

# Editeur musical avec lecture audio

*et génération analogique d'ondes sinusoidales et carrées*

Victor VEILLERETTE

Janvier 2017

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Sujet initial . . . . .	2
<b>2</b>	<b>Développement</b>	<b>3</b>
2.1	Sources . . . . .	3
2.2	Plateforme de suivi Git . . . . .	4
<b>3</b>	<b>Implémentations</b>	<b>5</b>
3.1	Partition . . . . .	5
3.1.1	Durée des notes . . . . .	5
3.1.2	Hauteur des notes . . . . .	6
3.1.3	Structure d'une note . . . . .	6
3.1.4	Représentation d'une mesure . . . . .	6
3.1.5	Portée et Partition . . . . .	7
3.2	Audio . . . . .	8
3.2.1	Principe . . . . .	8
3.2.2	Fréquences de base . . . . .	8
3.2.3	Remplissage du buffer et Mixage . . . . .	8
3.2.4	Lecture . . . . .	9
<b>4</b>	<b>Eléments Techniques</b>	<b>10</b>
4.1	Compilation et Lancement . . . . .	10
4.2	Format MED . . . . .	10
4.3	Format ABC . . . . .	11
4.4	Bugs connus . . . . .	12
<b>5</b>	<b>Manuel d'utilisation</b>	<b>13</b>
5.1	Interface générale . . . . .	13
5.1.1	Le mode Ajout . . . . .	13
5.1.2	Le mode Edition . . . . .	14
5.2	Explorateur de fichier . . . . .	15
5.3	Perspectives . . . . .	15

# Chapitre 1

## Introduction

### 1.1 Sujet initial

Le sujet de base était la réalisation d'un éditeur de partition générant ses propres sons.

Le projet était à réaliser durant un semestre, c'est-à-dire d'Octobre 2016 à Janvier 2017. Voici l'intitulé exact du sujet :

Le but du projet est de réaliser un éditeur classique de partition musicale similaire à MuseScore, proposant les fonctionnalités basiques d'édition d'une partition musicale.

On permettra à l'utilisateur d'éditer une partition musicale en le forçant à respecter les contraintes inhérentes (entre autre : l'empêcher de mettre trop de notes par rapport à la durée d'une mesure), en se focalisant dans un premier temps sur une seule voix. L'outil devra être capable de jouer la partition ainsi rédigée (sons générés par ordinateur - ondes sinusoïdales par exemple).

L'utilisateur devra en outre pouvoir "personnaliser" les fréquences de chaque note. Tout en gardant 12 divisions par octave, on pourrait par exemple, déterminer chaque "quantité de fréquences" à allouer à chaque division.

Les modalités du projet étaient les suivantes :

- Langage de programmation : C
- Utilisation de la librairie SDL
- Aucune utilisation de code tiers

# Chapitre 2

## Développement

### 2.1 Sources

Le projet est une application entièrement écrite en C ANSI (1989). Il est à noter que le programme n'utilise aucune image externe et génère de lui-même tous les dessins affichés à l'écran. Pour cela le programme utilise les bibliothèques suivantes :

- **SDL-1.2** : La bibliothèque *Simple Direct Media Library* permet l'ouverture d'une fenêtre, le remplissage de rectangles, la gestion des événements.
- **SDL\_gfx 2.0.25** : Cette extension de la SDL permet l'utilisation de primitives de dessin et de zoom plus évoluées.
- **SDL\_ttf 2.0** : Cette extension de la SDL permet l'affichage de texte à l'écran.

Le dossier principal du projet est organisé de la façon suivante :

- **doxygen/** : Documentation technique générée par Doxygen. Elle regroupe la documentation complète de toutes les structures et de toutes les fonctions du projet.
- **media/** : Fichiers de polices nécessaires à l'exécution de l'application.
- **include/** : Fichiers headers du projet .h ; chaque fichier est rattaché à un fichier source, et donc à un module.
- **doc/** : Quelques fichiers exemples .abc ou .med à charger dans l'application.
- **src/** : Fichiers sources du projet .c ; chaque fichier correspond à un module.
- **Makefile** : Fichier Makefile permettant la compilation avec la commande *make*.

