Editeur musical avec lecture audio

et génération analogique d'ondes sinusoidales et carrées

Victor Veillerette

Janvier 2017

Table des matières

Chapitre 1

Introduction

1.1 Sujet initial

1.2 Analyse des fonctionnalités

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 librairies suivantes :

— **SDL-1.2** : La librairie *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 simple avec la commande

make

— **Rapport.pdf** : Ce même rapport.

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**[1].

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

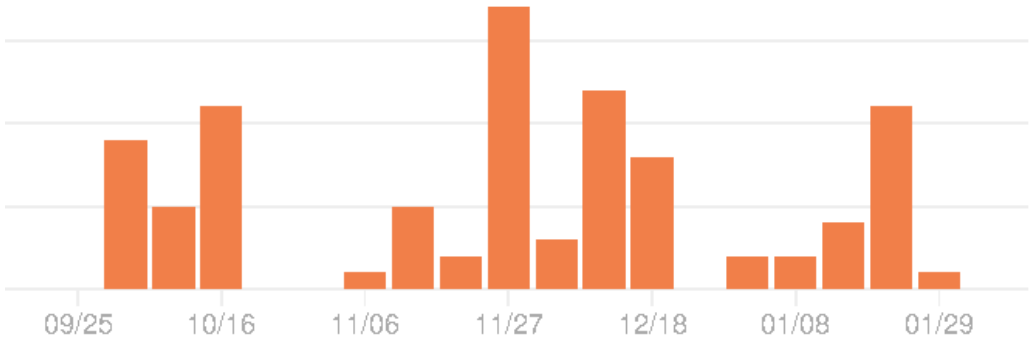

FIGURE 2.1 – *Evolution du nombre de commits par semaine*

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 :

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

```
typedef enum {
        RONDE = 1,              BLANCHE = 2,
        NOIRE = 4,              CROCHE  = 8,
        DOUBLECROCHE = 16,      TRIPLECROCHE = 32
        QUADRUPLECROCHE = 64
        } 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 :

```
struct Note
{
        char note;      /* Hauteur */
        char rest;      /* Note / Silence */

        Note_Flags flags : 24;  /* Drapeaux */
        Note_Duration duration : 8;     /* Duree */
};
```

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 :

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

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 |sign| bémols.

— Si **sign** est positif, alors l'armure est constituée de 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'échantillonage** de 44100Hz (qualité CD) : Une seconde d'un signal audio est divisé 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 :

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

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 :

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

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 [1] 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
ABCDEFG	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 bécarre 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**.

— ... Bla ... Bla

Chapitre 5

Manuel d'utilisation

5.1 Interface générale

5.2 Explorateur de fichier

5.3 Configuration Audio

Chapitre 6

Bilan

6.1 Conclusion

6.2 Perspectives