Project 2:

Wav Scrambeler

Glenn Latomme  
2 ICT

Inhoud

[Background 2](#_Toc341027303)

[Objectives 2](#_Toc341027304)

[Overview 2](#_Toc341027305)

[Progress 2](#_Toc341027306)

[Design 2](#_Toc341027307)

[Code 3](#_Toc341027308)

[GUI 3](#_Toc341027309)

[GUI - No border 3](#_Toc341027310)

[GUI - Slider 4](#_Toc341027311)

[Music player 4](#_Toc341027312)

[Recommendation 4](#_Toc341027313)

[Recommendation - Extra programma 5](#_Toc341027314)

[Recommendation - Ondersteunende code 5](#_Toc341027315)

[Recommendation - Engine 5](#_Toc341027316)

[andere plannen 6](#_Toc341027317)

[REFLECTIE 6](#_Toc341027318)

[Bibliografie 7](#_Toc341027319)

# Background

Toen ik hier aan begon, was het allemaal beetje vaag van hoe en wat er verwacht werd, en hoe we dit moesten bereiken, maar naargelang wat verdere research te doen op de opgave, was dit zo vaag omdat we enorm de vrijheid kregen om te doen wat we wouden, zolang die wav files er in voor kwamen en er gebruik werd gemaakt van de Fourier.cs

# Objectives

## Overview

Hier was mijn doel om een basic equalizer te maken, en terwijl ook bij te leren over fourier, aangezien dit één van de weinige dingen van muziek dat ik altijd al had eens willen aanraken, maar nooit echt in geslaagd ben om iets deftig er uit te halen

## Progress

Hier ben ik vergeleken met men vorig project iets anders te werk gegaan. In het vorige ben ik begonnen vanaf de design, om zo al een ID te hebben van wat ik wil bereiken en zo stuk voor stuk de dingen wegschrappen. Heb ik hier mijn doelstelling gelegd om Fourier te snappen en deze goed toe te passen op een basis programma

### Testing out

#### TestFile1

Eerst moesten we de wav file inlezen, dit leek op het eerste gezicht een gemakkelijk iets, en wat ik deed werkte ook.  
maar dan begon ik er fourier op te steken, en daar liep alles in het 100, daar begon ik al lichtjes door te hebben dat niet alles zo gemakkelijk ging gaan dan verwacht

#### TestFIle2

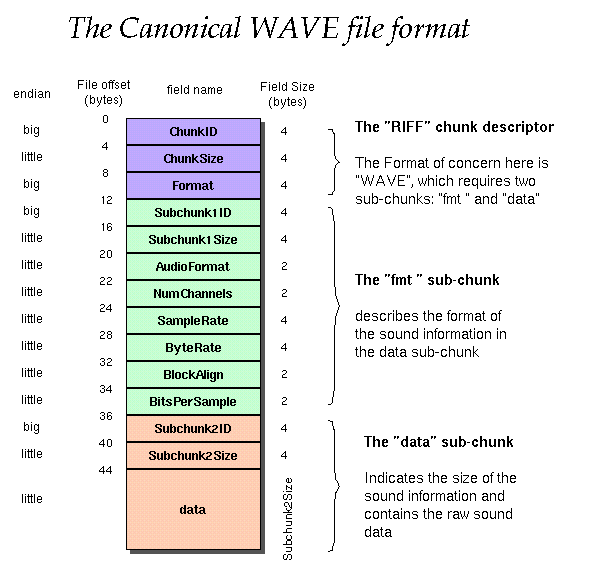
Omdat het in mijn eerste file niet echt wou werken, heb ik in de 2e file het al iets anders aangepakt daar heb ik gemekrt dat als je de header informatie in een struct steekt je het je eigen al veel gemakkelijker maakt maar tog woud die fourier niet werken dan.

### Lost all patients

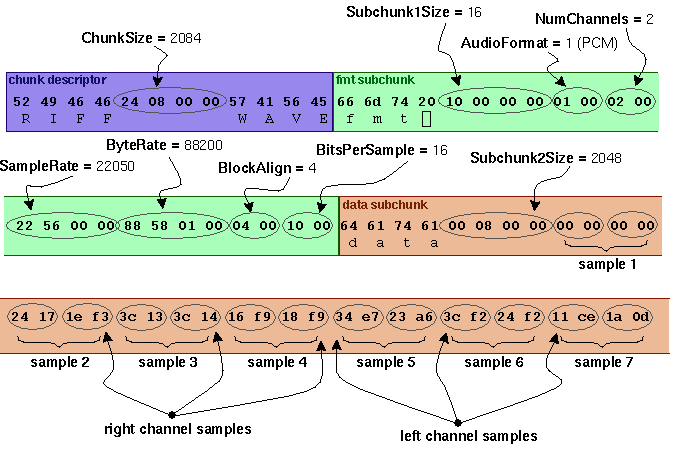
Toen na 2x alles nog nie wou werken, ben ik eens help gaan zoeken. En toen ben ik terecht gekomen bij Johan Donné die me enorm goed geholpen heeft, met me goed uit te leggen hoe die fourier ( met telkens ook verwijzingen naar hoe je dit moest doen met de fourier.cs) .

Maar dan moest ik nog alles omzetten in code, wat ik daar geleerd heb. Maar met al een goede basis over wat het ging, was dit al stukken gemakkelijker. Toen ik toen nog even terug keek naar men test files. Ben ik voor de 3e maal opnieuw begonnen.

#### WAV

Bij het inlezen van de wav files dacht ik het eerst hardcoded te doen volgens de afbeelding en zijn locaties, maar khad in mijn testfiles het inlezen van de wav al redelijk in orde, vond ik het niet zo’n slecht gedacht om toch iets meer checks te doen of alles wel goed loopt.