## 2.2 Plateforme de suivi Git

Malgré le fait que le développement du programme ne soit effectué que par une personne, j'ai voulu tester l'utilisation du logiciel de gestion de version **git** avec la plateforme en ligne **Github.com**<sup>1</sup>.

Le projet est constitué d'environ 110 commits pour plus de 10000 lignes :

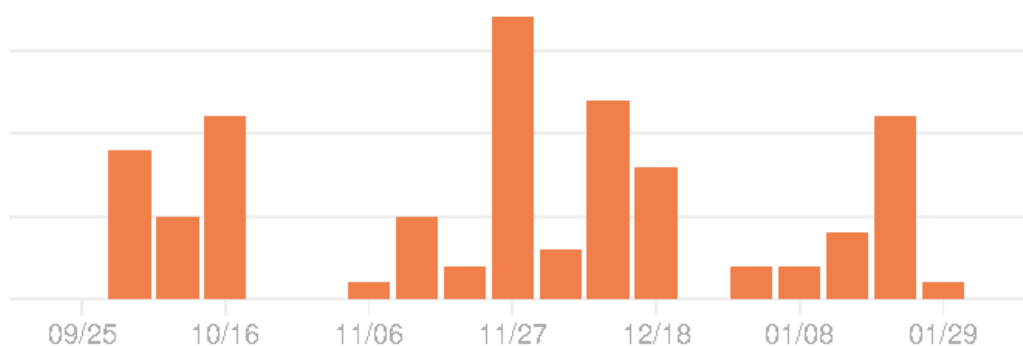


FIGURE 2.1 – *Evolution du nombre de commits par semaine*

---

1. <https://github.com/>

# Chapitre 3

## Implémentations

### 3.1 Partition

#### 3.1.1 Durée des notes

La durée des notes est représentée dans le programme comme dans le domaine de la notation musicale, c'est-à-dire qu'une Ronde se note 1, une Blanche 2, une Noire 4, etc.

Sachant que le programme représente les durées de base de la Ronde jusqu'à la Quadruple-Croche, il est alors possible de récupérer la durée réelle d'une note grâce à la formule :

$$Real\_Duration(duration) = \frac{64}{duration}$$

```
1 typedef enum {
2     RONDE = 1,          BLANCHE = 2,
3     NOIRE = 4,          CROCHE = 8,
4     DOUBLECROCHE = 16,  TRIPLECROCHE = 32
5     QUADRUPLECROCHE = 64
6 } Note_Duration;
```

### 3.1.2 Hauteur des notes

La hauteur des notes suit la spécification MIDI mais commence à l'octave 0 jusqu'à la fin de l'octave 8 :

Octave	Note Numbers											
	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
-1	0	1	2	3	4	5	6	7	8	9	10	11
0	12	13	14	15	16	17	18	19	20	21	22	23
1	24	25	26	27	28	29	30	31	32	33	34	35
2	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59
4	60	61	62	63	64	65	66	67	68	69	70	71
5	72	73	74	75	76	77	78	79	80	81	82	83
6	84	85	86	87	88	89	90	91	92	93	94	95
7	96	97	98	99	100	101	102	103	104	105	106	107
8	108	109	110	111	112	113	114	115	116	117	118	119
9	120	121	122	123	124	125	126	127				

### 3.1.3 Structure d'une note

Une note est représentée par sa hauteur, sa durée, des drapeaux (Altérations, note pointée, liaison). Ici un silence est représenté de la même manière (certains champs sont alors ignorés). La structure suivante d'une note combinant ces deux aspects est donc utilisée :

```
1 struct Note
2 {
3     char note;           /* Hauteur */
4     char rest;           /* Note / Silence */
5
6     Note_Flags flags : 24; /* Drapeaux */
7     Note_Duration duration : 8; /* Duree */
8 };
```

### 3.1.4 Représentation d'une mesure

Une mesure est représentée par un type de mesure (4/4, 3/4 , ...), une clé, des drapeaux, une armure et une suite de note structurée comme une liste chaînée simple. Au départ, une mesure est initialisée avec uniquement des silences (les plus grands possibles).

