Annexe 1

1 Un *flanger* en code

Pour créer un effet de *flange* (ou même de *chorus* ou d'échos transposés), il suffit simplement d'asservir le pointeur de lecture dans la mémoire-tampon à un oscillateur à basse fréquence (LFO). La gestion du délai moyen, de la profondeur ainsi que de la fréquence de l'oscillateur permettra de naviguer d'un effet à l'autre.

1.1 Un oscillateur à basse fréquence de forme sinusoïdale

Pour construire un oscillateur à basse fréquence, il suffit simplement de lire une fonction sinusoïdale à une fréquence donnée, de multiplier chacun des échantillons par la profondeur désirée et de leurs additionner une valeur centrale (délai moyen). On lit une fonction sinusoïdale en lui donnant une phase entre 0 et 2π.

L'incrément de la phase (la valeur qui lui est ajoutée à chaque instant d'échantillonnage) est calculé en fonction de la fréquence f et de la fréquence d'échantillonnage f_s.

$$inc = f/f_s$$

Ensuite, on lit la fonction sinusoïdale et on incrémente la phase courante ph à chaque période d'échantillonnage :

$$\begin{aligned} lfo &= \sin(2\pi ph) \\ ph &= ph + inc \\ ph &>= 1 \ ? \ ph = ph - 1 \end{aligned}$$

La fonction mathématique *sin* retourne une valeur entre -1 et 1. Afin d'obtenir des valeurs parcourant l'ambitus désiré, on multiplie la sortie de l'oscillateur par la profondeur (a) puis on lui ajoute la valeur centrale (c), c'est-à-dire le délai moyen :

$$lfo = lfo * a + c$$

L'équation généralisée se lit comme suit :

$$\begin{aligned} lfo &= a * \sin(2\pi ph) + c \\ ph &= ph + inc \\ ph &>= 1 \ ? \ ph = ph - 1 \end{aligned}$$

Le programme suivant illustre et commente la construction d'un LFO de forme sinusoïdale en langage JSFX :

```
/* LFO de forme sinusoidale. */
desc: LFO

slider1:5<0.1,10>Delai Moyen in ms
slider2:0.5<0,1>Profondeur du LFO
slider3:0.1<0.01,2>Frequence du LFO en Hz

@init
/* Initialisation de la phase de l'oscillateur. */
lfoPos = 0;

@slider
/* Delai moyen (valeur moyenne du LFO), valeur
   ajoutee a chaque echantillon de l'oscillateur. */
cdelay = slider1 / 1000 * srate;
/* Profondeur (ou amplitude) du LFO, en fonction du delai moyen.
   Cette valeur multiplie chaque echantillon de l'oscillateur. */
depth = slider2 * 0.99 * cdelay;
/* Increment de la phase de l'oscillateur, c'est-a-dire
   de combien elle doit avancer a chaque instant d'echantillonnage
   pour lire la forme d'onde "Hz" fois par seconde. */
lfoinc = slider3 / srate;

@sample
/* lfoPos va de 0 a 1 "Hz" fois par seconde. sin(2*pi*lfoPos)
   genere un lfo sinusoidal entre -1 et 1. On multiplie par
   la profondeur puis on additionne le delai moyen (pour
   osciller autour de ce dernier). */
lfo = depth * sin(2 * $pi * lfoPos) + cdelay;
/* Incremente la phase. */
lfoPos += lfoinc;
/* Si plus grand ou egale a 1, on rammene entre 0 et 1. */
lfoPos >= 1 ? lfoPos -= 1;
```

1.2 Un oscillateur à basse fréquence de forme triangulaire

Pour construire un LFO triangulaire (parfois utile pour obtenir des transpositions constantes), on procède de la façon suivante. On génère d'abord une phase entre 0 et 4, l'incrément vaut donc (en fonction de la fréquence f et de la fréquence d'échantillonnage f_s) :

$$inc = 4 \times f/f_s$$

Ensuite, on prend la valeur la plus petite entre la phase courante et la phase courante inversée $(4 - ph)$, ce qui donne un triangle entre 0 et 2 :

$$min(ph, 4 - ph)$$

Finalement, on soustraie 1 pour obtenir un LFO bipolaire (de -1 à 1). Voici la somme des opérations (a est la profondeur et c la valeur centrale) :

$$
\begin{aligned}
lfo &= min(ph, 4 - ph) - 1 \\
lfo &= a * lfo + c \\
ph &= ph + inc \\
ph &>= 4 \ ? \ ph = ph - 4
\end{aligned}
$$

1.3 Le *plugin* de *flanger*

Voici le code complet d'un *plugin* de *flanger* :

```
/* Un flanger recursif. */
desc: Flanger

slider1:5<0.1,10>Central Delay in ms
slider2:0.5<0,1>LFO Depth
slider3:0.1<0.01,2>LFO Speed in Hz
slider4:0<0,1>Feedback

@init
/* 2 fois le delai moyen d'espace-memoire. */
maxlen = ceil(0.02 * srate);
bufL = 0;
bufR = bufL + maxlen + 1;
curPos = lfoPos = 0;

@slider
/* Calcul du delai moyen. */
cdelay = slider1 / 1000 * srate;
/* Calcul de la profondeur, en fonction du delai moyen. */
depth = slider2 * 0.99 * cdelay;
/* Calcul de l'increment du LFO, en fonction de la frequence. */
lfoinc = slider3 / srate;

@sample
/* LFO sinusoidal. */
lfo = depth * sin(2 * $pi * lfoPos) + cdelay;
/* Incremente la phase du LFO. */
lfoPos += lfoinc;
/* Si plus grand que 1, rammene entre 0 et 1. */
lfoPos >= 1 ? lfoPos -= 1;

/* La position de lecture dans l'espace-memoire est la
   position courante - la valeur instantannee du LFO. */
```

```
pos = curPos - lfo;
pos < 0 ? pos += maxlen;
ipart = floor(pos);
frac = pos - ipart;
readL = bufL[ipart] + (bufL[ipart+1] - bufL[ipart]) * frac;
readR = bufR[ipart] + (bufR[ipart+1] - bufR[ipart]) * frac;
bufL[curPos] = spl0 + readL * slider4;
bufR[curPos] = spl1 + readR * slider4;
/* Necessaire pour l'interpolation lineaire quand le
   pointeur de lecture lit a la toute fin de la memoire.
   Le premier echantillon dans la memoire doit etre copie
   a la position tout juste apres l'espace alloue (maxlen). */
curPos == 0 ? (
  bufL[maxlen] = bufL[0];
  bufR[maxlen] = bufR[0];
);
curPos += 1;
curPos >= maxlen ? curPos = 0;
spl0 = (spl0 + readL) * 0.7;
spl1 = (spl1 + readR) * 0.7;
```