Travaux pratiques d'informatique musicale, **Préparations aux TP**

Licence 2, de physique et musicologie (version : 08/02/2024)

Frédéric Faure
Université Grenoble Alpes, France
frederic.faure@univ-grenoble-alpes.fr

Résumé

Dans ces **travaux pratiques** on utilise la librairie JUCE en C++ pour fabriquer des applications et plugins audio et midi. Dans ce TP, on prépare les TP suivants.

Table des matières

Remarque 0.1. Dans la version électronique de ce document (pdf), vous accédez aux pages de cours et autres documents en **cliquant sur les liens de couleur qui entourent le texte**.

Le travail à faire en TP est indiqué sous la forme « Exercice (TP) ». En préalable à la séance, on conseille de lire attentivement ce document, de faire les « Exercices (TD) ». Ne pas hésiter à demander de l'aide technique à ChatGpt par exemple.

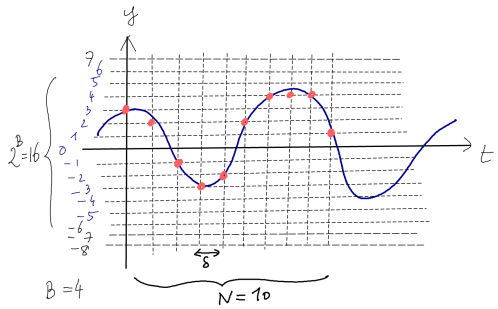
1 Exercices de TD sur les signaux Audio et MIDI

Dans cette première partie on résume des notion importantes à connaître pour la suite du TP.

1.1 Signaux audio en informatique

— Un **signal audio** est un signal sonore (une fonction $t \in \mathbb{R} \to s(t) \in \mathbb{R}$) qui a été transformée en une suite de nombre à l'aide d'une carte son (on parle de conversion analogique numérique).
— Pour cela on sélectionne des valeurs discretes du temps t espacés par un petit intervalle $\delta = \frac{1}{N}$, avec par exemple $N = 44100$Hz intervalles par seconde, appelée fréquence d'échantillonage ou « rate frequency ».

— A chacun de ces instants t, l'amplitude du signal $s(t)$ est approximée par un entier dans l'intervalle $(-2^{B-1}, 2^{B-1}-1)$ où l'entier B est appelée « bit sampling », par exemple $B = 4$ sur la figure ci-dessous, mais habituellement on prend $B = 32$ pour avoir une bonne précision. Cette approximation est parfois appelée quantification du signal)



1.2 Signaux Midi en informatique

— Un **signal MIDI** est une suite de nombres qui représentent des instructions entre appareils musicaux (ex : claviers, surfaces de controle, expandeur, ordinateurs etc) en musique électronique.

Exemple 1.1.

— l'instruction de « jouer la note C de l'octave 5 avec un volume maximal » est représentée par la suite de nombres $\{144, 60, 127\}$ (car 144 signifie 'note on', 60 signifie 'C5' c'est à dire 'C' de l'octave 5, et 127 est le volume).

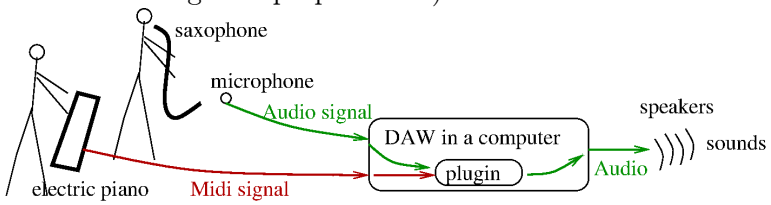
— l'instruction « éteindre la note C de l'octave 5 » est représentée par la suite de nombres $\{128, 60, 127\}$ (car 128 signifie 'note off', 60 signifie 'C' de l'octave 5 et 127 signifie coupure nette du son).

— La liste des instructions MIDI est définie par la MIDI association et un résumé des messages importants que l'on étudiera plus tard.

1.3 Logiciel DAW

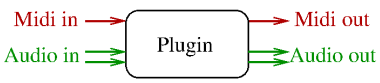
— Un logiciel **DAW** (signifie Digital Audio Workstation) est un logiciel qui permet **d'enregistrer des signaux audio et midi sur des pistes** (tracks) et de **rejouer ces pistes**, puis envoyer le son sur des enceintes.