Daarvoor check ik of het fmt files zijn, controleer ik waar de data block zit zodanig, moesten er einge corrupties zijn met de header kan het programma toch nog de data blok vinden en zouden er geen directe problemen zijn (alleen als de data blok in de eerste 200 bytes zit) de code is ook zodanig geschreven dat je relatief gemakkelijk ondersteuning kunt geven aan andere wav formats.



Bij het inlezen zijn er een paar dingen echt van belang

1. Het aantal channels
2. De groote van 1 data stuckje (8 bit, 16 bit, 32 bit)
3. De sample rate

Waarom zijn die dingen nu zo belangrijk?

##### Channles

Als je niet weet of je data al dan niet in groepjes werken die op de tijdsbasis even ver ligen, zal dit al niet zo praktisch zijn voor als je deze zou willen uitplotten, dit zal waarschijnlijk wel werken met fourier, maar het is daarom nog niet juist

##### Groote 1 data stukje

aangezien fourier alleen maar complexe getallen, of een float array accepteerd, heb ik ge kozen om deze te verwerken als een float array, maar daarvoor moet je de data omzetten naar floats, maar deze zijn in bytes, en een byte = 8 bits, maar onze data kan (en was ook zo in de voorbeeld files) 16 bit, dus deze moesten omgezet worden.

Dus daarvoor had ik een bitconverter en Int16 waardes nodig, wat niet zo voor de handliggend was voor meeste mensen blijkbaar (mocht je dit bij een paar mensen tegen komen, is er wel een grote kans dat dit via mij komt, want ik had zo niet het gevoel dat iemand op het eerste gezich zelfs doorhad dat de data gekoppelt was met info dat je uit de header kreeg)

##### Sample rate

Deze info is voor een basis programma dat inleest en dan weer gewoon wegschrijft niet echt van belang, maar als je fourier wilt gebruiken komt dit echt wel van toepassing om te weten hoeveel Hz er per return value terecht komt

#### Fourier

Bij deze toe te passen moest ik natuurlijk eerst snappen wat deze deed, basicly wat fourier deed is de analyse maken van je wav file, dus je stak er een float array in. En dan werd de data van die array aangepast (wat ik opzich nog altijd nie mee akkoort ga, de data dat er na in zit was veel beter afgeweest als return value, of toch veel duidelijker)

Wat je terug krijg is weer een float array maar de waardes zijn niet meer als sinus representiatie, maar complexe getallen die het f-domein voorstellen, dus van gewooone getallen naar complexe, dit is een punt dat je niet mag vergeten, dat nu al je waarden in groepjes van 2 werken.

Ook bij de RFFT backward, moest je oppassen voor die factor waarmee alles werd versterkt. Deze had ik niet gevonden dankzei de hulp van Johan Donné.

#### Graphs

Voor de graphs te tekenen leek dit op het eerste gezicht gemakkelijker dan verwacht. Bij mijn eerste test gebruikte ik een sinus van 1khz om te zien of alles deftig verliep (kheb deze gedeelt op de FB groep van de 2e jaars) Mijn eerste resultaten waaren waren redelijk bizar, van ver kon je er wel een sinus in zien, maar van dicht liep er duidelijk iets mis.

Voor de plots wat te versnellen gebruik ik een gemiddelde van een x aantal waardes (gebazeerd op de lengte van de image en het aantal waardes dat ik heb) en door te debuggen zag ik dat hier ook veel negative waardes in zaten, dus toen viel het mij in dat een sinus op en neer gaat, maar voor mijn plots zou dat niet zo praktisch zijn, dus splitste ik de wav in 2: een positieve kant een negative kant, die even lang zijn. En op elke positie in die array krijg je de gemiddelde amplitude van het positieve of negatieve deel van de sinus.

Het Frequentie domein afprinten was dan weer gemakkelijker. Dit was gewoon de modulus nemen van de complexe getallen, hier het gemiddelde weer nemen gebaseert op breede en aantal waardes in de array.

#### Play

Dit was eigenlijk veel gemakkelijker dan verwacht. Wel met je eigen weggeschreven files was er een probleem van corrupted wav header, maar dat lag aan de manier waarop C# zijn default libraries dit doet. Hier heb ik een fix voor geschreven. De truc was gewoon om de stream te reseten en te beginnne vanaf 0

Dit stukje code heb ik ook gepost op de Facebook pagina van de 2e jaars dus dit zal waarschijnlijk ook wel bij andere mensen te vinden zijn. Maar dit was iets doms waar je lang kan op zoeken.

## REFLECTIE

Dit project was weer een fun ervaring. Nu snap ik de Fourier tenminste ook weer. Wat ik zeker nog ga gebruiken in toekomstige projecten, en ik moet zeggen de keuze van de projecten dat we moeten maken voor MMT beviel me echt hard, want muziek en programeren zijn 2 passies van mij, en hier kan ik die beiden combineren.

Alhoewel deze waarschijnlijk voor veel mensen te moeilijk was. Vond ik het eens een uitdaging :D   
Ik wil hierbij nogmaals Johan Donné bedanken voor de goede uitleg die hij me gegeven heeft, waardoor ik praktisch geen externe sites moest raadplegen

# Bibliografie

Kade, J. (2004, Augustus 6). *A simple C# Wave editor, part 1: Background and analysis*. Opgehaald van CodeProject: http://www.codeproject.com/Articles/7906/A-simple-C-Wave-editor-part-1-Background-and-analy

Wilson, S. (2003, Januari 20). *WAVE PCM soundfile format*. Opgehaald van Standford: https://ccrma.stanford.edu/courses/422/projects/WaveFormat/