La durée totale de la mesure (num \* den) est conservée à l'ajout et à la suppression de note (qui équivaut à changer le champ rest d'une note).

Voici la structure pour représenter une mesure :

```
1 struct Step
2 {
3     ToNote *notes;           /* Liste chainees */
4     int num;                 /* Nombre de temps */
5     Cle cle : 16;            /* Cle */
6     Note_Duration den : 8;   /* Duree de base */
7     Step_Flags flags : 16;   /* Drapeaux */
8     signed char sign;        /* Armure */
9 };
```

L'armure est constituée comme ceci :

- Si **sign** vaut 0, alors la tonalité est de DoM/Lam : l'armure est vide.
- Si **sign** est négatif, alors l'armure est constituée de  $|\text{sign}|$  bémols.
- Si **sign** est positif, alors l'armure est constituée de  $\text{sign}$  dièses.

### 3.1.5 Portée et Partition

Une portée n'est qu'un tableau de plusieurs mesures les unes à la suite des autres. En effet, la suppression/déplacement d'une mesure est une opération coûteuse en performance mais rare dans cette utilisation.

De la même manière, une partition est un tableau de plusieurs portées. Il est à noter qu'il est évident que chaque portée doit avoir le même nombre de mesures : Ceci est vérifié par les fonctions d'ajouts/suppressions de mesures qui permettent de garder cet état vrai.



## 3.2 Audio

### 3.2.1 Principe

Il est à noter qu'ici, en aucun cas le programme n'utilise la librairie `SDL_mixer` qui faciliterait l'utilisation de sons MP3/WAV externes mais qui empêcherait la configuration des fréquences jouées.

Le programme travaille avec des signaux de caractéristiques suivantes :

- **Un taux d'échantillonnage** de 44100Hz (qualité CD) : Une seconde d'un signal audio est divisée en 44100 valeurs.
- des valeurs en **16 bits signés**
- **2 Channels** (Gauche / Droite)
- Un buffer audio **de 256 valeurs**.

### 3.2.2 Fréquences de base

Pour calculer la fréquence de base d'une note dans la gamme tempérée, le programme utilise la formule suivante :

$$f(h) = 33 \times 2^{\left(\frac{h}{12} - 2\right)} \times 1.059463^{(h \bmod 12)}$$

### 3.2.3 Remplissage du buffer et Mixage

La fonction `main_callback()` du fichier `Audio.c` se charge de remplir le buffer une fois que celui-ci est vide.

Dans un premier temps, cette fonction va faire appel à une fonction de mixage. Celle-ci va considérer toutes les notes jouées en cours et additionner leur signal et normaliser le signal résultant.

Ce signal de 256 valeurs va ensuite être copié dans le buffer pour être joué.

Voici comment calculer un signal à partir d'une fréquence :

$$\begin{aligned}
sinusoide(x, frequency) &= \sin(\pi \times frequency \times \frac{x}{44100}) \\
carre(x, frequency) &= \lfloor sinusoide(x, frequency) \rfloor \\
fusion(x, frequency) &= carre(x, frequency) + sinusoide(x, frequency) \\
harmonic(x, frequency) &= \sum_{i=0}^4 carre(x, frequency \times 2^i)
\end{aligned}$$

FIGURE 3.1 – *Génération des signaux*

### 3.2.4 Lecture

Un thread est créé pour chaque portée. Ainsi, chaque thread va analyser la mesure suivant, trouver la note à jouer et l'envoyer au mixeur du processus parent. Les threads utilisés sont les `SDL_Thread` gérés automatiquement par la librairie.

