Xavier Lambein
54621300

Rapport : DJ'Oz

Victor Lecomte
65531300

LFSAB1402

1 Structure et Conception

2 Limitations

3 Complexité

4 Extensions

Comme extensions, nous avons choisi d'ajouter des enveloppes sonores aux échantillons, la gestion des instruments, et une composition personnelle.

4.1 Enveloppes sonores

Afin de d'éviter des sauts désagréables entre les notes, et d'obtenir une qualité sonore plus lisse, nous avons décidé d'utiliser des enveloppes sonores, qui adoucissent le début et la fin des échantillons.

Dans le code, nous avons défini ces enveloppes comme des fonctions qui renvoient un facteur de volume pour chacune des positions de l'échantillon. Plus de détails sur l'implémentation se trouvent dans la section 1.

Nous avons essayé plusieurs types d'enveloppes :

— **Enveloppe en trapèze :** cette enveloppe prend deux paramètres : *attack* et *release*.

Au début de l'échantillon, le volume va augmenter linéairement pendant un temps *attack* avant d'atteindre son niveau de régime, puis à la fin il va diminuer linéairement pendant un temps *release* jusqu'à un volume nul.

Cette enveloppe est simple et permet de se débarasser du bruit de coupure, mais elle est assez peu paramétrable. Elle est implémentée par la fonction `EnvTrapezoid`.

— **Enveloppe ADSR :** analogue à la méthode du trapèze, elle accepte les quatre paramètres *attack*, *decay*, *sustain* et *release*.

Comme pour le trapèze, au début de l'échantillon, le volume augmente linéairement pendant un temps *attack* jusqu'à atteindre un volume maximal. Mais après, le volume décroit de nouveau linéairement, pendant un temps *decay*, avant d'attendre le niveau de régime *sustain*, inférieur au niveau maximal. À la fin, le volume va diminuer en un temps *release* comme pour le trapèze.

Cette enveloppe a plus de liberté dans l'attaque initiale, ce qui permet d'imiter avec plus de précision des instruments de musique. Elle est implémentée par la fonction `EnvADSR`.

— **Enveloppe en hyperbole :** Cette méthode est notre invention. Nous trouvions que les autres méthodes n'étaient pas assez souples : en effet, pour qu'elles puissent gérer les échantillons courts sans saut, il faut que les paramètres *attack* et *release* soient petits, ce qui implique un son assez dur, même sur les échantillons plus longs.

Pour éviter cela, il nous fallait donc une méthode qui s'adapte à la taille de l'échantillon, tout en gardant une dureté équivalente quelle qu'en soit la taille. Nous avons donc décidé de modéliser l'enveloppe comme un produit de deux hyperboles :

— l'une croissante, fixée à 0 au début et tendant vers 1 à la fin,

— l'autre décroissante, fixée à 0 à la fin et tendant vers 1 au début.

Elle s'exprime donc comme :

$$\text{Volume} = \frac{x}{x - \text{att}} \times \frac{L - x}{L - x + \text{att}}$$

où x est la position, L la longueur de l'échantillon et att un temps d'attaque.

Le grand avantage de cette enveloppe est que le domaine n'est pas séparé en sous-intervalles, ce qui évite des sauts non-intentionnels.

Un effet secondaire de cette méthode a été de diminuer le volume des courts échantillons quand leur longueur est trop proche du paramètre att. Nous avons résolu cela en multipliant l'expression par un facteur qui assure que l'amplitude au milieu de l'échantillon vaille 1.

C'est l'enveloppe que nous avons utilisé pour adoucir nos propres sons. Elle est implémentée par la fonction `EnvHyperbola`.

4.2 Instruments

4.3 Composition