Projet ISN :

Accordeur Numérique de Guitare

## 

## Membres du groupe:

## Emma NOUNKE

## Khoi NGUYEN

## Robert GACHECHEH

## 

## 

## 

## 

## 

## 

## Projet:

## Accordeur de guitare

## But:

Le but de ce projet est de créer un logiciel qui aura pour fonction d’accordeur (dit “standard”) de guitare: le logiciel suggérera de serrer ou de desserrer la clef (ou la laisser telle qu’elle est) après une analyse du fichier audio enregistré de la corde.

## Pour qui?

Les débutants qui apprennent seuls et/ou qui ne sauraient pas comment accorder une guitare acoustique.

## Avec quoi ?

# Micro

# Guitare acoustique

# Langage de programmation: Python 3.4

## Modules Python nécessaires:

* **PyAudio :** pour enregistrer et jouer des fichiers audio, utilisé conjointement avec le module Wave
* **SciPy :** pour l’utilisation des fonctions mathématiques telles que Fourrier dans l’analyse des fichiers
* **Numpy :** pour la manipulation des tableaux et des listes dans la création des sons et l’analyse des fichiers
* **Wave :** pour la manipulation des fichiers WAV
* **Math :** pour l’utilisation des fonctions mathématiques telles que sin () dans la création des sons
* **Struct :** pour la conversion des données décimales, sous forme de string, en données binaires dans la création des sons
* **Tkinter :** pour l’interface d’utilisateur graphique

## Principe du programme:

**Interface graphique**

**Menu principal**

Quitter

**Accorder la guitare**

Retour au menu précédent

**Première corde**

**Enregistrer le son de la corde:**

* Entrée: le son joué par l’utilisateur
* Sortie: un fichier WAV

**Raccourcir le fichier WAV**

* Entrée: un fichier WAV
* Sortie: un fichier WAV

**\***

**Déterminer la fréquence du son :**

* + Entrée: le fichier WAV d’une corde
  + Sortie: la valeur de la fondamentale (int)

**\*\***

**Comparaison de la valeur de la fréquence obtenue avec la fréquence de la corde (intervalle d’incertitude = +/- 5Hz):**

* Entrée: la valeur de la fréquence fondamentale obtenue (int)
* Sortie: un message indiquant le résultat de la comparaison (str)

**Créer et/ou Jouer la note correspondant à la corde**

**Suggestion pour accorder:**

* Entrée: le message indiquant le résultat de la comparaison (str)
* Sortie: un message indiquant comment accorder la corde (str)

**Deuxième corde…**

**Jusqu’à la sixième corde**

**\*** Paramètres du fichier WAV:

* Codage en 16 bit
* Fréquence d'échantillonnage = 44100Hz
* Paramètres couramment utilisés pour un disque CD audio
* Mono
* Pour réduire la taille du fichier WAV, puisque Stéréo nécessite plus de données (plusieurs signaux contre 1 seul signal pour Mono)
* Durée de 4s (durée qui nous semblait convenable pour une analyse)

**\*\*** Analyse du fichier :

On extrait et stocke les données du fichier WAV sous la forme d’un tableau à 2 dimensions, les valeurs étant en décimales, avec la fonction *wavfile.read()* du module SciPy.

Puis on fait une analyse de Fourrier de ce tableau avec la fonction *fft()* de SciPy et on utilise la fonction *numpy.fft.fftfreq()* du module Numpy pour avoir une liste *freqs* des fréquences des harmoniques (fonctions périodiques) composant le son du fichier WAV. On précise que la fonction *numpy.fft.fftfreq()* prend comme arguments la taille du tableau de données et l’inverse de la fréquence d’échantillonnage du signal numérique.

Ensuite on cherche l’indice *idx* de la plus grande valeur dans le tableau de Fourrier, les éléments étant mis en valeur absolue. Pour cela, on utilise les fonctions *numpy.argmax()* et *numpy.abs()* de Numpy.

Enfin, on peut alors trouver la fréquence du son du fichier WAV, qui est la valeur de la liste *freqs* d’indice *idx.*

def Frequence(filename):

fname = str(filename) + ".wav"

rate, data = wavfile.read(fname) # On obtient ici un tableau à 2 dimensions

a = fft(data) # Analyse de Fourrier FFT

n = data.size

timestep = 1/rate

freqs = numpy.fft.fftfreq(n, d=timestep) # On obtient ici une liste des fréquences (harmoniques)

idx = numpy.argmax(numpy.abs(a))

freq = freqs[idx] # La fréquence du son qu'on cherche

return int(freq)

## Détails supplémentaires :

Un accordeur de guitare ne peut pas entendre le ton, mais il peut détecter la fréquence. La fréquence d'un son est le nombre de maximums d’une fonction périodique qui dépassent un point fixe dans un temps donné. La fréquence est mesurée en Hertz (Hz).

