30 november 2015

Dwight Van der Velpen 2EO2

Odisee

Wetenschappelijke toepassingen 1

Workshop 1: Geluid

Inhoudsopgave

[1. Research: 2](#_Toc436676883)

[1.1. Onderzoek hoe WaveFile in elkaar zit. 2](#_Toc436676884)

[1.2. Hoe de sinus berekend word 3](#_Toc436676885)

[1.3. Csv bestanden 3](#_Toc436676886)

[2. Opbouw Project en schema 4](#_Toc436676887)

[2.1. GUI 4](#_Toc436676888)

[2.2. Logica 4](#_Toc436676889)

[2.2.1 BasisLogica 4](#_Toc436676890)

[2.2.2 Note 5](#_Toc436676891)

[2.2.3 Translate 5](#_Toc436676892)

[2.2.4 Wave 5](#_Toc436676893)

[2.3 Data 5](#_Toc436676894)

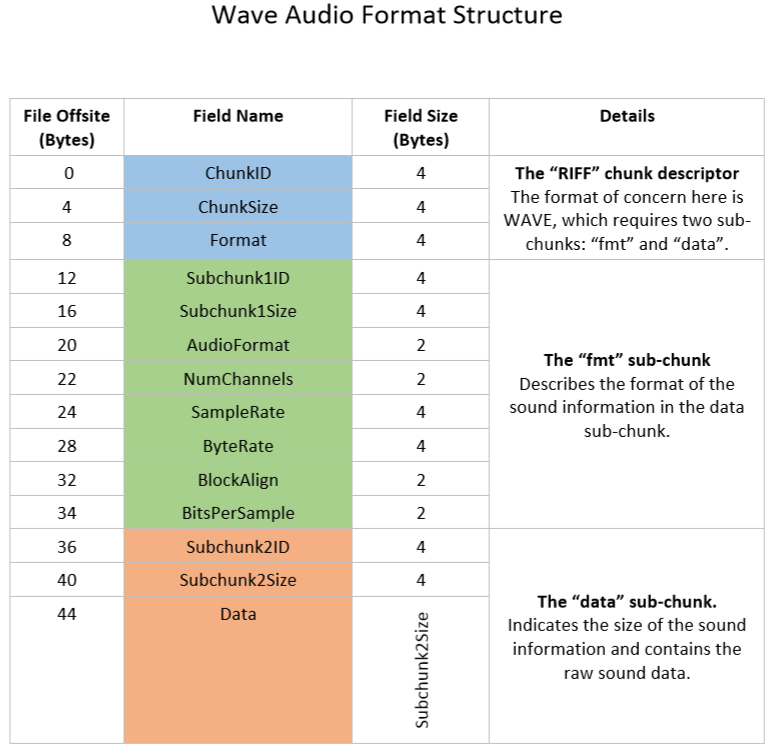
[2.4 globale data velden 5](#_Toc436676895)

[2.5 schema 5](#_Toc436676896)

[Verwijzingen 6](#_Toc436676897)

# Research:

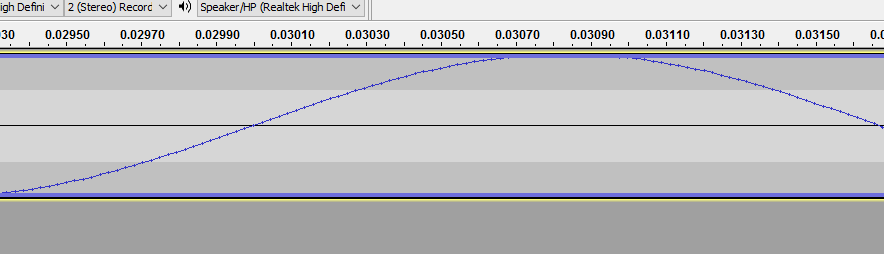
## Onderzoek hoe Wave File in elkaar zit.

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.



De meeste velden zijn in alle Wavefiles die ik aanmaak hetzelfde. Namelijk de ID’s, Format en alles wat in de format staat. Deze worden daarom in een de globale Data klasse als een soort template aangemaakt. Later word de data ingevuld (short Array), Data Size en de Total Size in de header.

## Hoe de sinus berekend word

Wegens we een tekstveld inlezen en moeten omzetten naar een sinus word er per letter een mini sinus gegenereerd van een bepaalde duur.

De sinussen worden opgeslagen in een data Array per sample en deze moeten dus berekend worden.   
Eerst ga je je periode bepalen van je sinus (2 \* pi \* frequentie) en dan wil je deze opdelen in het aantal samples per seconde zodat je weet wat je tijdstappen zijn. Daarna ga je van deze waarde de sinus nemen en daarna je stapwaarde een stap verder laten gaan. Dit zal (wegens de sinus functie) een mooie sinus genereren.