— On proposera d'utiliser Ardour, qui est un logiciel DAW libre et performant. Dans le monde professionnel les ingénieurs du son utilisent fréquemment Protools sur windows PC ou Logic-Pro sur MacOs (mais ce sont des logiciels propriétaires).



1.4 Logiciel Plugin (ou Greffon ou Module)

— Un **audio plugin** est un logiciel qui peut être utilisé dans un DAW pour transformer des signaux audio et/ou MIDI.



— Voici des exemples de plugins où on précise la nature des signaux Entrée→Sortie :
  — Audio → Audio, sont des plugins appelés « AudioFX » (effets audio). Par exemple une « réverbération », un filtre fréquenciel.
  — Midi→Audio, sont des plugins appelés « instruments ». Par exemple un synthétiseur.
  — Moins courants sont :
    — Midi→Midi, sont des plugins appelés « MidiFX » (Effet MIDI).
    — Audio,Midi→Audio, appelés « MusicEffect ».

— Audio→Midi, par exemple pour détecter des notes d'instruments acoustique et les envoyer vers un synthétiseur (voir par exemple Video on audio->midi).

1.4.1 Latence et buffer audio

La perception humaine de l'audio accepte des **retards, appelé latence, inférieur à** $L = 10\text{ms} = 0.01\text{s}$. **mais pas supérieurs**. Cela est du au temps de réponse des neurones dans notre cerveau.

Ainsi un plugin Audio comme celui que nous réalisons nécessite de gérer des évènements MIDI et Audio et faire des calculs sur une durée assez courte, inférieure à cette latence de 10ms. On appelle cela du **'calcul en temps réel'.** Pour cela, le logiciel DAW envoit (et recoit en retour) au programme interne plugin les signaux MIDI et AUDIO de façon périodique par petits paquets appelés buffer.

Par exemple, nous avons vu en section 1.1 que la fréquence d'échantillonage est $N = 44100\text{Hz}$ échantillons par seconde. Donc dans la durée de latence acceptable $L = 10^{-2}s$. il y a

$$n = N \times L = 441 \text{ échantillons.}$$

Ainsi on demandera au DAW d'envoyer et recevoir des buffers audio de taille $n = 441$ à chaque période $L = 10^{-2}s$. ou quelque chose de comparable.

1.5 Représentation des nombres en informatique, binaire, hexadécimal, complément à deux.

Ce sont des notions simples mais importantes sur la représentation des nombres, utilisés en informatique en général et plus particulièrement utile en informatique musicale. Voir l'annexe B4 de ce **cours d'acoustique musicale**. Pour rappel, voici une liste des premiers entiers codés sur 4 bits, très utile, qu'il faut savoir retrouver :

base 16	0	1	2	3	4	5	6	7	8	9	*a*	*b*	*c*	*d*	*e*	*f*
base 2	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
base 10	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Exercice 1.2. Ecrire les messages midi de l'exemple 1.1 en hexadécimal et en binaire.

Solution 1.3.

Signification	Note on C5, vel=127	Note off C5, vel=127
décimal	$144, 60, 127$	$128, 60, 127$
hexadécimal	$90, 3c, 7f$	$80, 3c, 7f$
binaire	1001 0000 0011 1100 0111 1111	1000 0000 0011 1100 0111 1111

2 Installation d'un compilateur pour C++

On aura besoin d'un environnement de programmation pour la langage C++. Selon votre ordinateur, suivre les instructions suivante.

2.1 Pour Windows

— Installer Visual-Studio option Community, avec le « Workloads Desktop develpment with C++ », qui est un environnement de programmation pour coder en C++.

2.2 Pour MacOs

— Installer XCode.

2.3 Pour Linux

3 Installation de JUCE

JUCE est un ensemble de bibliothèques C++ qui permettent de concevoir des plugins audio pour Linux, MacOs, IOS, Windows, Android, etc.

Pour approfondir on peut suivre des tutorials de JUCE.

3.1 JUCE avec Projucer

— Sur le site de juce.com/download/, cliquer sur « JUCE et Projucer » (pour windows/MacOs/-linux selon votre ordinateur), et extraire le fichier téléchargé. Cela crée le répertoire JUCE

— On peut déjà essayer JUCE/**DemoRunner** (cliquer sur « More Info/Run anyway » ou « informations complémentaires/Executer quand même »), onglet Browse Demos, et par exemple **Audio/PluckedStringsDemo** et cliquer sur les cordes de la harpe pour entendre le son. Prendre le temps d'essayer différents programmes de démonstration.