Les cordes se nomment : E, A, D, G, B et E, ordonnées de la plus basse fréquence à la plus haute. (Corde la plus épaisse (low E) à la plus fine (high E))



1

6

|  |  |  |
| --- | --- | --- |
| Corde | Fréquence | Nom de la corde |
| 1 (E) | 330 Hz | E4 |
| 2 (B) | 247 Hz | B3 |
| 3 (G) | 196 Hz | G3 |
| 4 (D) | 147 Hz | D3 |
| 5 (A) | 110 Hz | A2 |
| 6 (E) | 82 Hz | E2 |

## 

*Guitare acoustique*

*(*[*http://images.all-free-download.com/images/graphiclarge/acoustic\_guitar\_184808.jpg*](http://images.all-free-download.com/images/graphiclarge/acoustic_guitar_184808.jpg)*)*

## Répartition des tâches :

*Diagramme de Gantt en utilisant Smartsheet*

*Remarque : Les barres non remplies indiquent les tâches pas complètes à la date limite*

Rouge : Robert Gachecheh

* Interface graphique

Vert : Emma Nounke

* Enregistrer et jouer son ; Puis Interface graphique

Bleu : Khoi Nguyen

* Enregistrer et jouer son ; Analyse spectrale FFT + Planning :

Mon rôle est d’abord de m’occuper du planning du groupe. Pour cela, j’ai utilisé le site <https://freedcamp.com/> . Le site permet de mettre en place des tâches à faire et des dates limites qu’il faut respecter, ainsi que d’avoir un milieu pour partager les documents entre les membres du groupe. Néanmoins, le site ne permet pas d’avoir un organigramme du type diagramme de Gantt afin de mieux visualiser l’organisation du groupe. Ainsi, j’ai utilisé le site [https://www.smartsheet.com/](https://www.smartsheet.com/%20) pour créer ce graphique.

Au niveau du codage, je me suis occupé des algorithmes hors interface graphique, c’est-à-dire les algorithmes pour enregistrer, jouer et faire une analyse FFT d’un fichier audio WAV.

Les problèmes rencontrés / Les solutions envisagées :

* Le premier grand problème qu’on a rencontré était de trouver un éditeur de texte qui convient à nos besoins. Comme on a étudié Python en classe en utilisant EduPython (une version de PyScripter), on a voulu l’utiliser afin d’être plus efficace et ne pas avoir besoin de s’adapter à un nouvel éditeur de texte. Mais EduPython ne peut pas importer certaines modules nécessaires à notre projet, telle que PyAudio, et donc ne peut pas être utilisé.

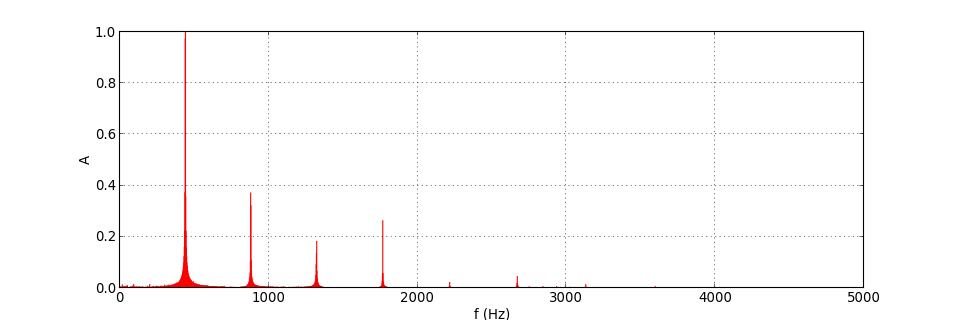
Ainsi, on s’est tourné vers Microsoft Visual Studio, un autre environnement de développement qu’on a gardé pour le reste du projet. Celui-ci est gratuit et est assez simple à utiliser, malgré de nouvelles options absentes dans EduPython. Toutefois, on utilise toujours EduPython dans le cadre de la classe pour le codage de l’interface graphique (seulement Tkinter à importer), puisque Visual Studio n’est qu’accessible avec notre ordinateur personnel.

* Au début du projet, un autre problème était de trouver un moyen pour trouver la fréquence d’un fichier audio, c’est-à-dire analyser le fichier. Comme on n’a pas encore manipulé le son en informatique auparavant, on a passé la grande majorité de notre temps pour le projet à faire des recherches.

Lorsqu’on a vu la possibilité de faire une analyse de Fourrier avec Python et que les professeurs, qui nous encadrent, nous ont suggéré de poursuivre cette voie, on a décidé de baser notre programme sur une analyse spectrale FFT.

