijds gebaseerd repetitief signaal, meet elk mogelijke repetitie, en geeft je een signaal recept. (De verschillende ingrediënten [sinussen], hun hoeveelheid[amplitude en offset] en volgorde[frequenties])

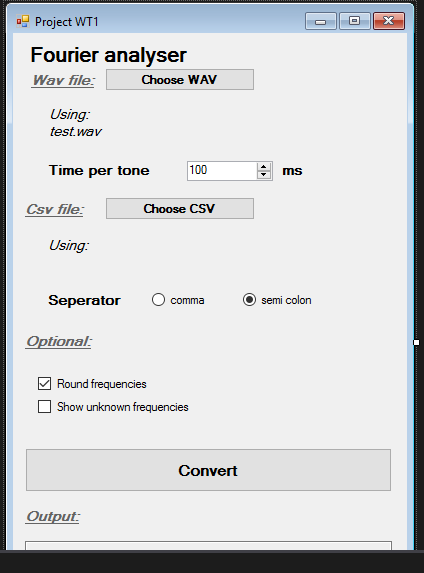


# Opbouw Project en schema

## GUI

In je Gui heb je de interface met volgende gebruik invoer methoden:

* Een knop om je WAV file te selecteren en een NumericUpDown voor de toonTijd
* Een knop om je CSV file te selecteren en 2 radio buttons die je separator kiezen
* Opties om je frequenties af te ronden (aangeraden wegens kleine fouten (+-1Hz)
* Optie om je niet herkende frequenties terug te geven
* Een convert knop die met huidige opties je aan de slag gaat
* Een output veld



## 2.2 Logica

### 2.2.1 Basis Logica

Deze klasse wordt maar voor 2 reden betreden. Of de gebruiker heeft een Input file verandert. Of de convert knop werd ingedrukt.

Deze klasse bevat daarom een doorverwijzing naar de Backend (Data) en enkele reken methoden. Je zou een punt kunnen maken dat je deze kunt splitsen maar hiervoor is net niet gekozen wegens deze klasse weinig doet. Het convert wat met lists en arrays, roept de Fourier analyse op en zet deze dan om naar een string. Ik zou hier per conversie een methode kunnen maken maar methoden voor 1 foreach loop lijken me wat overkill.

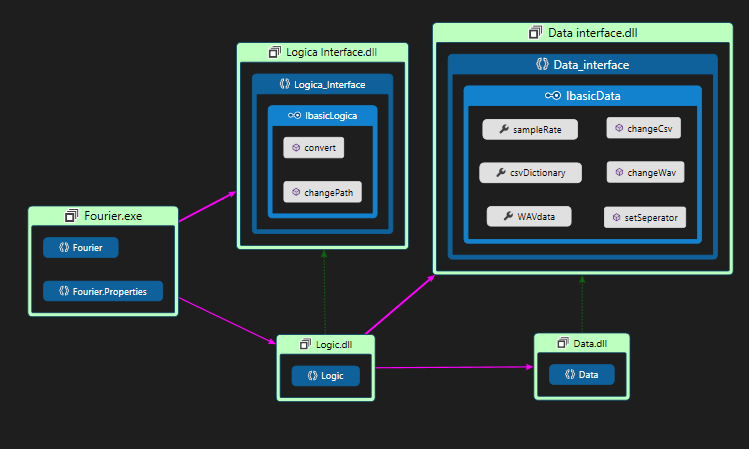
### 2.2.2 Fourier

Dit is een meegekregen klasse die met veel interne logica een Fourier analyse toepast op een floatarray.

## 2.3 Data

Hierin worden 2 bestandspaden opgeslagen en zijn er enkele publieke velden voor data die nodig zou kunnen zijn in de logica laag. Bij oproep van deze velden zal het bestand worden uitgelezen. Dit is niet de meest resource vriendelijke toepassing maar dit zorgt er wel voor dat er volgens de Logica laag er gewoon velden zijn. De logica beveelt dus niet wat er moet gebeuren. Hij vraagt enkel.

## 2.4 schema



# Verwijzingen

1. **(stack overflow), Andrew Hare.** *Simple CSV reader.* [Forum] http://stackoverflow.com/questions/1375410/very-simple-c-sharp-csv-reader : sn, 3 September 2009.

2. **(msdn blogs), dawate.** *Intro to audio programming part 1-3.* [Guide] http://blogs.msdn.com/b/dawate/archive/2009/06/24/intro-to-audio-programming-part-3-synthesizing-simple-wave-audio-using-c.aspx : sn, 24 juni 2009.

3. **Microsoft.** *Library.* [Code Library] https://msdn.microsoft.com/en-us/library : sn.

4. **Donné, Johan.** *C# Reference Guide.* sl : Odisee, 2015.

5. **phy mtu.** Physics of music. *phy.mtu.edu.* [Online] 1998-2015. http://www.phy.mtu.edu/~suits/notefreqs.html.

6. **Tutorialspoint.com.** CompileCodeFreeInline. http://www.tutorialspoint.com/compile\_csharp\_online.php : sn.

7. **An Interactive Guide To The Fourier Transform. *BetterExplained.com.* [Online] http://betterexplained.com/articles/an-interactive-guide-to-the-fourier-transform/.**