# Chapitre 4

## Eléments Techniques

### 4.1 Compilation et Lancement

Pour compiler le projet, un Makefile est fourni. Il créera le programme **med**. Pour lancer la compilation, il suffit de lancer la commande **make** et pour retourner à un dossier propre, d'utiliser la commande **make clean**, ce qui supprimera l'exécutable et les fichiers temporaires liés à la compilation :

```
make    #Compilation
make clean #Nettoyage
```

Pour lancer l'application il suffit d'exécuter le programme qui vient d'être compilé. Il est aussi possible d'importer un fichier MED ou ABC directement en ligne de commande :

```
./med #Lancement simple
./med file.med #Lancement avec ouverture du fichier MED file.med
./med -abc file.abc #Lancement avec importation du fichier ABC file.abc
```

### 4.2 Format MED

Le format MED est le format propre du projet. C'est un format binaire qui se base sur la recopie octet par octet de la mémoire dans un fichier. Cela permet une sauvegarde et une ouverture rapide.

Ce format est un fichier non compressé, il est donc relativement lourd (quelques ko à 5-6Mo en fonction de la taille de la partition) puisque la plupart des octets du fichier sont à 0 et que le programme autorise la redondance d'informations en mémoire.

## 4.3 Format ABC

Le programme permet d'importer ou d'exporter des fichiers au format ABC. Étant donné que c'est un format très libre et très permissif, seul le format ABC 2.0<sup>1</sup> peut être importé. Mais vu qu'il n'y a pas moyen de différencier les versions, le programme tentera quand même d'ouvrir tout type de fichier ABC.

Toutefois, toutes les fonctionnalités ne sont pas supportées et le programme commence par modifier la syntaxe du fichier en entrée (simplification et création d'un nouveau fichier temporaire), puis il tente de le parser.

Voici le format idéal reconnu :

Le format se compose d'abord d'un header :

```
X:7959 % Identifiant unique de melodie (obligatoire)
T:Titre % Titre
C:Compositeur % Compositeur
M:3/4 % Mesure
L:1/8 % Duree de base pour une note
Q:1/4=90 % Tempo
K:F % Tonalite
```

Puis le corps du format avec les informations sur les notes :

```
V:1 % Numero de voix (1 = 1ere voix)
c f2 a/ g/ g/ f/ e/ f/ | c g2 b/ a/ a/ g/ ^f/ g/ |
V:2 % Numero de voix (2 = 2eme voix)
F, C F, C F, C | E, C E, C E, C |
```

---

1. Documentation ABC 2.0 : <http://abcnotation.com/wiki/abc:standard:v2.0>

Voici quelques significations des symboles présentés ci-dessus et qui peuvent être retrouvées dans la documentation :

Symbole	Signification
abcdefg	Nom des notes
ABCDEFGG	Nom des notes un octave plus bas
	Séparation des mesures
z ou x	Silence
V:chiffre	Numéro de voix de la prochaine ligne (maximum 10)
/[nombre]	Divise la durée de base de la note
[nombre]	Multiplie la durée de base de la note
,	Diminue d'un octave
,	Augmente d'un octave
^	Ajout d'un dièse sur la prochaine note
=	Ajout d'un bécarré sur la prochaine note
—	Ajout d'un bémol sur la prochaine note

## 4.4 Bugs connus

Voici en vrac, dans un ordre non particulier, une liste de bugs, de comportements non souhaités ou de fonctionnalités non mis en place :

- Problème d'importation des fichiers ABC écrits sous Windows/Mac avec le retour à la ligne `\r\n` ou `\r`.

...

# Chapitre 5

## Manuel d'utilisation

### 5.1 Interface générale

L'interface se décompose en trois parties :

- **Le menu** : Au clic de la souris pour ouvrir les sous-menus, on peut Créer, Ouvrir ou Sauvegarder une partition dans un fichier, ou encore effectuer des actions sur la partie. Certaines actions demandent à ce qu'un élément soit sélectionné (Mesure, Note)
- **La barre de toolbar** : Situé juste en dessous du menu, elle permet de choisir le mode : A pour "Ajout", E pour "Édition"; de régler la durée des notes, les altérations, le tempo et est également composé d'un bouton Play/Pause pour jouer la partition en cours
- **Le corps** : Il permet de visualiser la partition. On peut la déplacer (Clic / Move), de zoomer/dézoomer, de sélectionner un élément, d'ajouter des notes, de supprimer des éléments, etc.