La fréquence qu’on cherche

Amplitude



*Exemple de spectre de fréquence (FFT) d’un son synthétisé*

*(*[*http://www.f-legrand.fr/scidoc/docmml/numerique/tfd/spectreson3/spectreson3.html*](http://www.f-legrand.fr/scidoc/docmml/numerique/tfd/spectreson3/spectreson3.html)*)*

Bilan :

* L’état final du projet ISN:

Par rapport à notre vision au départ pour le projet, on a pu se rapprocher beaucoup avec notre résultat final. On a pu véritablement coder un programme qui serait capable de trouver la fréquence de la corde enregistrée afin de suggérer à l’utilisateur comment accorder sa guitare. Une interface graphique fonctionnante a été aussi réalisée, mais n’a pas la forme qu’on souhaiterait avoir.

Avec plus de temps, on souhaite apporter une première amélioration au programme au niveau de l’interface d’utilisateur, en la rendant plus ergonomique et plus attirante à voir. On souhaite aussi pouvoir transformer ce programme en un logiciel exécutable depuis le bureau de l’ordinateur, sans passant par un éditeur de texte tel que Visual Studio afin de le rendre plus accessible à des utilisateurs potentiels.

* Mon ressenti par rapport au projet:

Ce projet a été très utile pour moi, puisqu’il m’a permis d’appliquer mes connaissances en programmation afin d’aborder un problème qui m’intéresse, dans le cadre d’un travail en groupe. En tant que la personne qui doit s’occuper du planning du groupe, je sens que mes compétences à travailler avec plusieurs personnes et à gérer le temps dans le but de résoudre un problème en respectant les dates limites ont beaucoup amélioré. Ceci était aussi une chance pour moi de rentrer en détail dans la manipulation du son en informatique, un sujet qui n’a été pas réellement abordé en cours d’ISN et qui m’a toujours intrigué.

Annexes :

**Sitographie/ Bibliographie:**

* <https://wiki.python.org/moin/PythonInMusic>
* <http://people.csail.mit.edu/hubert/pyaudio/>
* <https://docs.python.org/3/library/wave.html>
* <http://stackoverflow.com/questions/3694918/how-to-extract-frequency-associated-with-fft-values-in-python>
* <http://stackoverflow.com/questions/23377665/python-scipy-fft-wav-files>
* <http://stackoverflow.com/questions/20830791/using-python-for-ffts-on-audio-files-sampling-rate>
* <http://ubuntuforums.org/showthread.php?t=1882580>
* <http://docs.scipy.org/doc/scipy/reference/>
* <http://docs.scipy.org/doc/numpy/reference/index.html>
* <https://docs.python.org/3.4/library/struct.html>
* <https://fr.wikipedia.org/wiki/Guitare>
* Cours de Physique Terminale S

**Codes source :**

import pyaudio

import wave

import numpy

import math

import struct

from scipy.io import wavfile

from scipy.fftpack import fft

from tkinter import \*

# Paramètres nécéssaires pour manipulation audio

CHUNK = 1024 # nombre de points d'échantillonage

FORMAT = pyaudio.paInt16 # codage en 16 bit

CHANNELS = 1 # mono

RATE = 44100 # fréquence d'échantillonage

RECORD\_SECONDS = 4 # durée d'un fichier

"""---------------------------------------------------------------------"""

# Fonction pour enregistrer audio

def Enregistrer(filename):

fname = str(filename) + ".wav"

# intialiser PyAudio

p = pyaudio.PyAudio()

# ouvrir le flux audio (les paramètres)

stream = p.open(format=FORMAT,

channels=CHANNELS,

rate=int(RATE),

input=True,

frames\_per\_buffer=CHUNK)

# enregistrement

print("\* Enregistrement")

frames = []

for i in range(0, int(RATE \* RECORD\_SECONDS), int(CHUNK)):

data = stream.read(CHUNK) # On lit les données du flux audio

frames.append(data) # On ajoute les donnés lues dans une liste

print("\* Enregistrement fini")

# fermer le flux audio et PyAudio

stream.stop\_stream()

stream.close()

p.terminate()

# 'écrire' les données enregistrées sur un fichier WAV

wf = wave.open(fname, 'wb')

wf.setnchannels(CHANNELS)

wf.setsampwidth(p.get\_sample\_size(FORMAT))

wf.setframerate(RATE)

wf.writeframes(b''.join(frames))

wf.close()

# Fonction pour jouer audio

def Jouer(filename):

# ouvrir un fichier wav

wf = wave.open(str(filename) + ".wav")

# intialiser PyAudio

p = pyaudio.PyAudio()

