Project - Fourier

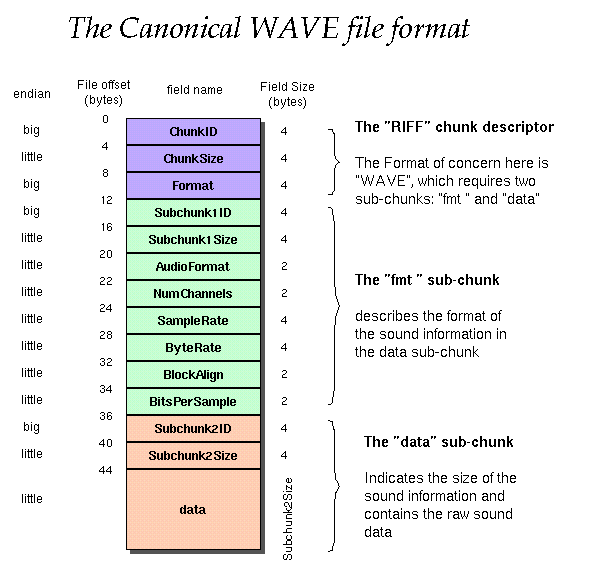
Wetenschappelijke toepassingen

Robbe De Geyndt – Odisee 2015-2016

Project 2 – Fourier

# Opdracht:

De bedoeling van het programma is om WAV-bestanden, aangemaakt met het programma van project 1, terug te kunnen omzetten naar de originele tekst. Dit gebeurt aan de hand van hetzelfde CSV‑bestand als gebruikt in project 1.



# Onderzoek:

In het diagram van het WAV formaat kunnen we zien dat de data begint vanaf byte 44, vanaf deze byte moeten we beginnen lezen om alle data (samples) te kunnen lezen.

De WAV-bestanden hebben een sample-rate van 44100 samples per seconde. Omdat de samples een grootte hebben van 2 bytes, en de data wordt ingelezen per byte, moeten deze samengevoegd worden in een short. Om de bits op de juiste plek krijgen doe ik een bitshift. De eerste byte die gelezen worden zijn de eerste 8 bit van het sample, de 2 byte zijn de laatste 8 bit.

De bijgeleverd Fourier-library berekend het verloop van de sinusgolf in de vorm van een reel deel en een complex deel. De formules om dit om te zetten staan in de werking van het programma.

Om het CSV-bestand in te lezen en de waarden op te splitsen op de tekens ‘,’ of ‘;’ heb ik de methode line.Split(‘char’) gebruikt.

# Gebruik van het programma:

De gebruiker kan bij het opstarten van het programma de knoppen ‘Open CSV’ en ‘Open WAV’ gebruiken om de nodige bestanden in te laden. Indien dit lukt worden de paden naar deze bestanden weergegeven in de UI en wordt de knop ‘Calculate’ aanklikbaar. Indien er iets niet lukt bij het inladen van deze bestanden zal deze knop niet aanklikbaar blijven en zal dit worden weergegeven in de status in de UI.

In de UI kan men ook instellen hoeveel seconden per letter men heeft gebruikt om het WAV-bestand te creëren.

Hierna kan de gebruiker op de knop ‘Calculate’ klikken, waarna de letters uit het WAV-bestand worden berekend en weergegeven in de textbox ‘Result:’.

# Werking van programma:

Bij het opstarten van het programma kan men een CSV-bestand kiezen, nadat men een bestand heeft geselecteerd wordt dit bestand ingeladen. Indien dit lukt, worden de waarden in een Dictionary opgeslagen.

Hierna krijgt de gebruiker de optie om een WAV-bestand te kiezen. Nadat het bestand gekozen is, wordt het ingeladen in een byte-array. De data-chunk wordt opgeslagen in een array, dit is vanaf positie 44, alle data die hiervoor komt, is informatie uit de header. De samples bestaan uit 16 bits, daarom worden de twee opeenvolgende bytes in de array samengevoegd in een float-array die dan uiteindelijk wordt doorgegeven aan de logica laag.

In de logica laag wordt op de basis van deze float-array dan de frequentie bepaald. Eerst wordt er bepaald hoeveel samples per letter er zijn, dit wordt berekend op basis van de sample-rate en het aantal seconden per letter (ingegeven via UI). Op basis van deze informatie wordt dan de float-array opgebroken in de verschillende letters.

Omdat de meegegeven Fourier-library enkel arrays kan gebruiken die een lengte heeft die een macht is van 2, worden de arrays ingekort tot de dichtstbijzijnde macht van 2.

De array kan hierna naar bewerkt worden door de Fourier-library, hierin worden dan de frequenties weergeven onder de vorm van complexe getallen. Hier wordt dan het vermogen van berekend met de volgende formule (Pythagoras): . Uit deze resultaten wordt dan de hoogste waarde gehaald, de index van deze waarde hebben we dan nodig om de frequentie te berekenen van de letter. Dit wordt gedaan op basis van de volgende formule: . Deze waarden van de frequenties worden dan doorgegeven.

Op basis van deze waarden worden dan de letters bepaald. De frequenties worden afgerond op 10 Hz, dit komt omdat de Fourier berekening niet altijd 100% accuraat is. Deze frequenties worden opgezocht in de Dictionary, die dan de overeenkomstige letter teruggeven. Frequenties die niet in de Dictionary voorkomen worden voorgesteld als een spatie.

# Besluit en reflectie:

Het moeilijkste aan het schrijven van de applicatie was de bewerkingen uitvoeren op de ingelezen float-array. Ik moest rekening houden dat de lengte van de arrays variabel kan zijn. Een voorbeeld hiervan is dat de array moet ingekort worden tot een lengte die een macht is van 2. Omdat een groter lengte zorgt voor een meer precisie, heb ik er voor gezorgd dat de lengte wordt ingekort tot de dichtstbijzijnde kleinere macht van 2. Dit heb ik opgelost door met een while-lus de machten van 2 af te lopen tot ik deze waarde tegen kwam.

Iets wat beter zou kunnen aan de applicatie is dat men niet zou moeten ingeven hoe lang iedere letter is. Om dit te kunnen doen zou men moeten kunnen detecteren wanneer er wordt overgegaan op een andere frequentie en daar dan de lengte van de letters uithalen. Hier zou men er dan ook rekening mee moeten houden dat er 2 maal dezelfde letter achter elkaar kan voorkomen.

# Bronnen:

Afbeelding WAVE-format:

<https://web.archive.org/web/20141213140451/https://ccrma.stanford.edu/courses/422/projects/WaveFormat/>

Hoe een WAVE file inlezen in een array:

<https://stackoverflow.com/questions/8754111/how-to-read-the-data-in-a-wav-file-to-an-array>

Formules voor transformatie complex getal naar frequentie:

<http://www.edaboard.com/thread57972.html>

Gebruik van de String.Split() methode:

<https://msdn.microsoft.com/nl-be/library/ms228388.aspx>