## 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 ‘,’.

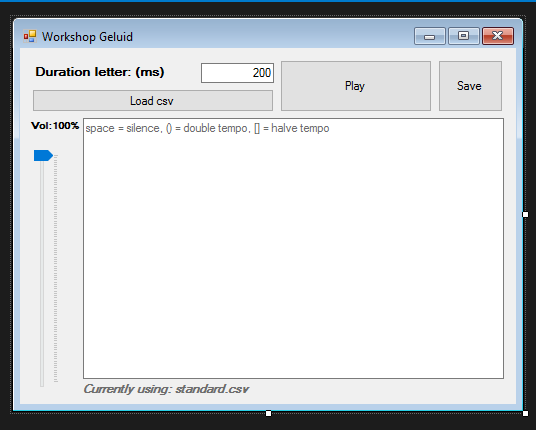
Ook heb ik er in toegevoegd dat je geen lijnen inleest die beginnen met //. Dit zorgt ervoor dat commentaar zou kunnen voorzien worden.

# Opbouw Project en schema

## GUI

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

* Tekstveld om naar audio om te zetten,
* Play knop en Save knop om genoemde audio op te slaan of af te spelen,
* CSV lader om snel van CSV bestand te wisselen,
* Duration tekstveld om de tijdsduur per noot in MS aan te passen,
* Volume Slider om je WAV file volume indien gewenst te beperken



## 2.2 Logica

### 2.2.1 Basis Logica

Deze klasse is de “entry” klasse van de Logica Layer, hij zal elke oproep uit de GUI Layer verwijzen naar andere Logica subklassen of doorstromen naar de backend (Data) indien nodig. Dit is handig zodat je in je GUI een beperkte verbinding hebt met de LogicaLayer. Dit voorkomt in mijn opinie spaghetti coderen en ook de complexiteit van de code, elke klasse zal een duidelijk nut hebben i.p.v. verschillende jobs.

### 2.2.2 Note

Deze klasse zal voor elke noot de juiste parameters bijhouden: frequentie als een float en Duration als een Time Span.

### 2.2.3 Translate

Deze klasse is zoals de naam impliceert de vertaler, hij zal een dictionary bijhouden om elke letter met de juiste frequentie te linken en daarna deze gebruiken in een methode die een string omzet naar lijst van noten.

### 2.2.4 Wave

Hierin worden de berekeningen gedaan. Hij krijgt een Noten Lijst en neemt uit de GlobaleDataVelden een instantie van elke chunk en berekent per noot de juiste sinus en stopt deze ook meteen in de Data-array.

## 2.3 Data

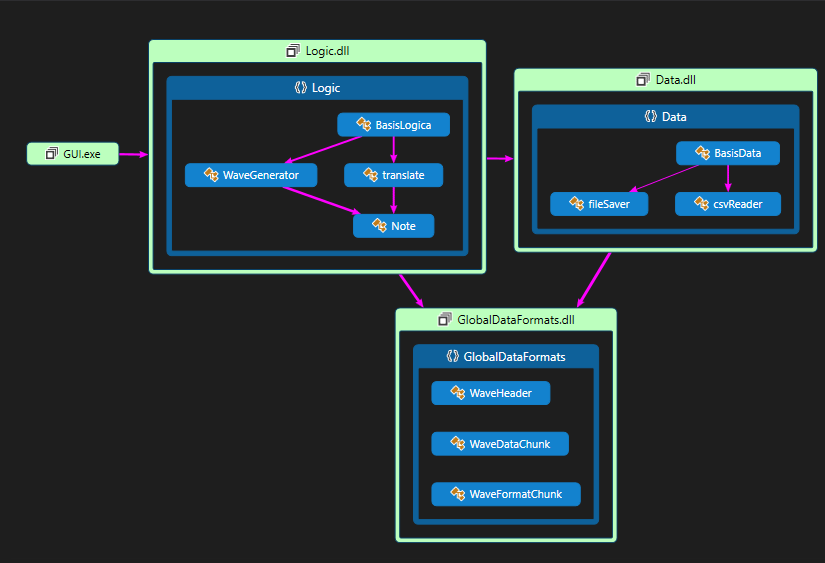
Hierin worden weer via een basisklasse verwezen naar 2 klassen die simpel samen te vatten zijn:

* CSV Reader: leest een CSV bestand in en geeft de Dictionary eruit terug.
* FileSaver: slaat een instantie van de GlobaleDataVelden op onder meegegeven pad.

## 2.4 globale data velden

Hierin staat de template van de Chunks om een wavefile te genereren. Deze kunnen als er een object van gemaakt wordt natuurlijk aangepast worden.

## 2.5 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.