# ouvrir le flux audio (les paramètres)

stream = p.open(format=FORMAT,

channels=CHANNELS,

rate=int(RATE),

output=True)

# jouer le flux audio

print("\* Lecture du son")

data = wf.readframes(CHUNK) # On lit les données du fichier WAV

while len(data) > 0:

stream.write(data) # On 'écrit' ces données dans le flux audio

data = wf.readframes(CHUNK)

print("\* Lecture finie")

# fermer le flux audio et PyAudio

stream.stop\_stream()

stream.close()

p.terminate()

# Fonction pour couper 1s au début et à la fin d'un fichier audio

def Raccourcir(filename):

fname = str(filename) + ".wav"

wf = wave.open(fname, "r")

# Les paramètres du nouveau fichier

nchannels = CHANNELS

sampwidth = wf.getsampwidth()

framerate = wf.getframerate()

nframes = framerate \* (3 - 1) # La taille du nouveau fichier, en coupant 1s au début et à la fin

comptype = "NONE"

compname = "not compressed"

start\_index = 1 \* framerate # On définit le début du fichier WAV

newfile = wave.open("Rac" + fname, "w")

newfile.setparams((nchannels, sampwidth, framerate, nframes,

comptype, compname))

wf.rewind() # On repositionne le curseur à la position 0

anchor = wf.tell() # La position du curseur dans le fichier WAV

wf.setpos(anchor + start\_index) # On positionne le curseur

# 'écrire' les données dans le fichier raccourci

newfile.writeframes(wf.readframes(nframes))

# Fonction pour créer un son de reference

def SonRef(f,filename):

freq = int(f) # La fréquence du son

data\_size = RATE \* RECORD\_SECONDS

amplitude = 64000.0

fname = str(filename) + ".wav"

# On crée un tableau de valeur de la fonction périodique du son

sine\_list\_x = []

for x in range(data\_size):

sine\_list\_x.append(math.sin(2\*math.pi\*freq\*(x/RATE)))

wf = wave.open(fname, "w")

# Les paramètres du fichier WAV

nchannels = CHANNELS

sampwidth = 2

framerate = int(RATE)

nframes = data\_size

comptype = "NONE"

compname = "not compressed"

wf.setparams((nchannels, sampwidth, framerate, nframes,

comptype, compname))

# 'écrire' les données en bit à partir du tableau de valeur

for s in sine\_list\_x:

wf.writeframes(struct.pack('h', int(s\*amplitude/2)))

wf.close()

# Fonction pour analyse audio/donner la fréquence du son d'un fichier wav

def Frequence(filename):

fname = str(filename) + ".wav"

rate, data = wavfile.read(fname) # On obtient ici un tableau à 2 dimensions

a = fft(data) # Analyse de Fourrier FFT

n = data.size

timestep = 1/rate

freqs = numpy.fft.fftfreq(n, d=timestep) # On obtient ici une liste des fréquences (harmoniques)

idx = numpy.argmax(numpy.abs(a))

freq = freqs[idx] # La fréquence du son qu'on cherche

return int(freq)

# Fonction pour comparer deux fréquences

def Comparer(f,fref):

if f < int(fref-5):

return "inf"

elif f > int(fref+5):

return "sup"

elif f <= int(fref+5) and f >= int(fref-5):

return "egal"

# Fonction pour suggestion

def Suggestion(resultat):

if resultat == "inf":

print("Déserrez la clef")

elif resultat == "sup":

print("Serrez la clef")

elif resultat == "egal":

print("Votre corde est bien accordée!")

"""---------------------------------------------------------------------"""

# Créer les fichiers sons de référence

"""SonRef(82,"6\_E2")

SonRef(110,"5\_A2")

SonRef(147,"4\_D3")

SonRef(196,"3\_G3")

SonRef(247,"2\_B3")

SonRef(330,"1\_E4")"""

Freq = ["1\_E4", "2\_B3", "3\_G3", "4\_D3", "5\_A2", "6\_E2"]

choix = str(input("Quelle demo? Accordage (1) ou Enregistrement puis Lecture (2)?"))

print("")

if choix == str(1): # Un exemple d'accordage

print("Quelle corde?")

n = input()

#Enregistrer("Corde" + str(n))

print("")

print("Corde n" + str(n))

fref = Frequence(str(Freq[int(n)-1]))

Raccourcir("Corde" + str(n))

f = Frequence("RacCorde" + str(n))

print(str(fref) + " Hz")

print(str(f) + " Hz")

resultat = Comparer(f, fref)

Suggestion(resultat)

elif choix == str(2): # Un exemple d'enregistrement puis lecture

Enregistrer("CordeTest")

print("")

Jouer("CordeTest")

else:

print("Erreur")