3.1.1 Création d'une application JUCE à partir d'un exemple existant

Préparation du logiciel Projucer

Remarque 3.1. Projucer est un logiciel proposé par JUCE qui permet de définir son projet de programmation.

— Lancer JUCE/**Projucer** (cliquer sur informations complémentaires/Executer quand même)
  — La première fois, Cliquer dans Menu/File/Global_Paths, indiquer le chemin des répertoires de JUCE et JUCE/modules.

Avec Windows :

— Dans Projucer
  1. choisir Menu/File/Open_example/Audio/PluckedStringsDemo
     (a) Menu/File/Save_Project_and_Open_in_IDE : sauvegarder ce projet dans un répertoire existant. Puis cela ouvre le projet dans le logiciel Visual Studio.
  2. Dans **Visual Studio** (qui est déjà lancé)
     (a) Pour **compiler**, faire Menu/Build/Build_Solution et quand c'est terminé, faire Menu/-Debug/Start_without_debugging pour **exécuter le programme**.
     (b) Remarque : le programme executable créé est dans le chemin PluckedStringsDemo/Builds/VisualStudio/x64/Debug/App/PluckedStringsDemo

Avec Linux :

— Dans Projucer
  1. choisir Menu/File/Open_example/Audio/PluckedStringsDemo
     (a) Menu/File/Save_Project
  2. Ouvrir un terminal dans le répertoire PluckedStringsDemo/Builds/LinuxMakefile
     (a) Pour compiler, écrire : **make -k** -> **BUG ? ?**
     (b) Pour exécuter, écrire :

Avec MacOs :

— Dans Projucer
  1. choisir Menu/File/Open_example/Audio/PluckedStringsDemo
     (a) Menu/File/Save_Project_and_Open_in_IDE : sauvegarder ce projet dans un répertoire existant. Puis cela ouvre le projet dans le logiciel XCode
  2. Dans **XCode** (qui est déjà lancé)
     (a) Pour **compiler**, faire Menu/Product/Build et quand c'est terminé, faire Menu/Product/-run pour **exécuter le programme**.
     (b) Remarque : le programme executable créé est dans le chemin PluckedStringsDemo/Builds/MacOsX/build/Debug/PluckedStringsDemo

3.1.2 Autre exemple avec un plugin synthétiseur

Suivre les même étapes que ci-dessus, voici les différences essentielles.

1. Dans **Projucer**
   (a) choisir Menu/File/Open example/Plugins/MultiOutSynthPluginDemo
   (b) Menu/File/Save_Project_and_Open_in_IDE
2. Dans le logiciel C++, **compiler et exécuter** le programme standalone (qui est un programme indépendant de tout DAW. Cela sert à tester le projet).
3. Pour **jouer avec le synthétiseur**, brancher un clavier MIDI externe, et dans le menu options du plugin, activé le clavier en entrée midi.
4. Pour utiliser le plugin dans Ardour (ou un autre DAW comme Garage Band etc)
   (a) Sous windows, indiquer le répertoire où est situé le plugin VST3. Menu/Fenetres/Gestionnaire_de_greffon puis 'Chemin_VST3' et 'Add' et indiquer le chemin.

4 Langage C++ et JUCE

Les projets nécessitent des concepts assez évolués d'informatique comme la programmation temps réel, multi thread, multi plateformes et programmation objet en C++. Cependant le but est de réaliser un projet intéressant d'un point de vue physique, mathématique et musical en s'initiant à la programmation. Pour cette raison, **on ne demande pas de maîtriser à priori les concepts d'informatiques, on présentera leur logique au fur et à mesure.** Il faut savoir que ces concepts ne sont pas compliqués d'un point de vue logique mais sont souvent compliqués en apparence car leur syntaxe suit des conventions précises (l'informatique actuelle ne supporte pas la moindre erreur de syntaxe sinon le programme bug).

— Pour une initiation à C++ on propose le tutorial ou celui ci : tutorial C++ orienté physique numérique
— Pour approfondir on donnera des liens à la documentation des classes C++ de JUCE utilisées.