#### 5.1.1 Le mode Ajout

Le mode **Ajout** est activé si la lettre **A** est affichée dans la barre de Toolbar toute à gauche.

Ce mode permet l'ajout de notes sur la partition. Une fois ce mode activé, des zones bleutées apparaissent sur la partition au survol de la souris. Lors du clic dans une de ces zones, la note est rajouté au bon emplacement et à la hauteur du clic. On peut configurer les caractéristiques de la note ajoutée au préalable en sélectionnant sa durée, son altération dans la Toolbar.

Il est également possible de déplacer la partition, de zoomer.

### 5.1.2 Le mode Edition

Le mode **Edition** est activé si la lettre **E** est affichée dans la barre de Toolbar toute à gauche.

Il permet de sélectionner un élément : une fois sélectionné, un rectangle légèrement transparent rouge indique de la bonne sélection d'une note, et un rectangle bleu indique de la sélection d'une mesure.

Une fois un élément sélectionné on peut en fonction de son type :

Pour une mesure :

- Supprimer la mesure avec Selection->Supprimer ou SUPPR : Supprime toutes les mesures en parallèle sur les autres portées.
- Régulariser la mesure (fusionner les silences quand cela est possible) avec Selection->Régulariser.
- Ajout des mesures avant ou après avec Ajouter->Mesures
- Changer l'armure avec Selection->Armures

Pour une note/silence :

- Supprimer la note avec Selection->Supprimer ou SUPPR (La transforme en silence de même durée)
- Diviser un silence en deux parties égales avec Selection->Diviser ou la touche d
- Changer la durée de la note avec la Toolbar
- Rajouter / Enlever une altération avec la Toolbar

## 5.2 Explorateur de fichier

Un explorateur de fichiers du disque est disponible. On peut notamment le retrouver lors de l'ouverture d'un fichier (Fichier->Ouvrir), ou lors de la sauvegarde (Fichier->Export en MED).

Seul les dossiers, les fichiers .med et les fichiers .abc sont affichés étant donné que seuls ces fichiers sont supportés par le programme.

Pour ouvrir un dossier, il suffit de double-clic dessus, pour sélectionner un fichier, un simple clic suffit et le colorise en bleu.

Le bouton *Parent* permet de monter d'un répertoire tandis que le bouton **Valider** valide le choix du fichier ou ne fait rien si aucun fichier n'est sélectionné.

Une fois un fichier validé, le programme détermine avec l'extension si c'est un fichier ABC ou MED et tente de l'ouvrir. Un message d'erreur s'affiche en cas d'échec.



# Chapitre 6

## Perspectives

Je compte bien évidemment continuer le développement du programme, qui n'est pour le moment qu'un prototype. Parmi les prochaines améliorations intéressantes qui pourraient avoir lieu :

- Importation / Exportation des fichiers MIDI : Ces fonctionnalités, proposées en tant qu'améliorations à la base n'ont pas été intégrées notamment par manque de temps. En effet, la spécification MIDI est complexe et il aurait été trop compliqué de mettre en place un système fiable dans les temps.
- Amélioration du format MED avec une compression : Par exemple, un codage de Huffman et une compression RLE. En effet, ce format contient (généralement) beaucoup de motifs communs et la compression pourrait être performante.
- Utilisation du format Soundfont et de FluidSynth (libre) pour jouer des sons. A la base proposé en tant qu'améliorations, cette méthode ne permet pas de générer un son de fréquence libre mais permet l'utilisation simple de sons d'un très grand réalisme.
- Enfin, l'ajout des diverses fonctionnalités manquantes, par rapport à MuseScore.